F5 LABS
2018

APPLICATION PROTECTION REPORT

AUTHOR

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.

CONTRIBUTORS

Debbie Walkowski
Threat Research Evangelist, F5

David Holmes
Principal Threat Research Evangelist, F5

Sara Boddy
Director, F5 Labs

Justin Shattuck
Principal Threat Research Evangelist, F5

BUSINESS AND DATA PARTNERS

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

PONEMON INSTITUTE conducts independent research on privacy, data protection, and information security policy. For this report, we drew on results derived from two separate surveys of security professionals, conducted on behalf of F5.

WHATCOM COMMUNITY COLLEGE CYBERSECURITY CENTER Special thanks to Seaver Milnor and Christy Saunders, both members of the computer information systems faculty at WCC, who performed an extensive review, analysis, and categorization of breach notification records filed with the attorney generals' offices in California, Washington, Idaho, and Oregon.

WHITEHAT SECURITY is dedicated to helping enterprises ensure safe digital experiences by securing their applications. For this report, we cross-referenced much of the data collected from sources mentioned above with WhiteHat vulnerability data.

UNIVERSITY OF WASHINGTON TACOMA graduate students assisted with background research and data source analysis as part of their Masters of Cybersecurity and Leadership Capstone research project.

F5 LABS
2018

APPLICATION PROTECTION REPORT

TABLE OF CONTENTS

TABLE OF CONTENTS


01

EXECUTIVE SUMMARY

Like coral reefs, teeming with a variety of life, Web applications are “colony creatures.” They consist of a multitude of independent components, running in separate environments with different operational requirements and supporting infrastructure (both in the cloud and on premises) glued together across networks. In this report, we examine that series of interacting tiers—application services, application access, Transport Layer Security (TLS), domain name services (DNS), and the network—because each one is a potential target of attack.

To get an objective viewpoint on how applications are being attacked, F5 Labs looked at data from a variety of sources, including our own internal datasets, WhiteHat Security vulnerabilities, Loryka attack data, and a Ponemon security survey of IT professionals commissioned by F5.

In addition, we worked with faculty from the Whatcom Community College Cybersecurity Center to perform an extensive review of breach notification records in California, Washington, Idaho, and Oregon. (In each U.S. state, it is the state attorney general's office that oversees and is responsible for enforcing state breach disclosure laws and notifications. Because of this role, some states publish data breach notification letters.)

In these four states, we analyzed 301 breaches in 2017 and Q1 2018 and found that web application attacks were the top cause of all reported breaches at 30%. Earlier research done by F5 Labs into 433 major breach cases spanning 12 years and 26 countries found that applications were the initial targets in 53% of breaches.

Protecting applications has always been a critical task and will continue to be in the future. But, what do CISOs need to know now?

WHAT APPS DO WE USE AND WHERE ARE THEY?

THE MAJORITY OF ORGANIZATIONS HAVE LITTLE CONFIDENCE IN THEIR ABILITY TO KEEP TRACK OF ALL THEIR APPLICATIONS.

Our F5 Ponemon survey, *Web Application Security in the Changing Risk Landscape: Global Study*, found that a majority of organizations have little confidence in their ability to keep track of all their applications. Thirty-eight percent of respondents said they had "no confidence" in knowing where all the applications were in their organization. Yet, at the same time, respondents reported that 34% of their web applications were mission critical. Among the most commonly used web apps were backup and storage (83%), communication applications like email (71%), document management and collaboration (66%), and apps in the Microsoft Office suite (65%).

HOW COULD APPLICATION ATTACKS AFFECT MY ORGANIZATION?

When apps are attacked there are many different impacts. Denial of service was the most painful for many organizations, with 81% of respondents rating loss of availability at 7 out of 10 on a scale of 1 to 10 (10 being highest). Breach of confidential or sensitive information (such as intellectual property or trade secrets) placed second with 77% of survey respondents rating it 7 to 10. Similarly, 73% of respondents rated tampering with an application at 7 to 10. Finally, 64% of respondents rated the loss of personally identifiable information (PII) of their customers, consumers, and employees at 7 to 10.

WHAT ARE THE SIGNIFICANT RISKS?

Within the 2017 and Q1 2018 breach notification letters from the states' attorneys general, we examined web attacks in detail. Specific application breaches included payment card theft via web injection (70%), website hacking (26%), and app database hacking (4%). We cross-referenced this with the relevant WhiteHat Security vulnerabilities, Loryka attack surveillance, and known exploits published by Exploit-DB, a CVE-compliant archive of public exploits[1] and corresponding vulnerable software,[2] to give us an idea of the most significant new risks.

70%

THE HIGHEST PERCENTAGE OF BREACH REPORTS FOR Q1 2018 WERE WEB INJECTIONS THAT STOLE PAYMENT CARD INFORMATION.

Injection attacks against app services

The highest percentage (70%) of the breach reports for Q1 2018 were web injections that stole customer payment card information. Injection attacks allow an attacker to insert commands or new code directly into a running application (also known as tampering with an app) to pull off a malicious scheme. Over the past decade, 23% of breach records involved SQL injection attacks, the most infamous type of injection attack. Injection vulnerabilities (weaknesses that have not yet been exploited) are prevalent, as well. WhiteHat Security reported that 17% of all discovered vulnerabilities in 2017 were injection vulnerabilities. This problem is so significant that injection flaws are rated as the number one risk to applications on the OWASP Top 10 2017 list. For this reason, high priority should be given to finding, patching, and blocking injection vulnerabilities.

Account access hijacking

Breach records analysis shows that 13% of all web app breaches in 2017 and Q1 2018 were access-related. These broke down as follows: credentials stolen via compromised email (34.29%), access control misconfiguration (22.86%), brute force attacks to crack passwords (5.71%), credential stuffing from stolen passwords (8.57), and social engineering theft (2.76). Nearly 25% of the web app Exploit-DB scripts were also found to be access-related. The F5 Ponemon security survey showed that 75% of respondents were only using username and password for application authentication to critical web applications. For any important application, stronger authentication solutions such as federated identity or multi-factor should be considered. For external applications over which you don't have full control, a cloud access security broker (CASB) can consolidate and augment authentication.

Deserialization attacks against app services

In 2017, deserialization attacks were somewhat low in number but vast in impact. The Apache Struts deserialization injection vulnerability was the hole that attackers used to breach Equifax and steal the identities of 148 million Americans and 15.2 million UK citizens.[3] Serialization occurs when an app converts its data into a format for transport; deserialization is the process of converting that data back again. These attacks are becoming more common because applications are now networked clusters of subsystems that require data-serialized communication streams. Attackers embed commands in the serialized data stream and pass them unfiltered directly into the heart of application engines. Thirty Exploit-DB scripts are related to deserialization. Applications should scan and filter all user inputs, including serialization data streams.

Attacks against transport layer protection

While 63% of survey respondents said they always use SSL/TLS for their web applications, only 46% of survey respondents said they use SSL/TLS encryption for the majority (76-100%) of their applications. With so many transport layer encryption standards (such as SSL and TLS 1.0) still in use, even though they've been retired as "broken," there's an ongoing risk of eavesdropping or man-in-the-middle hijacks from attackers. Furthermore, 47% of organizations said they use self-signed certificates, which reduces the trustworthiness of their applications. Organizations need to ensure all applications are running acceptable levels of encryption and have proper third-party signed certificates in place.

Denial-of-service attacks against any component of the app

Denial-of-service attacks can strike in many ways, sometimes directly against a discovered flaw in software. Exploit-DB has 5,665 denial-of-service exploits in its database. More commonly, we see a distributed denial-of-service (DDoS) deluge from an army of attacker-controlled devices, or thingbots, with direct or amplified/reflected traffic that overloads an application. Even more dangerous is a hybrid attack that combines traffic floods with pre-targeted assaults against vulnerabilities in application services. These attacks are sized and custom-configured to manipulate web application infrastructure, stressing the site to its limits. F5 has seen attacks like this from hundreds of thousands of separate IP addresses with over 2,000 page requests per minute. DDoS attacks are pervasive across all levels of the application tier, so it's critical that every organization have a DDoS response strategy.

5,665

EXPLOIT-DB HAS 5,665 DENIAL-OF-SERVICE EXPLOITS IN ITS DATABASE.

Scripting attacks against clients to hijack access

Attacks against app clients are often underreported because they target individuals, who are unlikely to be mentioned in a publicized breach report, and there are no regulatory reporting mandates like those that exist for application breaches. A common way a client is hijacked is via cross-site scripting (XSS), which is one of the most prevalent vulnerabilities (30% of 2017 WhiteHat Security vulnerabilities, 9.24% of Exploit-DB scripts). XSS attacks can often lead to stolen user credentials or hijacked access. Cross-site request forgery (CSRF) is another way a client can be hijacked into unknowingly running unauthorized commands on a website. Both attacks involve a client app encountering malicious scripting code planted by an attacker somewhere on a website. Sites can help reduce scripting attacks by using web server options such as session cookies set to HTTP-only and domain restricted, as well as setting the X-frame-options to DENY.

Malware attacks against app clients

Clients are also attacked directly with malware that hijacks the browser to sniff or intercept the application authentication credentials. Malware that targets financial logins is quite common for both browser and mobile clients. While protection of the client device is something that has largely been ignored to date since it is difficult to control, tighter data privacy laws, such as the EU's General Data Protection Regulation (GDPR), are likely to come down on poorly written app clients. Some web application firewall solutions can watch for suspicious connections by detecting compromised clients and filtering their access.

AM I DOING ENOUGH TO PROTECT MY APPS? HOW DO I COMPARE WITH MY PEERS?

The F5 Ponemon security survey offers some insight into how other organizations are wrangling application security. The first question is one of ownership: 28% of respondents said the CIO or CTO owns responsibility for the application security risk management process. Only 10% of CISOs own it, yet they will be in the hot seat in the event of a breach.

The top three cited barriers to achieving a strong application security posture were "lack of visibility in the application layer," "lack of skilled or expert personnel," and "migration to the cloud environment." The first and third response both speak to analysis and instrumentation of the application tiers, which are solvable problems with scanning, monitoring, and collaboration with the development team.

AT 26%, WEB APPLICATION FIREWALLS WERE THE TOP MEANS FOR SECURING APPLICATIONS.

At 26%, web application firewalls were the top means for securing applications; others were application scanning (20%) and penetration testing (19%). Surprisingly, 26% of organizations do not deploy application hardening procedures, a useful way to bolster application security.

FOUR STEPS TO TAKE TO PROTECT APPLICATIONS

While all of these findings might seem to paint a bleak picture, taking these four steps will have a high impact on improving your application security and, for the most part, are not difficult to do.

1. Understand your environment.

Know what applications you have and what data repositories they access. Yes, determining this can be hard work, but it needs to be done. Focus on the apps your organization needs and the apps you build that your customers depend on. For the apps your organization needs, scan and inventory them regularly. For external applications your users depend on, a cloud access security broker (CASB) can be very helpful in counting and tracking app usage. For internal apps, it's essential to have a good relationship with the developer team to track down applications, future app plans, and development environments.

2. Reduce your attack surface.

Any part of an application service that is visible on the Internet, either directly or indirectly, will be probed for possible exploitation by attackers. This surface is broad, given an app's multiple tiers and the ever-increasing use of application programming interfaces (APIs) to share data with third parties. All exposed pieces should be access-controlled, patched, and hardened against attack. A good web application firewall (WAF)—our survey respondents' #1 tool of choice—can buy you time to do this. Some WAFs can perform "virtual patching" by scanning application traffic and blocking known exploit attacks. They know what to block from automatic signature updates from threat intelligence

feeds and vulnerability scans of your environment. This alleviates the time pressure to patch immediately when a new exploit is released, and it gives the operations team time to properly test and roll out fixes. Numerous preventable security incidents have occurred because security teams have not enabled the necessary security blocking features on their WAF. You should also segregate and partition your applications so that if a low-priority application gets breached, it can't become a conduit to reach the higher priority systems. This can be done in code, with server isolation, sandboxes, lower privileged users, and even with firewalls.

3. Prioritize defenses based on risk.

YOUR RISK ANALYSIS DOESN'T HAVE TO BE PERFECT, JUST BETTER THAN RANDOM GUESSING OR BIASED DECISION MAKING.

Once you know which applications are important and have minimized your attack surface, identify applications that need additional resources. Your risk analysis doesn't have to be perfect, just better than random guessing or biased decision making. So, use data to drive your risk strategy, figuring out what attackers would go after. Key data about your applications comes from security testing and scanning. Test internally developed code using internal scanners, code reviews, or a third party, which can give you an independent and knowledgeable perspective. With this information, you can properly assess the risk of any internally developed applications.

4. Select flexible and integrated defense tools.

You need a good but manageable selection of flexible, powerful solutions to cover controls for prevention, detection, and recovery from existing and emerging threats. Beyond the technical controls we've already mentioned—a web application firewall, a vulnerability scanning solution, and a CASB—protection needs to extend to all tiers that an application depends on. DNS servers should be well protected with DNS-savvy firewalls and be made highly available. Transport layer communications should be encrypted with a current acceptable standard of ciphers, and web servers should use HTTP Strict Transport Security (HSTS) to ensure encryption is fully covering all critical data flows. Security solutions should be consistent and well understood by security engineers. Many breaches occur despite these solutions simply because of a misunderstanding or misconfiguration of a product.

Based on the potential impacts of DDoS attacks, it's essential to protect your applications at the network, application, and infrastructure levels with on-premises scrubbing equipment or hosted solutions. The important thing is to size the solution based on the risk to your applications and the likely threats.

When it comes to protecting app clients, remember you have two classes of users: customers who access your apps and internal users who access apps on the Internet. To protect internal users, consider using federated identity or multi-factor authentication (MFA) for strong, reliable access control. For applications that only support username and password authentication (leaving your users open to access attacks like password guessing or stolen credentials), a CASB can help by consolidating and augmenting authentication for external apps. Protecting your customer's app client sessions should also be done for any high-value applications. Some of the more powerful and flexible WAF systems can help you protect customers' app client sessions by detecting bot attacks, brute-forcing, and logins from suspicious locations. This simple validation is great way to add another layer of protection for your customers as they access your apps.

WHAT DOES THE FUTURE OF APPLICATION PROTECTION LOOK LIKE?

Serverless applications

"Serverless" is a new way for developers to build web applications without having to worry about servers and the app's supporting infrastructure. The code and scripts developers write connect to APIs that directly trigger functions and services, instead of the traditional approach where the code runs within a server to call other servers. Serverless computing enables developers to focus on faster, more streamlined app development, and in the long run, serverless apps provide more flexibility and scalability. Just be aware that they are still vulnerable to the same kinds of attacks as traditional apps at every tier, especially against the primary user login pages and the APIs.

765

WITH THE AVERAGE ORGANIZATION USING 765 DIFFERENT WEB APPS, WE CAN EXPECT OUTSOURCING OF APPLICATION SECURITY TO GROW.

Outsourcing of application security

The lack of skilled or expert personnel for security, cited as a main barrier in the F5 Ponemon security survey, will only continue to get worse for application defenders. Furthermore, with the average organization using 765 different web applications, we can expect outsourcing of application security to grow, whether that means security functions such as anti-DDoS or web application security monitoring or moving to hosted platforms that provide security services as part of their offering. Outsourcing will require matching your specific security requirements to the outsourcer's capabilities, focus, and experience in application security.

Transport Layer Security improvements

Because Transport Layer Security (TLS) 1.3 is a radical departure from previous iterations of the protocol, the security community will struggle with adoption. The specter of quantum computing is also hanging over TLS in the longer term. It is our opinion that Transport Layer Security will suffer more severe shocks from an angle other than quantum computing, but those shocks are yet unknown. Organizations should keep an eye on browser support for TLS 1.3 and the major compliance standards regarding network encryption and updates in quantum computing.

SINCE MODERN APPS ARE COLLECTIONS OF SCRIPTS, LIBRARIES, SERVICES, AND DEVICES, WE HOPE TO SEE SECURITY TOOLS CATCH UP TO THIS PARADIGM.

Application protection that follows the tiers

Since modern applications are collections of scripts, libraries, services, and devices, we hope to see security tools catch up to this paradigm. In the future, developers will have a broader spectrum of secure components and frameworks to choose from in a way that is dramatically different and better than today's fragile and overly dependent ecosystem. Developers should ask for application frameworks that are "secure by default" with the capability to report security status and events in a standardized format. Defenders should look for security scanners that can continuously test application components in production in a safe but useful manner.

02

INTRODUCTION

Why the singular focus on applications in this report? Simple: they are the reason we use the Internet. Applications communicate, calculate, process, store, search, coordinate, and forecast for us. They are the muscles of business. So, they must work when we need them to, and they must work as expected.

APPLICATIONS ARE THE REASON WE USE THE INTERNET

Applications are also the containers of our data—they shape and hold the information we need. We feed data through and extract data from our applications. They are the gatekeepers and the interpreters of data. And since data is gold, applications are the repositories of our most valuable asset.

The 2018 F5 and Ponemon *Web Application Security in the Changing Risk Landscape: Global Study* found that the typical organization uses 765 web applications and, on average, 34% of them are considered mission critical. Survey respondents also estimated the average loss associated from a serious web application security incident at nearly eight million dollars.

ORGANIZATIONS SURVEYED CONSIDER 34% OF THEIR APPLICATIONS TO BE MISSION CRITICAL.

It's because of numbers like these (and the integral role applications play in business) that we have spent a year researching and compiling data about applications from both within and outside of F5 Labs. Our goal is to answer the following questions:

- **What constitutes an application?**
- **What are the threats to applications and how are they attacked?**
- **What does it take to protect apps?**

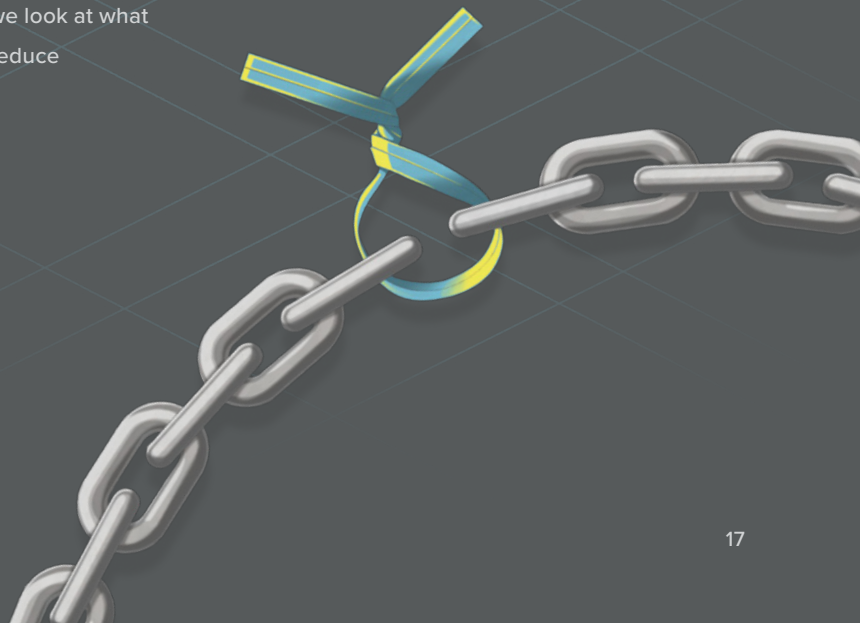
FIGURE 1: APPLICATIONS ARE THE BUSINESS

FIGURE 2: APPLICATIONS ARE THE GATEWAY TO YOUR DATA

These are difficult questions—ones that have not been fully answered. That's no surprise. Application security is complex and involves many different domains, including programming, infrastructure, supporting utilities, third-party dependencies, system operations, application users, and security controls. Any one of the links in this chain can become undone and lead to a breach in security. And we haven't even begun to factor in attackers and their evolving tactics and appetites for mayhem.

RESPONDENTS ESTIMATED THE AVERAGE LOSS FROM A SERIOUS WEB APPLICATION SECURITY INCIDENT AT NEARLY $8 MILLION.

Security in and of itself is meaningless; we need to consider risk to give it some context. We do this using good old-fashioned, thorough risk analysis. Yes, we know risk analysis involves some uncertainty. But, we are also aware that it's more useful to measure something than to guess and hope for the best. We explore how likely it is that threats will exploit vulnerabilities and create unwanted impacts. After this, we look at what kinds of security controls can reduce those risks. We hope you find insights in this report that will help you better defend your applications.



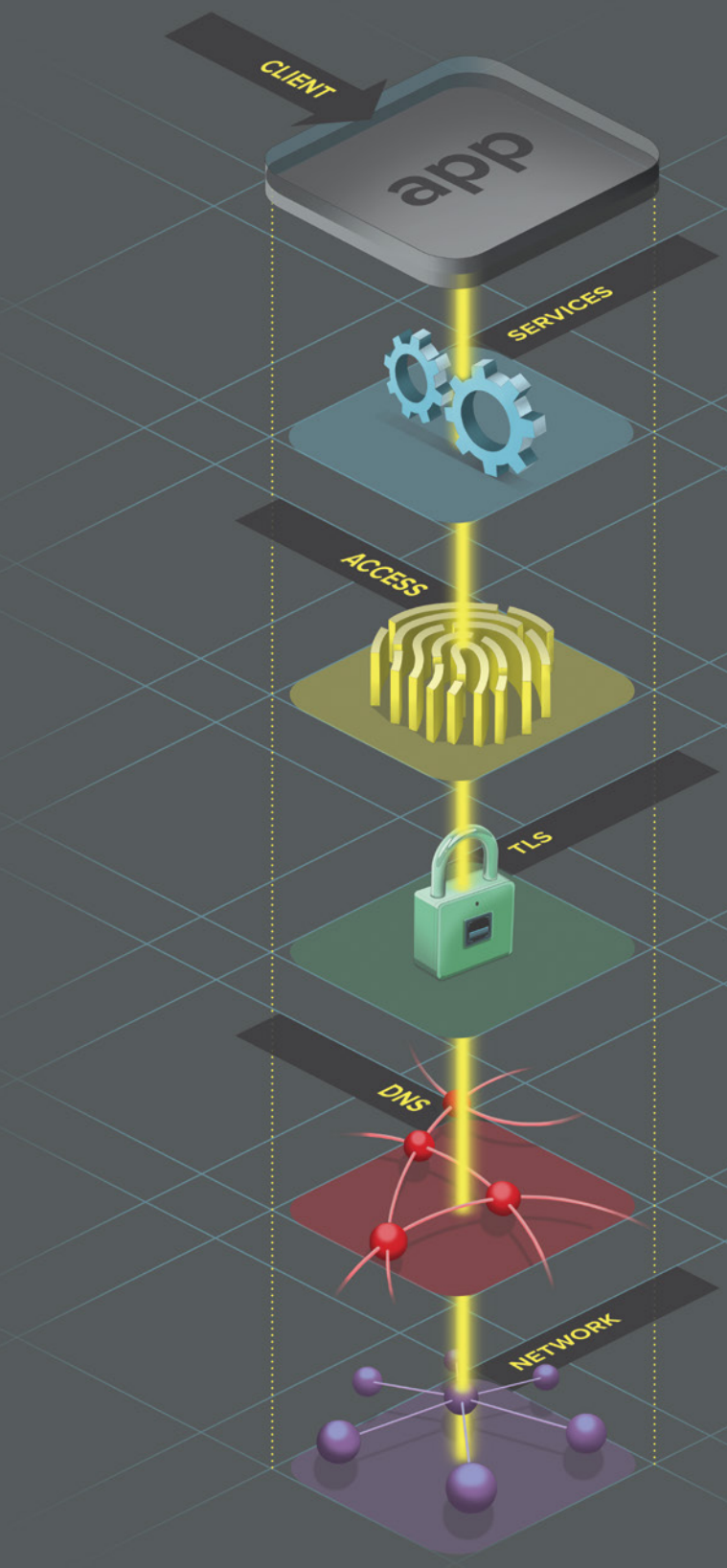


FIGURE 3: APPLICATION TIERS

WHAT IS AN APP?

Most of the web applications we use daily are "colony creatures." They consist of a multitude of separate, independent components, running in separate environments with different operational requirements and supporting infrastructure (both in the cloud and on premises) glued together over networks. With this in mind, we represent an app as a series of interacting tiers and sub-tiers *because each one is a potential target of attack*. We need to understand the individual attack surfaces so we can evaluate appropriate defenses.

Figure 3 provides an overview of the primary tiers of a typical web application: app services, TLS, DNS, and the network.

Within these tiers, there are sub-tiers and components, which are shown in Figure 4 (following page).

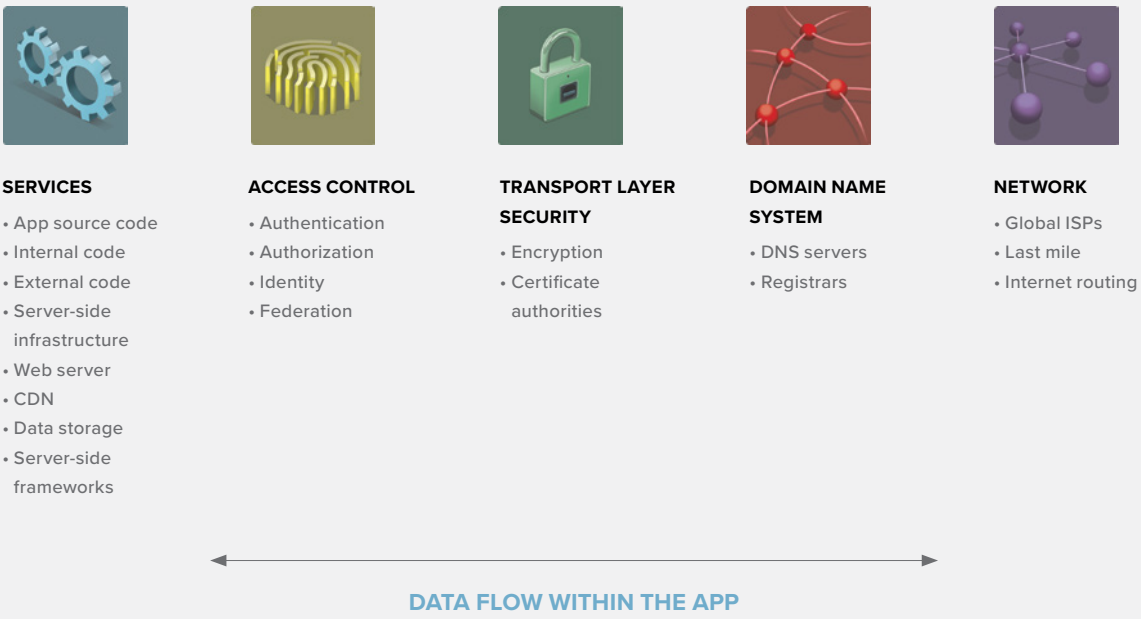


FIGURE 4: APPLICATION SUB-TIERS AND COMPONENTS

Following is a breakdown of the application tiers and their sub-tiers:

APPLICATION SERVICES TIER

When the Internet first began, websites were static HTML documents with hyperlinks published on a web server. Then came the Common Gateway Interface (CGI) standard with dynamic pages based on user input. Suddenly, dynamic web applications were born. This also opened the door for anonymous, untrusted users to inject malicious content into web pages. With that, web application hacking began to take off.

As noted in the 2016 TLS Telemetry Report from F5 Labs, the three most popular web servers are Apache, NGINX, and Microsoft Internet Information Server (IIS), with Apache being number one. These servers are the basis for web applications, but they also allow add-ons such as modules, plugins, libraries, frameworks, and extensions that add functionality. A typical web application often makes use of at least one or more web server add-ons in its architecture. This, too, increases complexity and broadens the attack surface of an application.

A TYPICAL WEB APP OFTEN USES WEB SERVER ADD-ONS IN ITS ARCHITECTURE—INCREASING COMPLEXITY AND BROADENING THE ATTACK SURFACE.

The services tier encompasses three sub-tiers: server-side infrastructure, server-side frameworks, and application source code, the latter of which we break down further as follows:

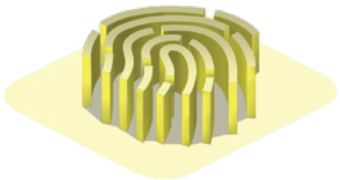
INTERNAL CODE

Internal code is the part of a web app that is unique to the app's specific function. It could be the core application (such as Microsoft SharePoint or Salesforce) or it could be the internal code developed or modified by an organization specifically for their needs.

EXTERNAL CODE

External code is the part of the app that refers to the reusable or third-party code that is linked to the core application. Examples include linked libraries, plug-ins, frameworks, server-side scripts, and external linked code. External code has usually had some level of testing, but also a lot of attackers scanning for holes, as well. Apache Struts falls into this category and is a prime example of where gaping holes appear when patches aren't kept up to date.

The server-side infrastructure sub-tier consists of the standalone servers that support an app. This includes things such as web servers and content delivery networks (CDNs) as well as app, database, and file servers. Server-side infrastructure is more monolithic yet more interchangeable than externally linked code so, in theory, it's easier to load balance and patch. Yet, it is often the server-side infrastructure, not the network, that is the direct target of app layer-focused (layer 7) DDoS attacks designed to knock down the supported app.

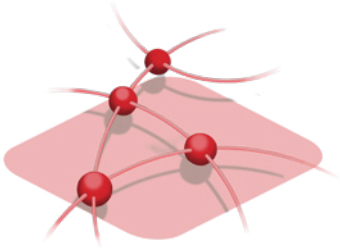


ACCESS CONTROL TIER

The application access control tier is the gateway that users pass through for authentication and authorization. Applications can deploy access control in many different ways. Client credentials are often stored in a database or apps can leverage shared on-premises solutions, for example, using a Lightweight Directory Access Protocol (LDAP) server. They can also connect to single sign-on (SSO) gateways either internally or externally, as in the case of federation services.

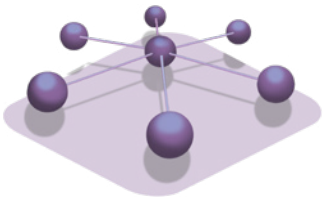
TRANSPORT LAYER SECURITY TIER

The transport layer tier provides encryption as network packets pass over untrusted networks like the Internet or convenience WiFi services. Encrypted encapsulation of packets usually happens close to the web application server and follows all the way to the client. Transport Layer Security (TLS) also ensures that attackers haven't tampered with the data in transit and verifies the application with a proper domain certificate from a trusted certificate authority. This tier includes the common HTTPS protocol, TLS, and the outdated SSL protocol.



DOMAIN NAME SYSTEM SERVICES TIER

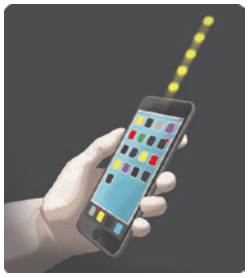
DNS—the "address book" of the Internet—is a globally distributed service running on agreed-upon standards of operation. Clients connecting to an application rely heavily on a functional and trustworthy DNS. If DNS is disrupted or worse, tampered with, applications can suffer severe security impacts. Since the apps themselves may need connectivity to other services outside the application's direct control, the app is also dependent on accurate and functional DNS. This tier includes all the DNS servers needed by a client and the app, as well as the relevant registrars of those domains.



NETWORK TIER

Clients need to connect to application servers, which almost always happens over the Internet. One of the most common protocols for web traffic, whether it's for public-facing websites or machine-to-machine API calls, is HTTP. In more security-savvy applications, these connections are encrypted using HTTPS.

The network tier also includes all application network services such as Internet Service Providers (ISPs), last-mile connections from ISPs to their customers' premises, and Internet routing protocols.



APP CLIENTS

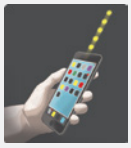
Most applications can run as servers in either physical or virtual environments, but they need a client interface to push or pull data to users. Currently, it is rare to find a useful application that is free-standing and unconnected from the Internet (such as Windows Notepad).

The most common web application client is the web browser. Web browsers have evolved far beyond Mosaic, released by the National Center for Supercomputing Applications in 1993. Nearly every web application now expects web clients to run active scripts like JavaScript or Flash.

Increasingly, applications themselves are also running on active scripting in the browser itself. In these cases, the client and browser communicate via HTTP, passing data and commands back and forth for processing. This creates new security consequences and therefore new requirements for testing and defense. Code parsers often add a new layer of security problems because their results can be hard to prioritize, and they can have trouble distinguishing code from user input.

App clients also include mobile apps, which are often preconfigured web browser interfaces to an existing web app. Apps and IoT devices can often call other apps to push, pull, or process data. This is usually done via APIs and application service connections.

APPLICATION THREATS AT EACH TIER



CLIENT

- Cross-site request forgery
- Cross-site scripting
- Man-in-the-browser
- Session hijacking
- Malware

APP SERVICES

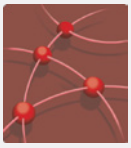
- API attacks
- Injection
- Malware
- DDoS
- Cross-site scripting
- Cross-site request forgery
- Man-in-the-middle
- Abuse of functionality

ACCESS CONTROL

- Credential theft
- Credential stuffing
- Session hijacking
- Brute force
- Phishing

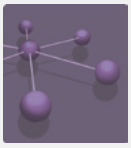
TRANSPORT LAYER SECURITY

- DDoS
- Key disclosure
- Protocol abuse
- Session hijacking
- Certificate spoofing



DOMAIN NAME SYSTEM

- Man-in-the-middle
- DNS cache poisoning
- DNS spoofing
- DNS hijacking
- Dictionary attacks
- DDoS



NETWORK

- DDoS
- Eavesdropping
- Protocol abuse
- Man-in-the-middle

FIGURE 5: APPLICATION THREATS

THREATS AT EACH TIER

Each tier of the application stack introduces unique vulnerabilities that attackers can target. To secure an app, it's important to understand the intricacies under the hood of each app tier, as shown in Figure 5. Note that the various types of threats shown at each tier are analyzed deeply in their respective sections of this report.

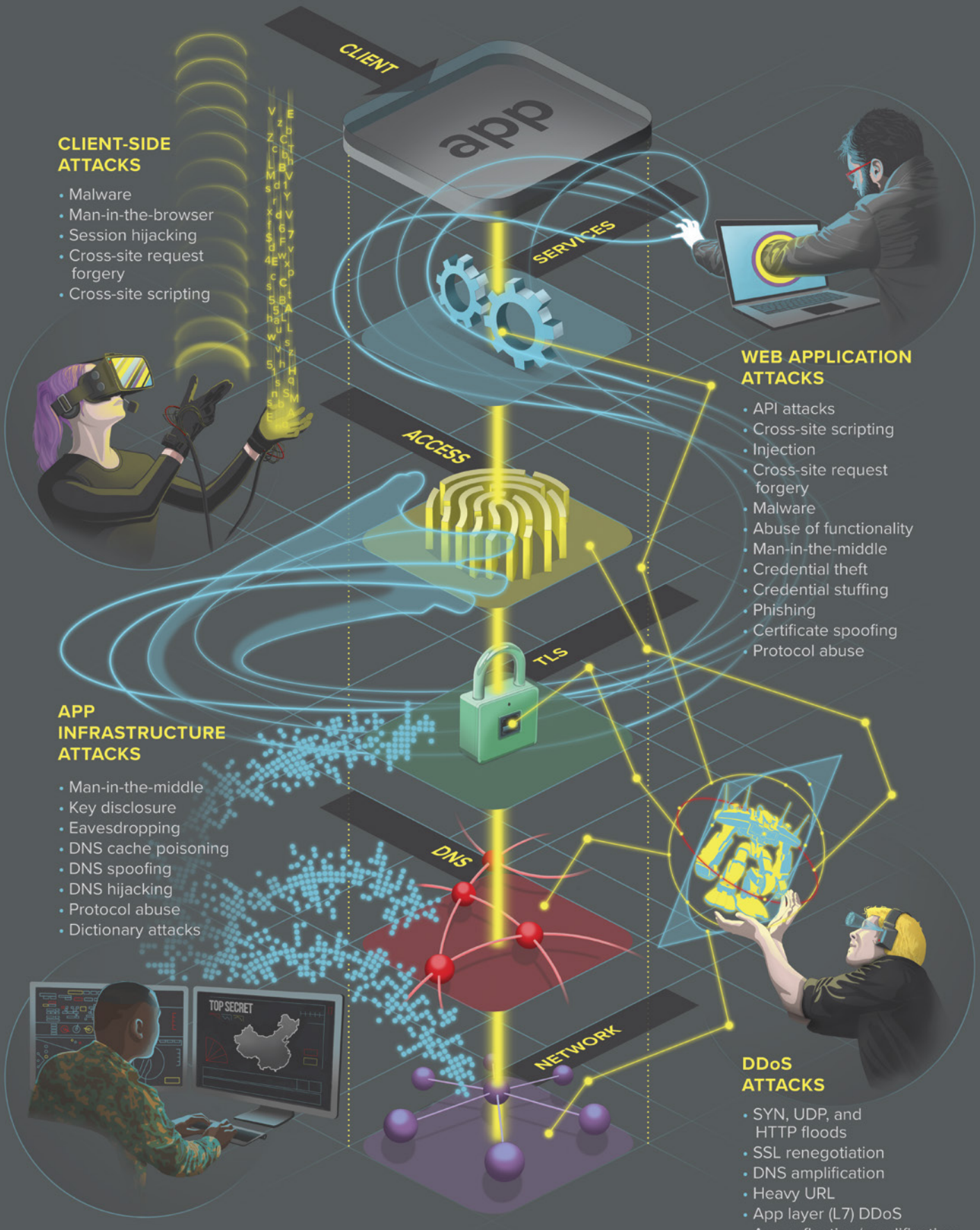
HOW ARE APPS ATTACKED?

IT'S A SIMPLE FACT: ANYTHING YOU PLACE ON THE INTERNET IS GOING TO BE ATTACKED—AND QUICKLY.

It's a simple fact: anything you place on the Internet is going to be attacked—and quickly. It costs attackers nearly nothing to poke at an application service and see what they get out of it. And they can do it around the clock. Whenever a new exploit technique is discovered, attackers rapidly cycle through the entire Internet to see which servers are vulnerable.

With thousands of attacks (many of them new) striking the front doors of our applications every day, the way we classify those attacks can immensely improve our defensive capability. Attacks can and do happen across all tiers of an application—sometimes to several tiers simultaneously. When we look at application attacks, we categorize them as follows: web application attacks, app infrastructure attacks, denial-of-service attacks, and attacks against the client (figure 6).

FIGURE 6: SPECIFIC THREATS MAPPED TO APPLICATION ATTACK CATEGORIES



WEB APPLICATION ATTACKS

Because of the complexity and profusion of application services, it is inevitable that there will be lots of risk found here. Analysis by F5 Labs of 433 breach cases spanning 12 years shows that 86% of breaches started with an attack that targeted the application itself or a user with credentials for the app.

In the application stack, two main areas are hit. The first is the services tier, which involves attacks against the following sub-tiers: internal code, external code, and server-side Infrastructure. Attacks here include things like buffer overflows in web servers and exploits in web services such as Apache Struts or unique vulnerabilities found in custom code, like business logic abuse. F5 Labs analysis found that the services tier was the initial target of attack in 53% of breaches, making it the primary attack vector.

Web application attacks also include access control attacks, such as credential stuffing, brute force attacks by botnets, man-in-the-middle attacks, and credential theft as a result of phishing. F5 Labs breach analysis found that attacks against application access were the second most common initial vector, targeted in 33% of breach cases.

APPLICATION INFRASTRUCTURE ATTACKS

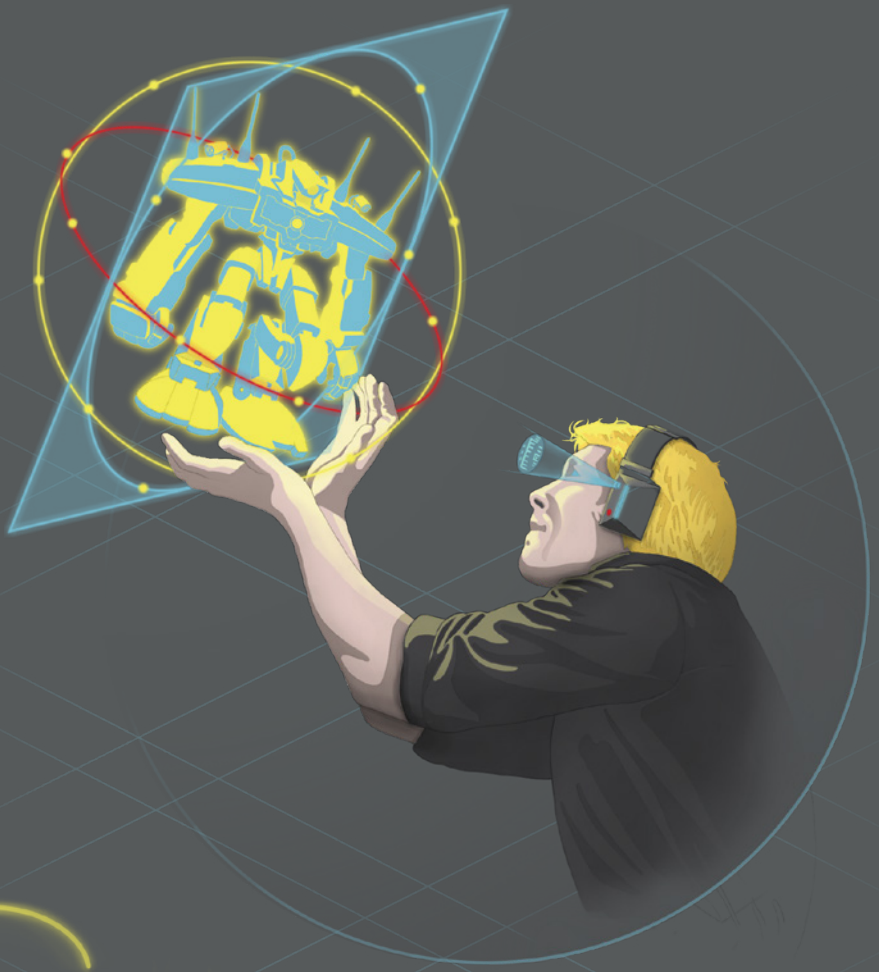
Application infrastructure refers to the systems that applications depend on yet are external to the application itself. Attacks against application infrastructure include those that target the Transport Layer Security (TLS) tier, Domain Name System (DNS) tier, and the network tier.

86% OF BREACHES STARTED WITH APPLICATION OR IDENTITY ATTACKS.

ATTACKS AGAINST APPLICATION ACCESS WERE THE SECOND MOST COMMON INITIAL VECTOR, TARGETED IN 33% OF BREACH CASES.

DENIAL-OF-SERVICE ATTACKS

Attackers can strike at every tier and visible component of an application, which means denial-of-service attacks are an ever-present threat. Since an application is dependent on all layers functioning properly, you should consider DoS attack protection a vital part of your application security strategy. Most denial-of-service attacks are distributed (DDoS), meaning they originate from an army of hacker-controlled bots. However, there are some denial-of-service attacks that can be done with a single packet, such as the TKEY query handling flaw in BIND DNS.[4]



CLIENT ATTACKS

Application clients can be exploited as well, often via malware or physical attacks. This report's discussion of client threats focuses on the app client's relationship to the application. For example, malware may infect an app client, but this report focuses solely on that malware's effect upon app security, not necessarily the other damage it inflicts upon the user's security outside of app usage.


03

WHAT HAPPENS TO AN ORGANIZATION WHEN APPS ARE ATTACKED?

When we talk about application risk, what do we really mean? Risk is only meaningful in the context of the likelihood of a threat occurring and the resulting impact it would have on what the organization cares about. In this sense, impact measures the bad things we don't want to happen to us.

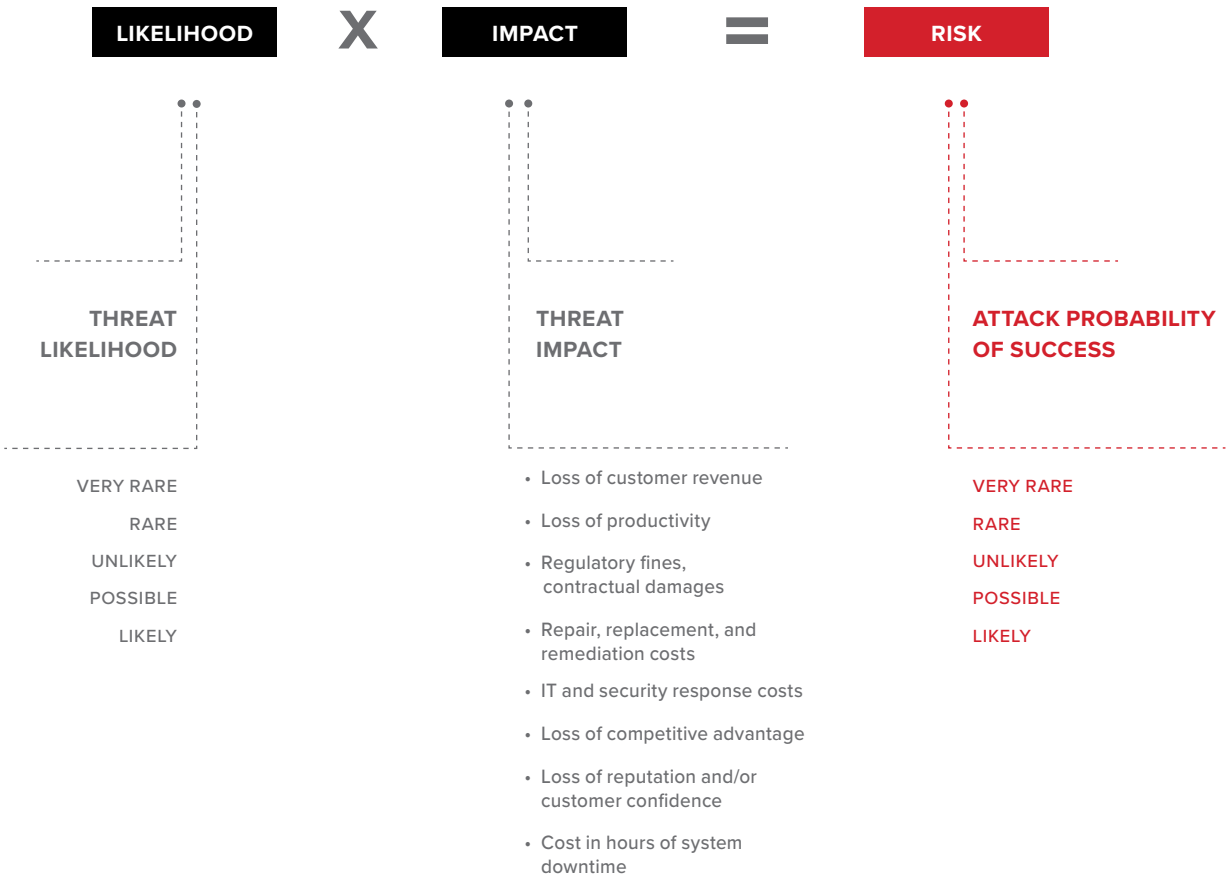
CISOs STATE THAT PREVENTING APPLICATION DOWNTIME IS THE #1 MISSION FOR THEIR ORGANIZATION.

For the most part, impact is calculated in estimated dollars lost and ultimately becomes the organization's threshold for what it considers an acceptable loss. One hour of downtime per month on a major e-commerce site may be tolerable while two hours may not be.

In the F5 and Ponemon report, *The Evolving Role of CISOs and their Importance to the Business*,[5] CISOs said that preventing downtime (or, stated positively, ensuring availability) was the most important mission for the organization. That's probably no coincidence since availability is one pillar of what's known as the security CIA triad: confidentiality, integrity, and availability.

When it comes to the data stored and processed by applications, protecting the integrity of applications and data may become even more important in the future, especially in the global financial system. Recently, the Carnegie Endowment for International Peace urged countries to refrain from conducting cyber-attacks that threaten to "undermine the integrity of data and algorithms of financial institutions in peacetime and wartime."[6]

FIGURE 7: CALCULATION OF IMPACT AND RISK



ANALYZING APPLICATIONS FOR RISK

The correct way to begin a risk assessment is with a complete inventory and consideration of the critical applications in use.[7] Counting, analyzing, and tracking your applications will tell you what you need to protect and where.

Counting and tracking applications

As critical as applications are, we are still struggling to keep track of them. There's no question it's far more difficult these days with users being able to download apps to their devices, both company-owned and personal. Perhaps even more threatening is "Shadow IT"—the use by many employees of web-based and mobile apps without IT's knowledge or permission. This can pose a significant security risk for organizations as these apps are often used to conduct business, so they often contain or are used to share confidential company information. There's also the problem of employees using their personally owned (unmonitored, and often compromised) devices for job-related activities.

Data from the 2018 F5 and Ponemon security survey indicates that the majority of organizations have little confidence in their ability to keep track of all applications. Only 24% of respondents were somewhat confident, and a disturbing 38% had no confidence at all.

38% OF ORGANIZATIONS SURVEYED HAD NO CONFIDENCE AT ALL IN THEIR ABILITY TO KEEP TRACK OF ALL THEIR APPLICATIONS.

Where do applications reside and how important are they?

Applications are moving out of the organization and becoming increasingly dispersed, whether they're hosted in the cloud, as mobile apps, or as SaaS solutions. Of the apps commonly used by most organizations, 52% or fewer are hosted on premises.

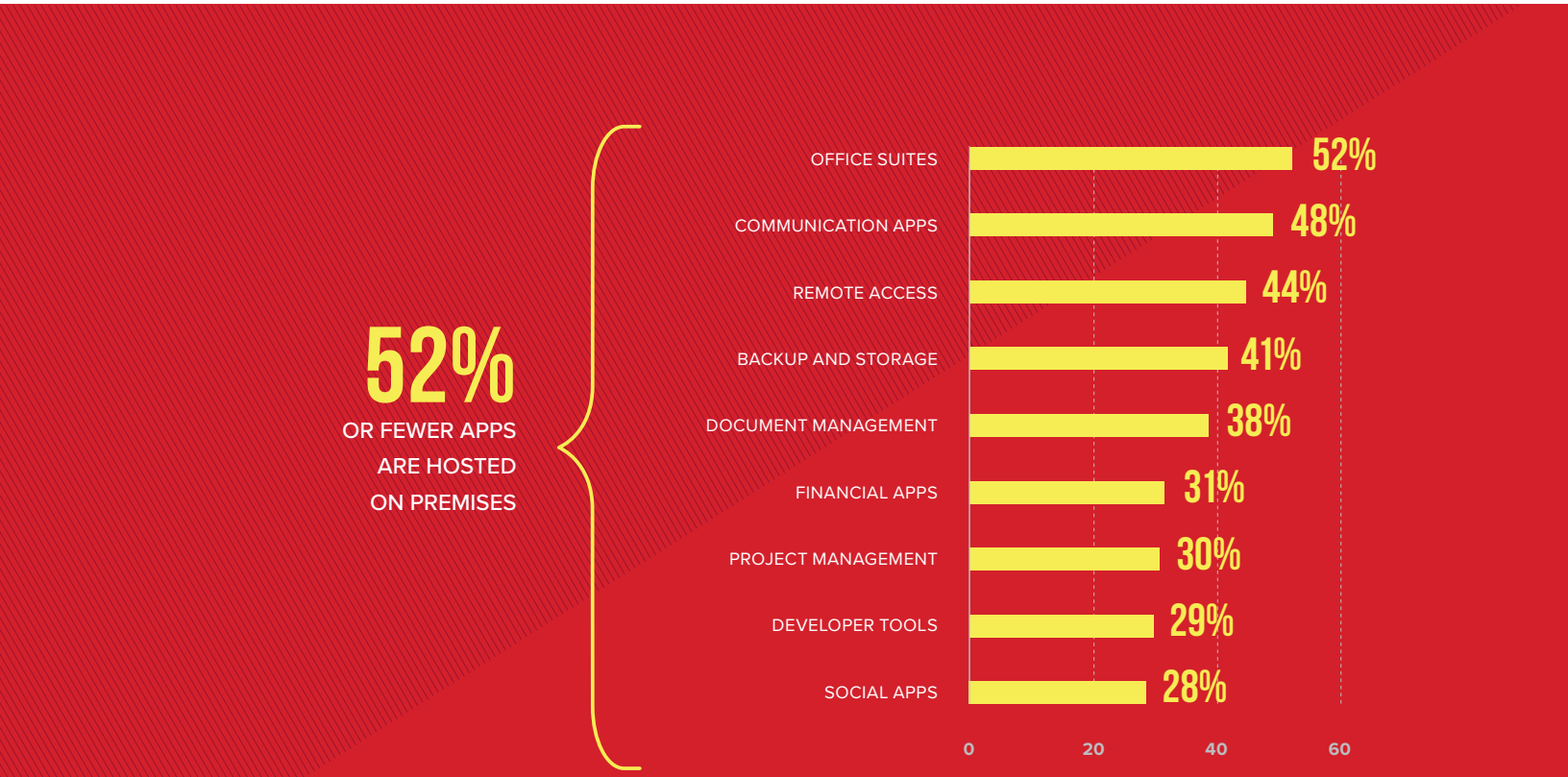
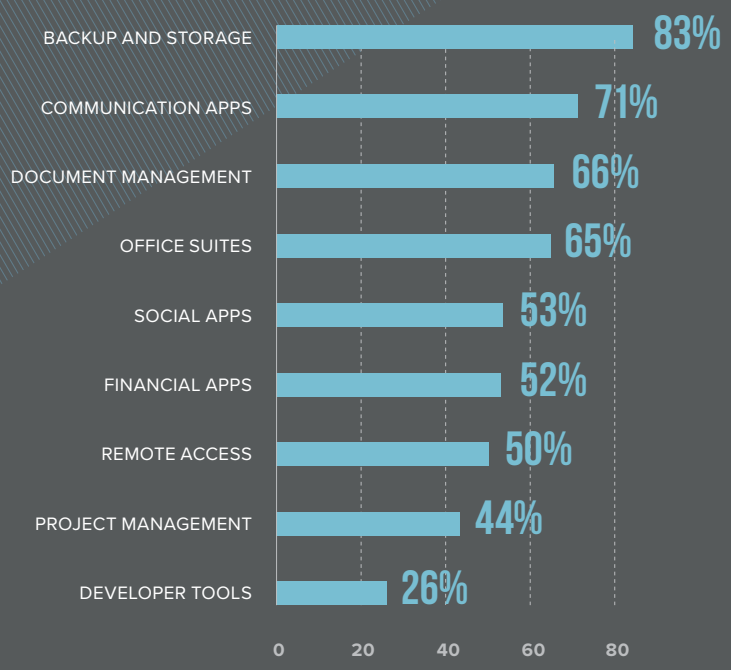


FIGURE 8: PERCENTAGE AND TYPES OF APPS THAT ARE STILL HOSTED ON PREMISES

FIGURE 9: TYPES OF APPLICATIONS MOST COMMONLY USED IN ORGANIZATIONS



83%

IT'S NOT SURPRISING TO SEE BACKUP AND STORAGE AT THE TOP OF THE USAGE LIST SINCE MANY DEVICES TODAY PROVIDE LIMITED LOCAL STORAGE.

What kind of apps are we talking about? Backup and storage are some of the most commonly used web applications. It's not surprising to see this category at the top of the usage list (83%) since many devices today provide limited local storage and vendors continue to push consumers to use their cloud-based storage solutions.

Other commonly used web-based apps include communication applications like email (71%), document management and collaboration (66%), and apps in the Microsoft Office suite (65%). To a lesser degree but still above 50%, organizations cited the use of web-based social, financial, and remote access apps (see figure 9).

FIGURE 10: MOST IMPORTANT WEB APPLICATIONS

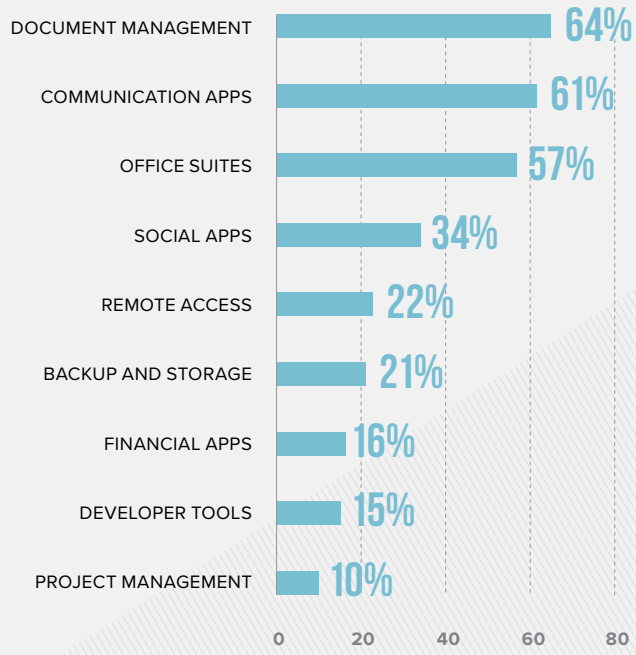
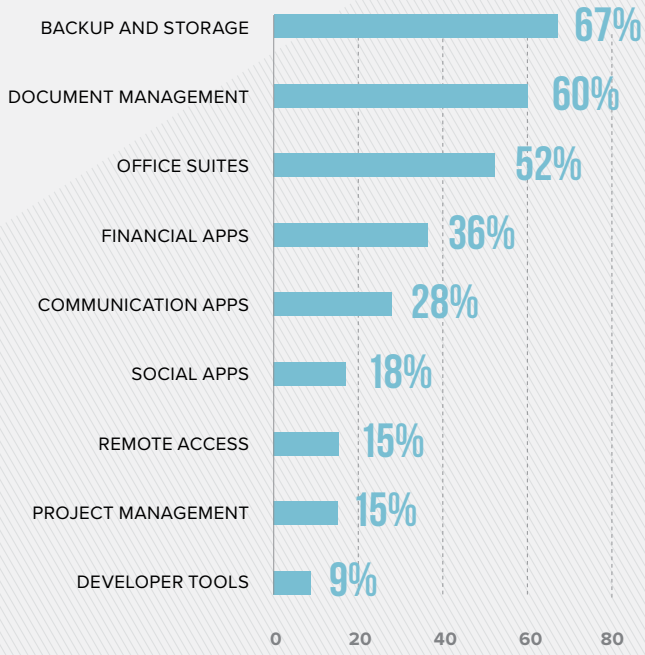


FIGURE 11: WEB APPLICATIONS THAT STORE THE MOST CRITICAL DATA



28% OF SURVEY RESPONDENTS SAID BETWEEN 1/4 AND 1/2 OF THEIR APPLICATIONS WERE MISSION CRITICAL.

When we compare the apps most used to those that organizations consider most important to their mission, there is some expected overlap, for example, with web-based document management and collaboration, communication apps, and the Office suite. These are the tools people rely on day to day to get their jobs done.

Interestingly, web-based backup and storage apps fall to 21% in terms of importance to the organization's mission (figure 10), yet they take the top spot at 67% (figure 11) when we asked which web-based apps store the organization's most critical data. Document management and collaboration apps and Office apps take second and third place here. In terms of percentages, over one-fourth (29%) of respondents said 11-25% of their apps were mission critical; about the same percentage (28%) of respondents said 25-50% of their apps were mission critical.

FIGURE 12: PAIN LEVEL OF AN ATTACK THAT RESULTS IN THE LEAKAGE OF CONFIDENTIAL OR SENSITIVE INFORMATION

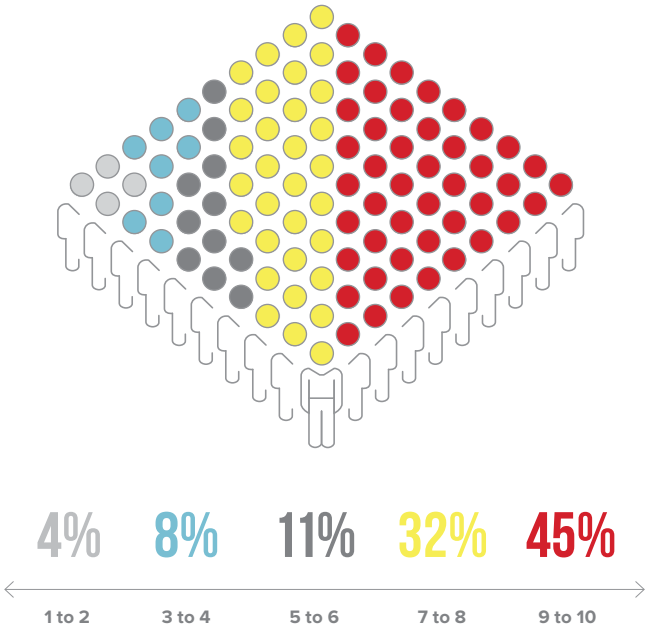
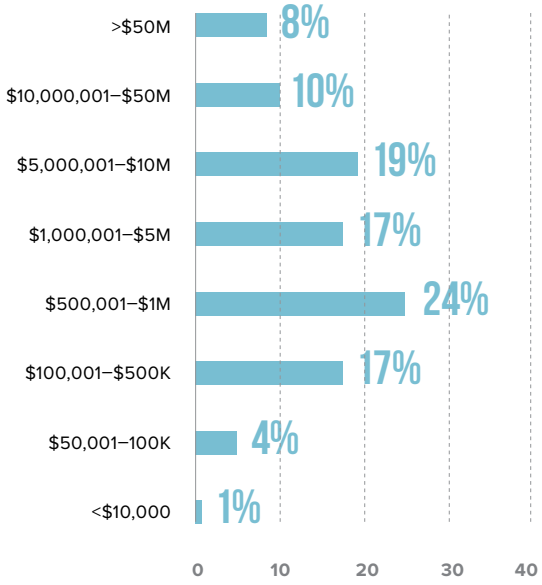


FIGURE 13: THE COST OF A BREACH OF CONFIDENTIAL OR SENSITIVE INFORMATION



HOW MUCH DOES AN ATTACK HURT? WHAT ARE THE IMPACTS?

Certainly, attacks of any kind can have a significant impact on the organization, but singular events like a confidential data breach, a web application being altered, or the loss of availability of a critical application can have significant impact on their own as well. Each organization measures impact differently depending on their type of business, industry, business model, and how they answer questions like these:

- How long can systems be down without affecting revenue, compliance, or contractual commitments?
- How would an attack impact the ability to provide service?
- How long before an outage would affect employee productivity?
- What would be the impact of an irretrievable loss of data?
- How could the loss of one system impact others?

Loss of sensitive or confidential data

On a scale of 1 to 10, with 10 being the most painful, 77% of survey respondents agree that a breach of confidential or sensitive information (such as intellectual property or trade secrets) rated from 7 to 10 in terms of impact to the organization (see figure 12). Only 4% of respondents said it would have little to no impact.

When we look at the impact in actual dollars (on a scale ranging from less than $10,000 to over $50 million), more than three-quarters (78%) of respondents said a breach of this sort would cost their companies over $500,000 (see figure 13). While 22% of respondents said the cost to their companies would be $500,000 or less, 8% pegged the cost at over $50 million.

FIGURE 14: PAIN LEVEL OF AN ATTACK THAT LEAKS PERSONALLY IDENTIFIABLE INFORMATION (PII)

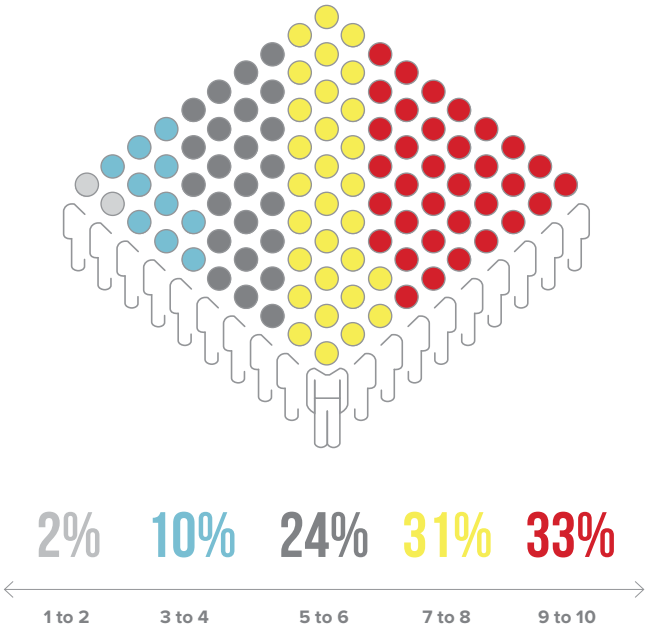
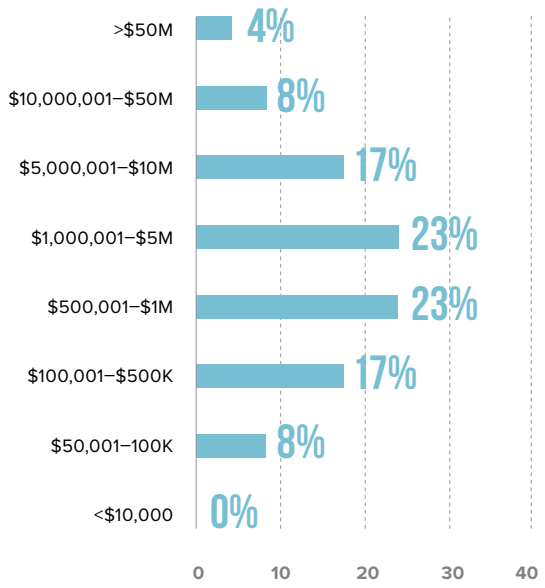


FIGURE 15: ESTIMATED COST OF AN ATTACK THAT LEAKS PERSONALLY IDENTIFIABLE INFORMATION (PII)



Loss of personally identifiable information (PII)

The loss of PII can be particularly devastating to a company's reputation as it signals their inability to protect the privacy of their customers, consumers, and employees. Again, on a scale of 1 to 10 (with 10 being the highest), 64% of survey respondents rated the impact of a PII breach very high at 7 to 10, 24% rated the impact moderate at 5 to 6, and only 12% at low to none (figure 14).

Again, when evaluating the cost of a breach resulting in the loss of PII data, the survey responses were very similar to those found with the cost of the breach of sensitive or confidential information. There were 6% more responses in the $1 million to $5 million range at 23% (versus 17%) (see figure 15).

Tampering with the application itself

Breaches that result in data loss get a lot of attention in the press because they often directly affect millions of unsuspecting consumers. Attacks that tamper with an application itself (for an attacker's own nefarious purposes) get far less public attention, although organizations that are victims of such attacks are required to report them under the Sarbanes-Oxley Act (SOX). Application tampering includes things like website defacement, which changes the appearance of the application or perhaps the contents that it displays. Tampering can also include modifications that alter the way in which an application functions, such as slowing its performance; displaying wrong, altered, or unexpected pages; or redirecting users to a completely different website. Organizations discovering slow-performing applications should be concerned about the possibility that they're being used to mine cryptocurrency. Organized cyber-criminals are quick to shift their focus to the most lucrative form of hacking and are turning to crypto-mining to make a pretty penny. All of these changes compromise the integrity of the application.

FIGURE 16: PAIN LEVEL OF AN ATTACK THAT COMPROMISES THE INTEGRITY OF AN APP (APP TAMPERING)

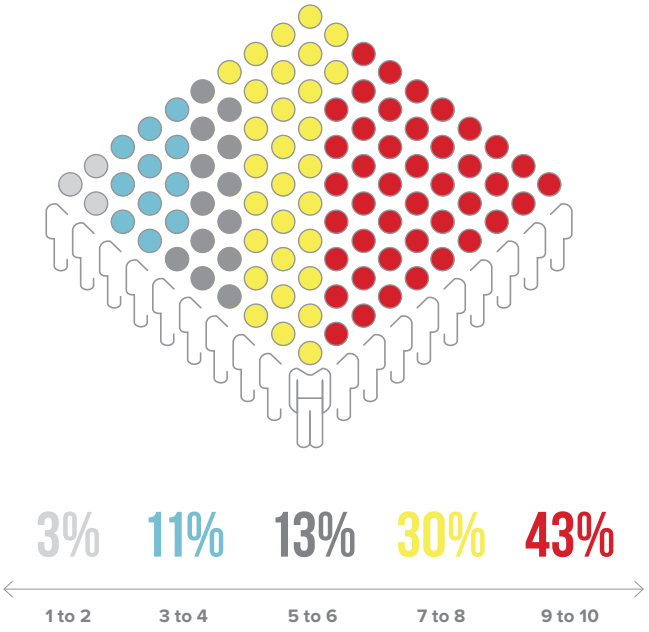
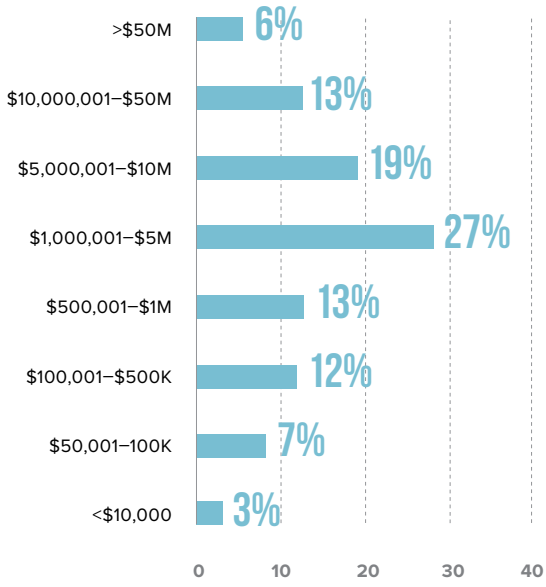


FIGURE 17: ESTIMATED COST OF AN ATTACK THAT COMPROMISES THE INTEGRITY OF AN APP (APP TAMPERING)



Imagine the impact to an e-commerce site, for example, if a user chose to purchase a specific product but the application was altered to fulfill the order with a different company's product. Or imagine the money transfer function of an online banking app being altered in such a way that it withdraws funds from a user's account as expected but deposits those funds in the attacker's bank account rather than the intended recipient's account.

Most organizations consider these types of attacks nearly as devastating as data breaches because they can disrupt business operations, lead to significant errors or lost business, and, to compound the issue, can go unnoticed for a period of time. When survey respondents were asked to rate the impact of this type of attack to their organizations, 73% rated it 7 or higher on a scale of 1 to 10. Only 3% of respondents said it would have little to no impact (see figure 16).

Tampering with an application can be just as costly to an organization as a data breach. In fact, more than three-quarters of respondents (78%) put the total cost at over $500,000. More specifically, 40% estimated the cost would fall in the $500,001 to $5 million range; 32% put the cost somewhere between $5 million and $50 million. Fewer than one-fifth (22%) estimated the cost at below $500,000, while 6% estimated costs exceeding $50 million (see figure 17).

Impacts related to denial-of-service attacks

We've already briefly mentioned the CIA triad—confidentiality, integrity, and availability. While many security practitioners focus on the confidentiality and integrity of data, less focus is put on ensuring that the correct data is available to the right people at the right time. Denial-of-service (DoS) attacks, which directly affect the availability of an application or website, can significantly impact an organization's ability to do business with its customers or conduct business internally.

FIGURE 18: PAIN LEVEL OF A DOS ATTACK THAT PREVENTS USERS FROM ACCESSING AN APP OR DATA

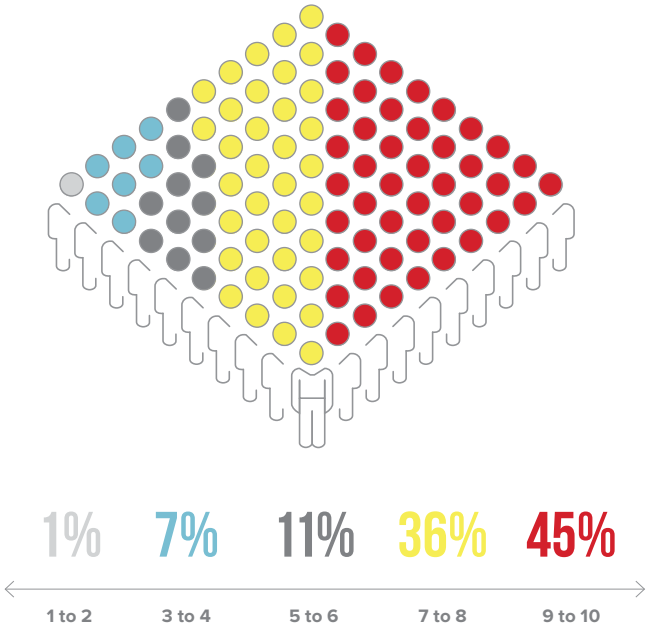
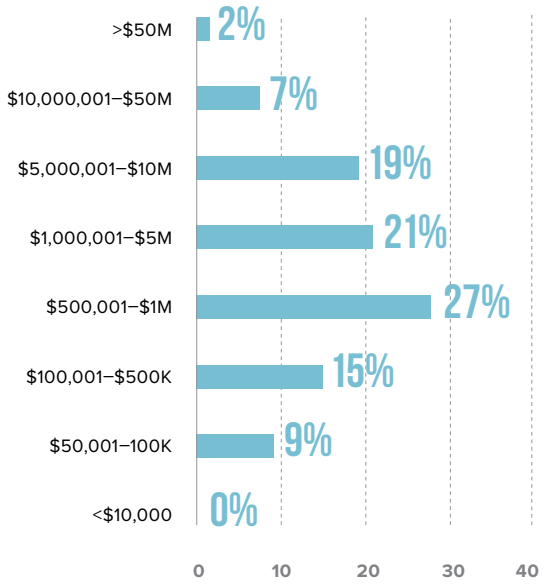


FIGURE 19: ESTIMATED COST OF A DOS ATTACK THAT PREVENTS USERS FROM ACCESSING APPLICATIONS OR DATA



These types of attacks ranked very high on the impact scale with survey respondents. As many as 81% rated loss of availability at 7 to 10 on a scale of 1 to 10 (see figure 18). Only 8% rated this type of attack as having low to no impact. In those cases, the type of application or business was likely a factor. An application or website with mostly static content can weather a DoS attack far better than an ecommerce, banking, or stock-trading site, for example, where availability is crucial to the business.

81% OF RESPONDENTS RATED THE LOSS OF AVAILABILITY AT 7 TO 10 IN IMPACT ON A SCALE OF 1 TO 10.

Nearly three-quarters of respondents (74%) rated the cost of this type of attack in the $.5 million to $50 million range, with the highest percentage (27%) falling right in the middle at $500,000 to $1 million (see figure 19). Twenty-four percent put the cost at $500,000 or less; only 2% put it at over $50 million.

It's clear from our survey results that the majority of organizations—on average, about three-quarters of all respondents—are acutely aware that any kind of a data loss, whether confidential or PII—as well as the loss of availability of critical applications—would have a dramatic effect on their organizations, both in terms of reputation and dollars lost.

Perhaps the greatest challenge, then, comes back to organizations being able to track all of the apps in use by their employees. Clearly, when 62% of survey respondents say they are not confident or are only somewhat confident in their ability to track applications and the data used by them, there's a need for improvement in this area. Once organizations have a handle on the applications they're using, they're in a better position to decide how best to protect them from various types of attacks, which we take a closer look at in the next section.


Your email has been sent!
1111 2222 3333 4444
5555
SSN
Password
Username
Username
Password
1111 2222 3333 4444
5555

WEB APPLICATION ATTACKS

Web applications are formed from the interaction between the constituent tiers and sub-tiers we discussed in the *What is an App?* section. Each of these tiers operates at scales, locations, and orders of complexity. This complexity means that when these sub-tiers interact with the harsh environment of the Internet, negative consequences amplify. A perfect example of this is the Jakarta Multipart parser vulnerability in Apache Struts, discovered in March 2017,[8] a minor software bug in an external code sub-tier that came out of nowhere to puncture our concept of what is secure.

30% OUT OF 304 ANALYZED, WEB ATTACKS WERE THE TOP BREACH BY FAR, AT 30%.

TOP BREACHES INVOLVING APPLICATION SERVICES

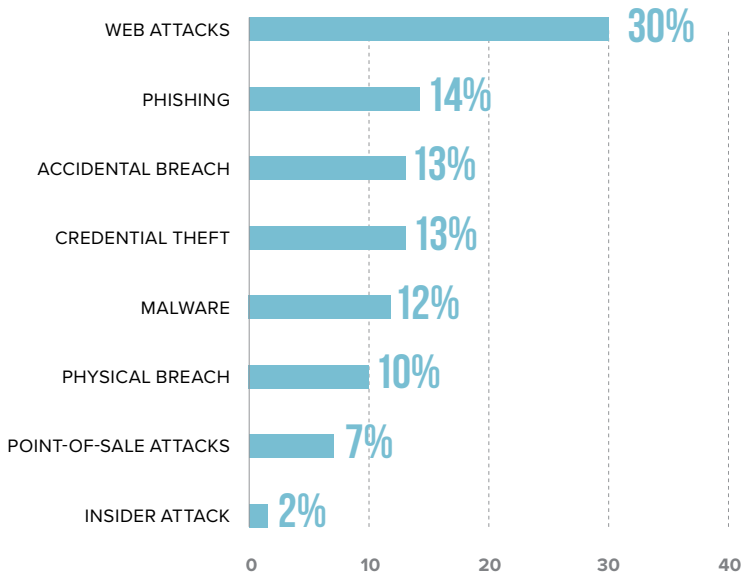
The number of applications that get attacked can be derived from public breach records. When we use the term "breach," we're referring to any kind of a penetration of networks, systems, or applications, whether data was stolen or not. Note that not all attacks (such as DDoS) are breaches.

Although most U.S. states require that victims be notified of data breaches, only a few states' attorneys general collect and share breach letters on their websites. California, Washington, Idaho, and Oregon represent over 16% of the U.S. population—a decent sample size to provide insight into what's going wrong.

As part of a research project with F5 Labs, the Whatcom Community College Cybersecurity Center faculty read and reviewed every single breach letter in these states for all of 2017 and Q1 of 2018. Of the 384 reported breaches in 2017 and Q1 2018, 304 (79%) provided sufficient explanation for the cause of breach for analysis. The remaining 21% did not report the cause of the breach. (Note that some of these breaches may have occurred prior to 2017 but were only reported recently.)

We analyzed the 304 breaches and broke them down into the following categories: web attacks, phishing, credential attacks, point-of-sale attacks, physical and accidental breaches, malware (as categorized in the breach letters), and insider breaches. Web attacks, by far, came in at the top of the list at 30%, followed by phishing (14%), credential hacks (13%), and accidental breaches (13%) (see Figure 20).

FIGURE 20: APPLICATION BREACHES BY INITIAL ATTACK TYPE (WA, OR, ID, CA 2017)

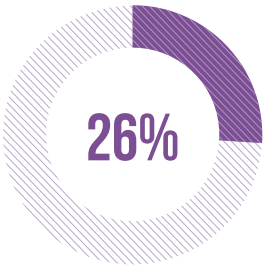


APPLICATION ATTACKS

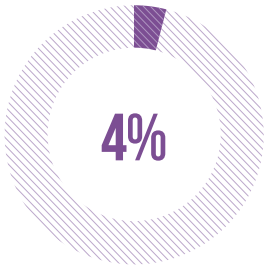
Application attacks were the leading known breach cause in 57 breaches (30%). This is the data we are interested in because application breaches represent the largest threat. The breakdown of the specific application breaches included credit card stealing via web injection (70%), website hacking (26%) and database hacking (4%).

FIGURE 21: BREACHES BY ROOT CAUSE (WA, OR, ID, CA 2017)

70% OF WEB BREACHES ARE FROM INJECTIONS



26% OF WEB BREACHES ARE FROM WEBSITE HACKING



4% OF WEB BREACHES ARE FROM DATABASE HACKING

EXPLOITS INVOLVING WEB APPLICATION SERVICES

Before you can begin to address application security, you need to first ensure you have covered the minimum threat capability. For anyone on the Internet, this minimum threat is "script-kiddies" who don't have the technical expertise to write their own scripts but can break into your applications with point-and-click exploits readily available on the Internet. To understand their capability, one needs to keep an eye on what scriptable, easy-to-use exploit kits are available. To this end, F5 analyzed all the web application exploits available at Exploit-DB.[9] Since the scripts available at this website are free to any and all, they represent this bare minimum threat capability and give us a baseline of what the most common threat is.

Examining the corpus of Exploit-DB records relating to web attacks, we found that 69% of the exploits are against PHP, a widely-used web development scripting language. Hardware-based device (IoT) exploits came in second at 10%. We looked deeper into what the PHP exploits were doing and categorized them as shown in Figure 22.

At 46%, SQL injection via PHP is by far the most prevalent exploit script available, which contributes significantly to the risk of SQL injection attacks on PHP-driven web apps.

We examined all the other non-PHP exploits and categorized them as shown in Figure 23. Among non-PHP exploits, cross-site scripting (XSS) at 13.3%, and cross-site request forgery (CSRF) at 9.6% come to the forefront, followed closely by authentication bypass at 9.1%. In the hands of a knowledgeable attacker, all three of these exploits can lead to an adversary gaining unauthorized access to a web application by impersonating a user. So, for non-PHP websites, protecting access is a high priority.

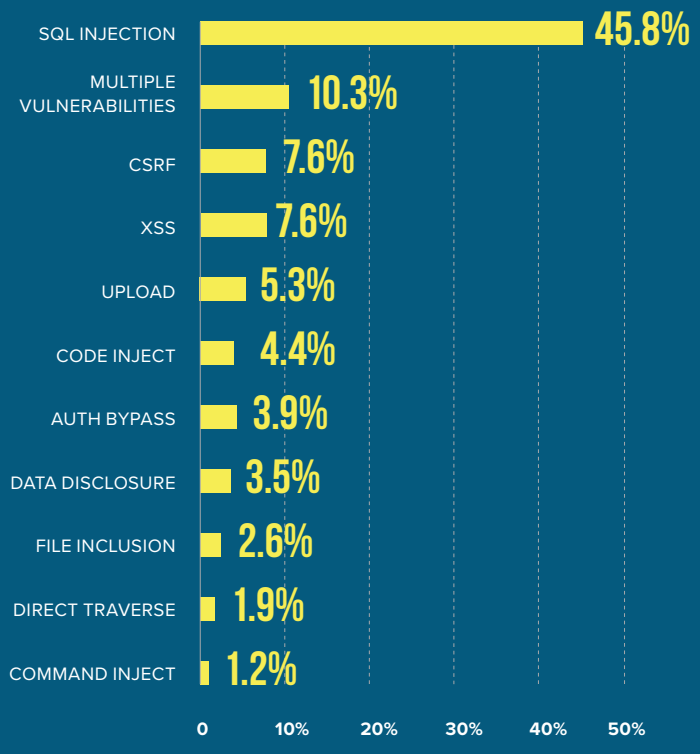


FIGURE 22: EXPLOIT-DB PHP EXPLOIT CATEGORIES

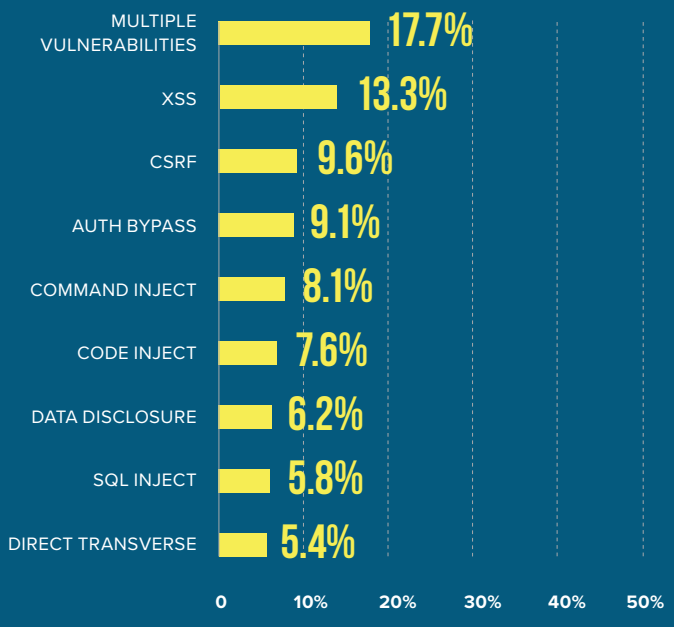
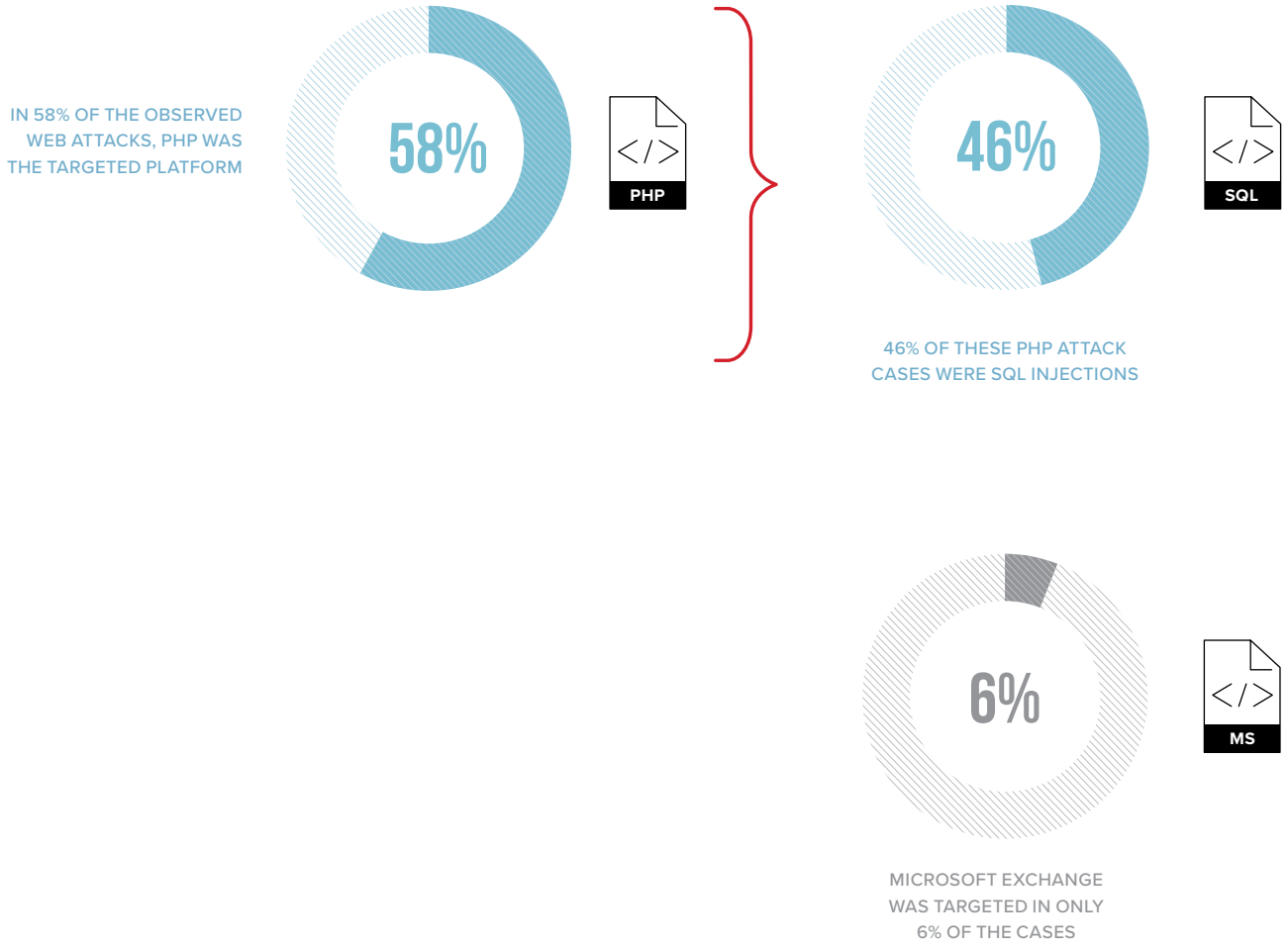


FIGURE 23: EXPLOIT-DB NON-PHP EXPLOITS BY CATEGORY

FIGURE 24: TOP 3 INTRUSION ATTACK TARGETS

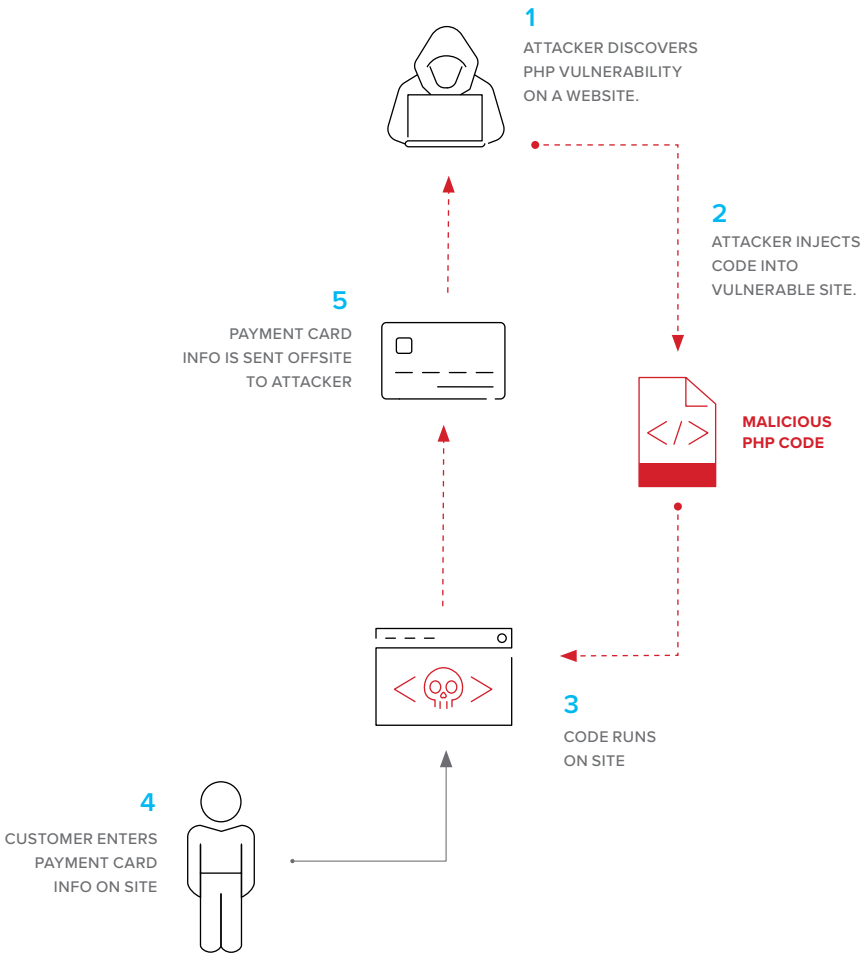


TOP ATTACKS INVOLVING APPLICATION SERVICES

Working with our data partner Loryka, we also looked at global intrusion and honeypot data collected from web attacks on 21,010 unique networks over 2017.

In 58% of the cases, PHP was the targeted platform and 46% of those attacks were SQL injections. Microsoft Exchange was targeted in only 6% of cases. Across all targeted platforms, 34% of attacks were SQL injections.

FIGURE 25: COMMON INJECTION ATTACK PATH



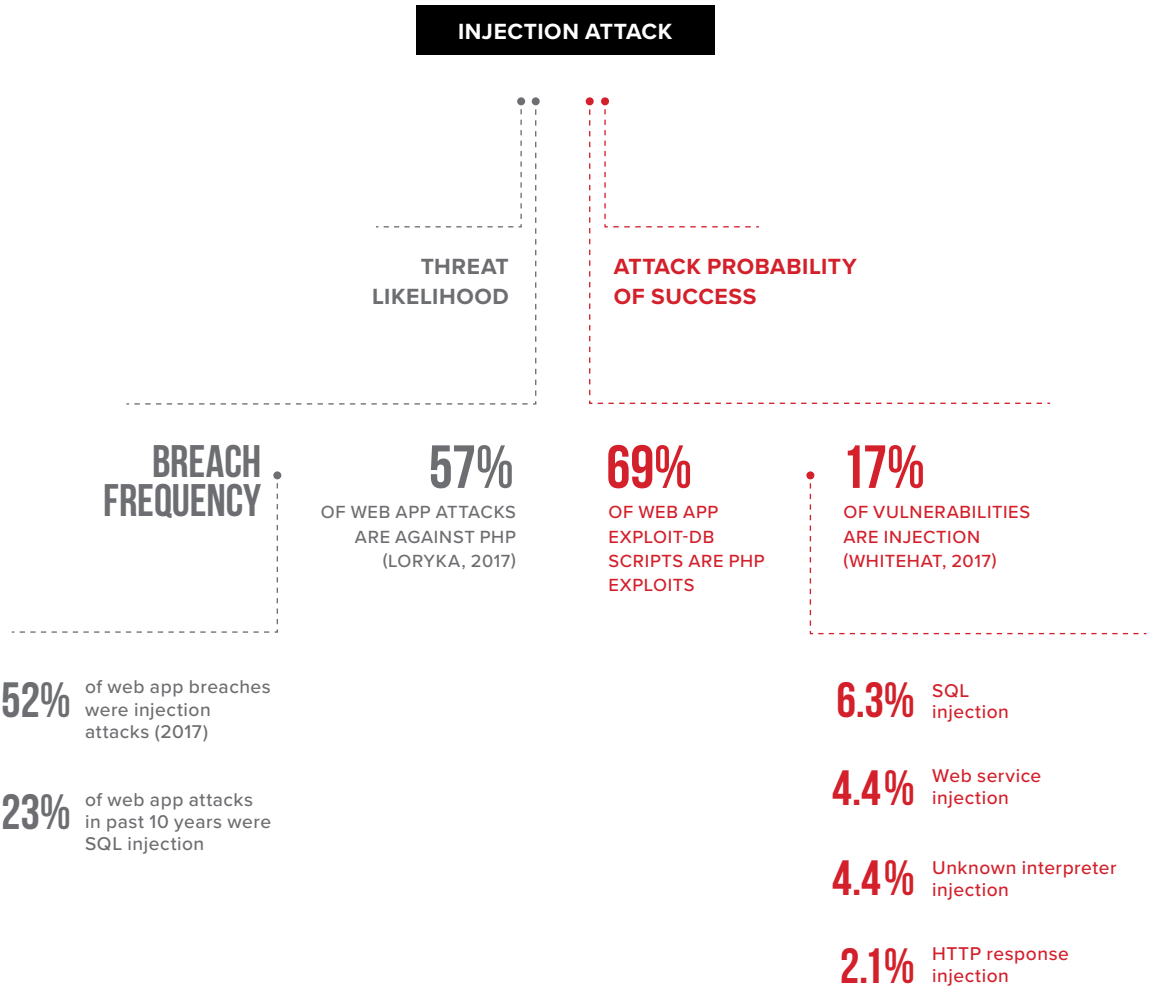
PHP IS A TEMPTING TARGET FOR HACKERS BECAUSE IT'S VERY POPULAR. FIND A HOLE IN PHP AND YOU CAN COMPROMISE MILLIONS OF SITES.

INJECTION ATTACKS

Injection attacks were the most significant web application breaches in 2017. This was likely due to malware campaigns by cybercriminals involving a specific set of well-known shopping cart vulnerabilities that have been used for injection attacks in the past few years.[10]

Cybercriminals start by finding a vulnerability in an e-commerce website that allows them to change the website source code. The attacker injects code into the site to silently copy a customer's payment card information. Figure 25 illustrates a web application version of point-of-sale card stealing malware.[11] Attacks can get sophisticated with the malware being served from another site, the data being stored at another remote site, and the entire conversation encrypted with a valid HTTPS certificate.

FIGURE 26: SUMMARY OF INJECTION ATTACK LIKELIHOOD AND PROBABILITY OF SUCCESS



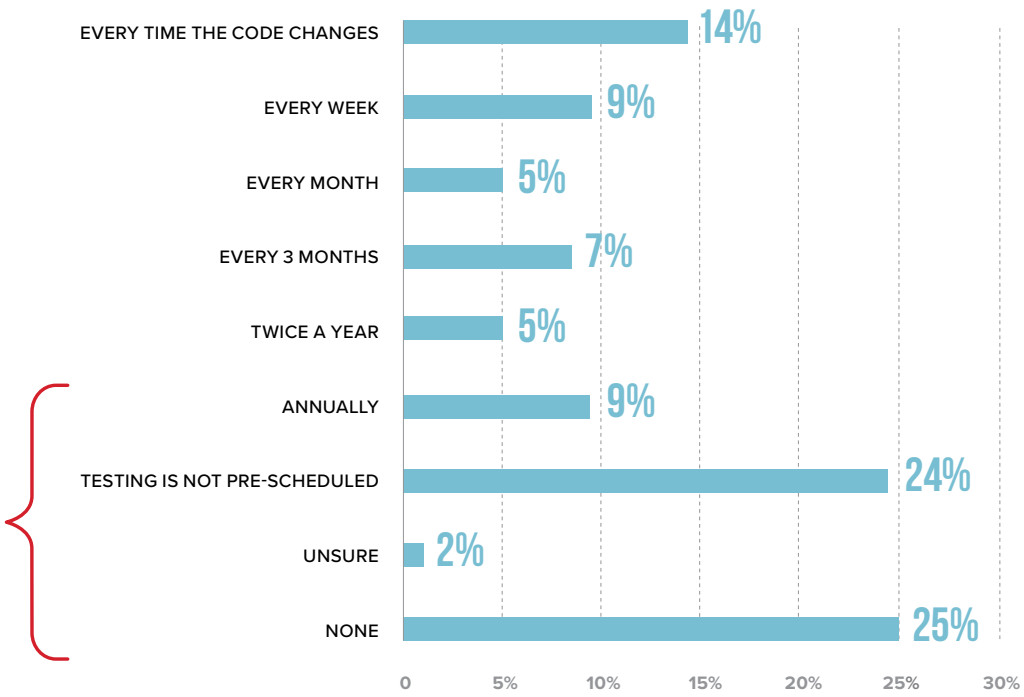
The most famous list of web application security risks is the OWASP Top 10,[12] which compiles and ranks the ways applications can be compromised. In 2017, the Top 10 list was the most data-driven version yet, reviewing industry experts and application security firms, as well as surveying members on the exploitability, prevalence, and detectability of web application risks. It's no surprise that, once again, injection attacks came in as the highest priority risk.

IT'S NO SURPRISE THAT, ONCE AGAIN, INJECTION ATTACKS CAME IN AS THE HIGHEST PRIORITY RISK ON THE 2017 OWASP TOP 10 LIST.

There are two sets of victims in these cases: the first are the companies running e-commerce sites that have code that collects payment card information from their customers, who are the second set of victims. Easily compromised shopping cart code in your application would be classified as an OWASP risk *A9—Using Components with Known Vulnerabilities*.[13] These kinds of attacks show the importance of using secure components in your web application, testing for vulnerabilities, and watching the integrity of your site code.

A majority of injection flaws are in external libraries, which means that there are usually patches available for these flaws. However, finding and patching these holes is not always completed. WhiteHat Security noted that 8.2% of discovered vulnerabilities in 2017 were for code libraries that remained "unpatched."

FIGURE 27: HOW OFTEN ORGANIZATIONS TEST FOR WEB APPLICATION VULNERABILITIES



60%

OF RESPONDENTS HAVE GROSSLY INDADEQUATE SCENARIOS FOR MODERN WEB APP SECURITY

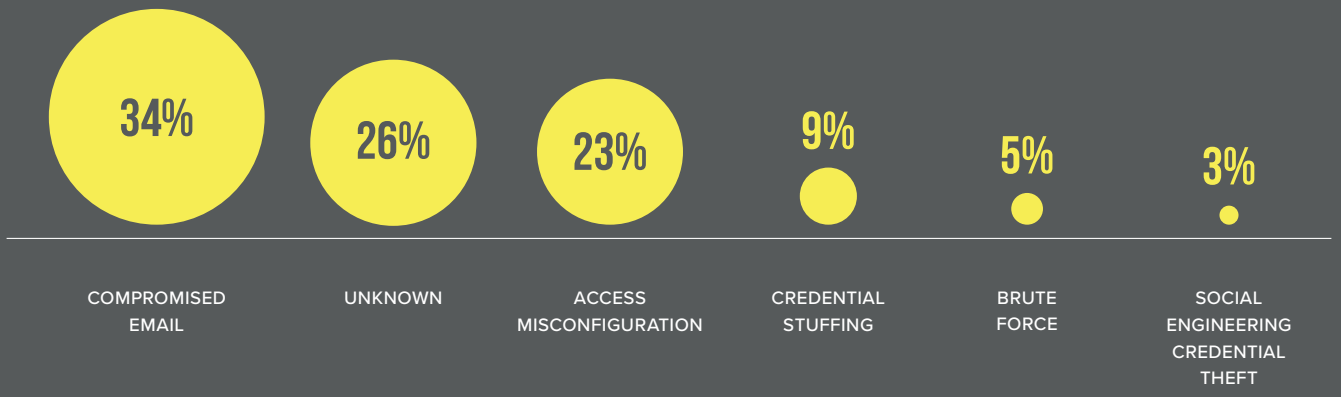
THE APACHE STRUTS ATTACK ON EQUIFAX, ONE OF THE WORST ATTACKS IN 2017, WAS A SERVER-SIDE TEMPLATE INJECTION VULNERABILITY.

The F5 Ponemon security survey asked, "How often does your organization test web applications for threats and vulnerabilities?" and the results were unsettling. Sixty percent of respondents said they don't test for web application vulnerabilities at all, there is no pre-set schedule, they are unsure if it happens and at what frequency, or they only test annually. All four of these scenarios are grossly inadequate for modern web application security.

Referring back to Figure 26, why is PHP such a focus for web site hackers? For one, it's very popular. It creates a tempting target for attackers—find a hole in PHP and you can compromise millions of sites. More significantly, perhaps, is its popularity among novice programmers whose lack of app security training results in a huge number of vulnerable websites. PHP looks for files with the php extension in its directory structure, making it very easy to inject new code by uploading new PHP files. PHP programmers are advised to review the OWASP PHP Security Cheat Sheet[14] and patch their systems regularly.

The Apache Struts attack on Equifax, one of the worst attacks in 2017, was a server-side template injection vulnerability.[15] Apache Struts is an open-source framework used to generate dynamic web pages from templates based on data inputs, including user-supplied data. This particular vulnerability enabled an attacker to inject commands into the web application and take over the system. That's why user input sanitization to ensure only appropriate input is allowed into an application is a crucial control to prevent injection attacks.

FIGURE 28: HOW UNAUTHORIZED ACCESS IS OBTAINED



ACCOUNT ACCESS HIJACKING

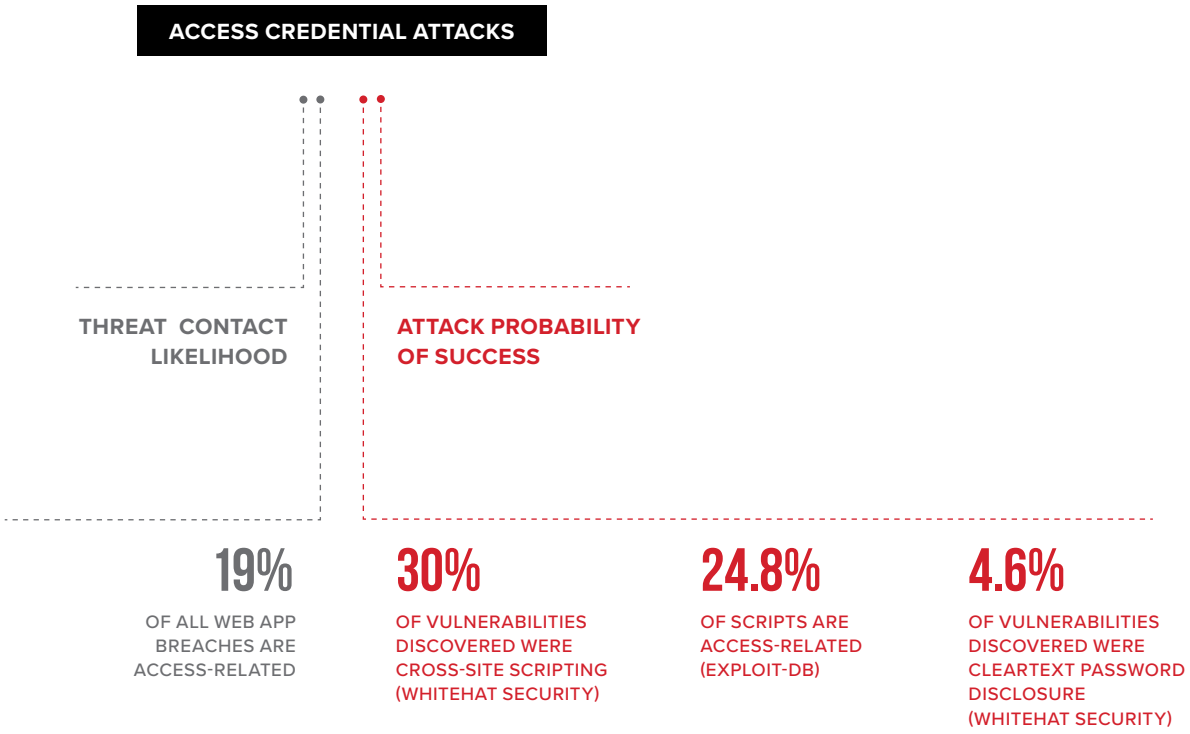
The keys to applications are the access credentials. Once attackers have either stolen credentials or hijacked a login in process, they can fully impersonate a user.

Botnets orchestrate the majority of attacks against application access. Botnets used to be made up of home computers, but now they are primarily IoT devices such as IP cameras, televisions, and home Internet routers. These fleets of botnets can be used to brute force accounts, test stolen passwords, or look for web apps with weak access controls. According to the breach notification letters we reviewed for 2017 and Q1 2018, Figure 28 lists some of reported ways that access credentials were obtained.

Credentials can also be stolen directly from the user via XSS, man-in-the-browser attacks, man-in-the-middle attacks, malware, and phishing attacks (see figure 29). CSRF attacks can also hijack user sessions in progress to inject unauthorized commands directly into a site.

FIGURE 29: OTHER COMMON USER ATTACK PATHS

FIGURE 30: SUMMARY OF ACCESS CREDENTIAL ATTACK LIKELIHOOD AND PROBABILITY OF SUCCESS



ONE OF THE UNFORTUNATE SIDE EFFECTS OF AN ATTACKER IMPERSONATING A LEGITIMATE USER WITH STOLEN ACCESS CREDENTIALS IS THAT THEIR UNAUTHORIZED ACCESS DOES NOT SET OFF ANY ALARMS.

One of the unfortunate side effects of an attacker impersonating a legitimate user with stolen access credentials is that their unauthorized access does not set off any alarms. The attack is often discovered long after the fact, usually after the victim reports fraud on their account. The access attacks that were reported to the California, Washington, Idaho, and Oregon state attorneys general involved unauthorized access leading to large-scale breaches. However, these statistics are just the tip of the iceberg. The cases reported for breach are large-scale, whereas singular fraud events are not subject to public disclosure. Because individual users can have their access hijacked in a one-off fashion, there are likely exponentially more access breaches occurring than we know about.

Figure 30 shows what data we have about the risk of access attacks: 24.8% of Exploit-DB scripts are access-related; 30% of vulnerabilities discovered by WhiteHat Security were XSS and 4.6% were cleartext password disclosure, both of which can lead to stolen user credentials; 19% of all web app breaches are access related.

Many web applications do not have sophisticated access control mechanisms that can detect and repel unauthorized access or stolen credentials. In some of the worst cases, access controls were set up improperly, leaving applications and databases exposed with little or no protection. Access control misconfiguration falls under the OWASP Top 10 risk *A5—Security Misconfiguration*.[16]

75% OF RESPONDENTS USE USERNAME AND PASSWORD CREDENTIALS UNIQUE TO EACH APPLICATION TO ACCESS CRITICAL WEB APPS.

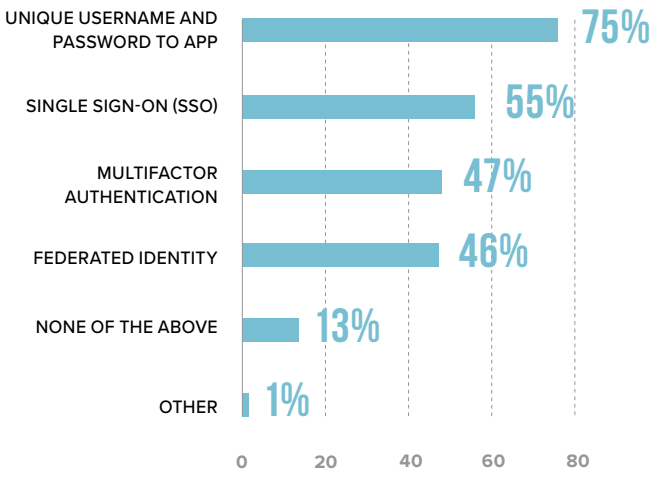


FIGURE 31: HOW ACCESS TO CRITICAL APPLICATIONS IS MANAGED (MULTIPLE ANSWERS ALLOWED)

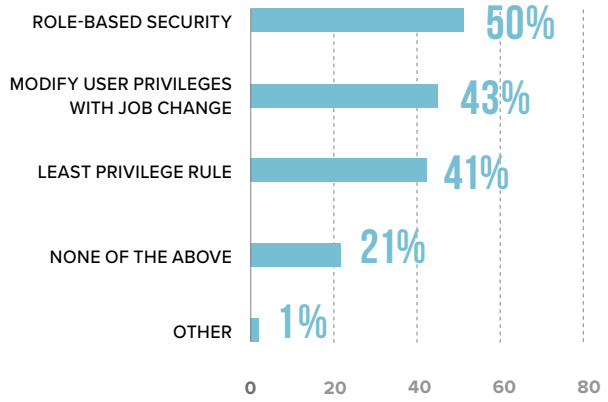
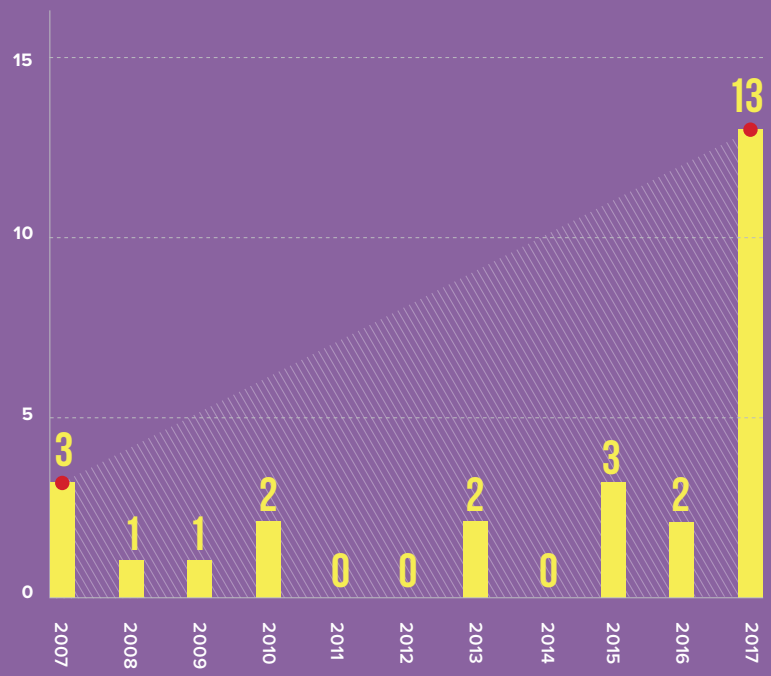


FIGURE 32: ACCESS CONTROL MODELS IN PLACE

Like most things in security, access is both a technological process as well as an operational process. With respect to how organizations manage their access to applications, the F5 Ponemon security survey asked respondents how they authenticated access to critical web apps used within their organizations. By far, the majority (75%) used username and password credentials unique to each application (see Figure 31). Another 55% said they used single sign-on. About half (47%) of respondents used two- or multi-factor authentication or federated identity (46%), and 14% said they either use some other means of authentication, or no authentication at all.

The same survey also asked respondents how their organizations authorize the use of, and access to, web applications. Here the results were slightly more encouraging, with many security teams following accepted operational practices. Fifty percent of respondents said they use role-based security; 41% said they use the “least privilege” rule, which gives users the least amount of access they need to do their jobs (see Figure 32). In addition, 43% said they modify users’ privileges when their job roles change. On the downside, however, as many as 22% use either some other means of access control or don’t control their users access to applications at all.



DESERIALIZATION ATTACKS HAVE BEEN AROUND FOR OVER A DECADE, BUT THEY EXPLODED IN 2017.

FIGURE 33: DESERIALIZATION EXPLOITS OVER PAST DECADE

DESERIALIZATION ATTACKS

Although deserialization attacks have historically been somewhat rare, they are becoming more prevalent, and they can be extremely devastating. In 2017, they cast a wide shadow with yet another Apache Struts vulnerability, which was a command injection attack facilitated by deserialization.[17] They have been around for over a decade, but they exploded in 2017, as shown in Figure 33.

Because applications are often deployed as swarms of services, each offering different components, they need some kind of communication method to interact. Often this communication is carried over HTTP, but the format of the data is also important. Serialization occurs when apps convert their data into a format (usually binary) for transport, typically from server to web browser, from web browser to server, or machine to machine via APIs. Java serializes most objects for transport and uses embedded libraries with formats like Remote Method Invocation (RMI) or Java Management Extensions (JMX).

FIGURE 34: DESERIALIZATION ATTACK PATH

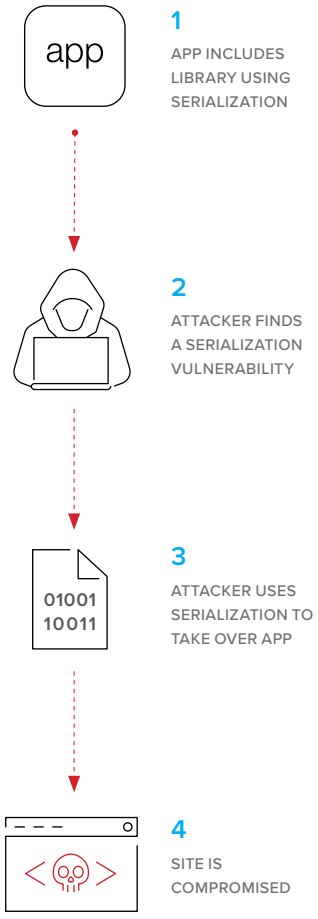


FIGURE 35: DESERIALIZATION ATTACK LIKELIHOOD AND PROBABILITY OF SUCCESS

Attackers can embed commands (as was the case with Apache Struts) or tamper with existing parameters in that serialized data stream. If the application deserializes the data stream without filters[18] or checks, these attacks can flow directly into the heart of an application. In the case of the Apache Struts vulnerability, an attacker was able to execute arbitrary code or shell commands in the context of the server running the XStream process, giving the attacker full access to the application itself.

A basic tenet of security is to always scan and screen data from untrusted sources. Many applications designed for fat clients use serialization to send data to the app. However, a web client is not a trusted source; anything that runs in a browser can be altered to include attack code.

This type of attack is such an up-and-coming threat that it was added in 2017 to the OWASP Top 10 as risk *A8—Insecure Deserialization*.[19] This was done based on an industry survey, indicating that a significant number of web application security experts see this as an important problem.

Although the numbers are low, as shown in Figure 35, deserialization is an emerging threat that bears keeping an eye on.

DESERIALIZATION IS SUCH AN UP-AND-COMING THREAT THAT IT WAS ADDED TO THE OWASP TOP 10 IN 2017.

143,000,000

BREACH OF 143 MILLION AMERICANS' PERSONALLY IDENTIFIABLE INFORMATION WAS TIED TO DESERIALIZATION VULNERABILITY.

ADVANCED PERSISTENT THREATS TO APPLICATIONS

Advanced persistent threats (APTs) are attacks that require time for reconnaissance, testing, and preparation of custom exploits. These kinds of attacks are rare, depending on how valuable a target is to an adversary. A typical example might be a nation-state spy agency attacking a web-based email system to spy on dissident email conversations.[20] However, APTs also have powerful impacts and are difficult to detect, making them a threat worth considering.

76% OF THE VULNERABILITIES DISCOVERED BY WHITEHAT SECURITY IN 2017 WERE CLASSIFIED AS ABUSE OF FUNCTIONALITY.

Abuse of functionality

Sometimes called a business logic attack, abuse of functionality manipulates an oversight in application design to do something beyond the application's intended functionality. The effects of these attacks can vary greatly based on the functionality and purpose of the app. Some classic examples of abuse of functionality are:

- A shopping application that accepts negative bids, which grant the shopper money back as well as the purchased good
- Subverting the password reset mechanism to verify passwords
- Using a site's file upload attachment feature to upload malware
- Repeatedly exercising a resource-heavy function to bog down an application in a denial-of-service attack

In 2017, 76% of the vulnerabilities discovered by WhiteHat Security were classified as abuse of functionality. These kinds of vulnerabilities are difficult to detect in an automated fashion, which is why WhiteHat Security also leverages human penetration testers in its analysis. Even still, the prevalence and impact of these kinds of attacks is limited only by the attacker's ingenuity.

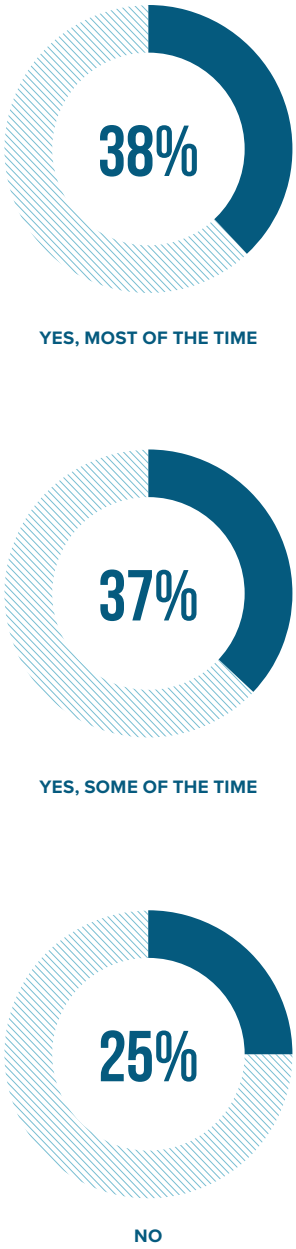


FIGURE 36: USE OF ADDITIONAL API ACCESS AUTHORIZATION PROCEDURES

Application programming interface attacks

Wherever an application accepts data, often by way of an application programming interface (API), it is a potential target for attack. Some application owners think that APIs are invisible to attackers since no human is supposed to interact with them directly. However, APIs are easily found by attacker reconnaissance scans and can be attacked by most of the traditional web application attack methods. APIs are especially enticing targets since they often have administrative capability within the application as well as direct access to valuable data stores.

APIs ARE ENTICING TARGETS SINCE THEY OFTEN HAVE ADMINISTRATIVE CAPABILITY WITHIN THE APPLICATION, AS WELL AS DIRECT ACCESS TO VALUABLE DATA STORES..

Because they are intended for machine-to-machine interactions, APIs sometimes also have different authentication schemes than the normal user interface. There have been cases where there is strong authentication in place for the main user login page but weaker authentication for the API. A common oversight is API authentication based on a single password or cryptographic key that is never changed or adequately tracked. In some of the worst cases, a singular shared secret is used for an entire organization's API access to the app. The most common error on GitHub is for the application programmer to accidentally include the API key in the source code. This is so common that GitHub scans for it regularly. In fact, because of the unfettered access APIs have to an application and its data, there should be stronger access control in place for APIs than for users.

The F5 Ponemon security survey asked respondents if they deploy additional authentication for their APIs. While 25% of respondents said no, an encouraging 75% indicated that they do either some of the time or most of the time.


TOP SECRET

APP INFRASTRUCTURE ATTACKS

Attackers have learned that if it's too much trouble to break into the application or steal access credentials, they can strike at the application's supporting infrastructure instead. Although the number of actual breaches that stem from infrastructure attacks is low compared to attacks against the application itself (because application attacks are typically easier to pull off), there are significant cases where application infrastructure provides attackers an easy target.

25.4%

WEAK TRANSPORT LAYER ENCRYPTION MAKES UP 25.4% OF THE REPORTED DYNAMIC TESTING VULNERABILITIES.

This section looks at the numerous attacks against the different tiers that power the application, including the encryption, certificates, domain name services (DNS) and networks tiers that knit everything together.

WhiteHat Security vulnerability data shows that weak transport layer encryption makes up 25.4% of the reported dynamic testing vulnerabilities. Even more worrisome, the trending of vulnerabilities in 2017 is going up, as shown in Figure 37. But weak transport layer protection isn't the only problem. Several high-profile breaches that appeared to be encryption breaches were, in reality, DNS hijacking events. Those attacks are also examined in more detail.

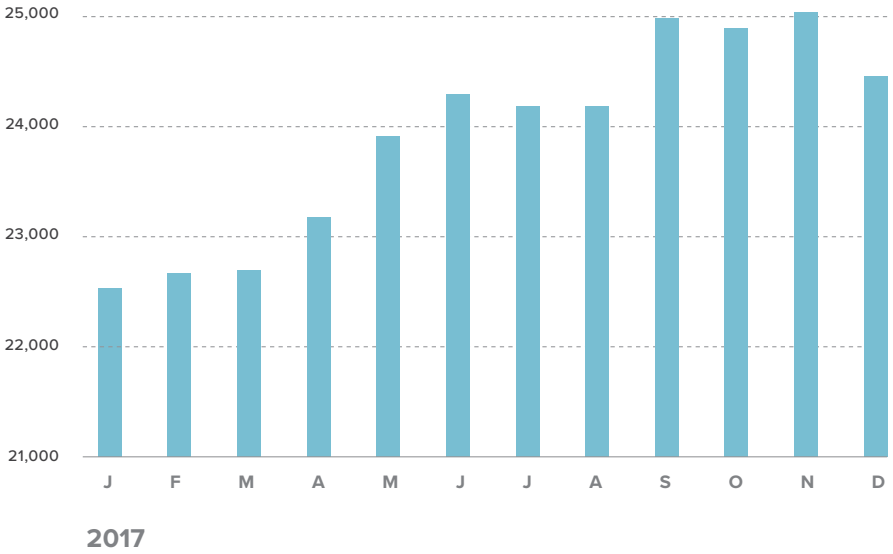
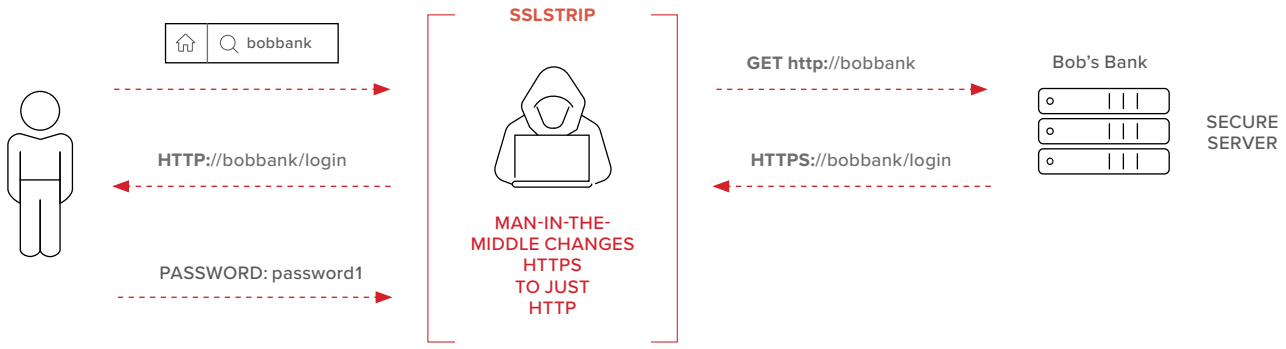


FIGURE 37: WEAK TRANSPORT LAYER PROTECTION FINDINGS BY MONTH IN 2017

ATTACKS AGAINST TRANSPORT LAYER PROTECTION

The most common transport layer protection is the Transport Layer Security (TLS) protocol and its predecessor, Secure Sockets Layer (SSL). These are the world's de facto transport security protocols, long ago surpassing IPsec, traditionally used for VPNs. Without encryption, networks are vulnerable to being spied upon or having data modified as it traverses untrusted networks (like the Internet or public wireless networks). The most powerful attacks are man-in-the-middle (MitM), which strip away all privacy and let attackers snoop and modify data while the victim is completely unaware. Figure 38 shows how one MitM attack called SSLStrip[21] tricks a victim (in this case, Bob) so that he never establishes an encrypted connection with the target server (Bob's bank), thus allowing the attacker to steal Bob's bank login credentials.

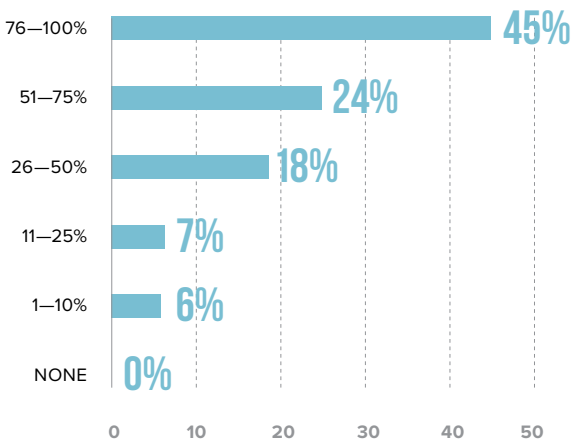
FIGURE 38: SSLSTRIP MAN-IN-THE-MIDDLE ATTACK PATH



A less impactful but still annoying and worrisome attack is advertising injections on unencrypted web sessions. This has reportedly happened in airport[22] and hotel[23] wireless hotspots, and with Internet Service Providers.[24] Imagine what kind of dangerous malware or code injection a more aggressive MitM attack could introduce.

What does "weak" mean with respect to transport layer protection? In the worst case, it means using no transport layer protection at all. We asked survey participants what percentage of their web applications use encryption. Less than half (45%) of respondents said the majority (76% to 100%) of their web apps use TLS/SSL (see Figure 39).

FIGURE 39: PERCENTAGE OF APPLICATIONS THAT USE SSL OR TLS



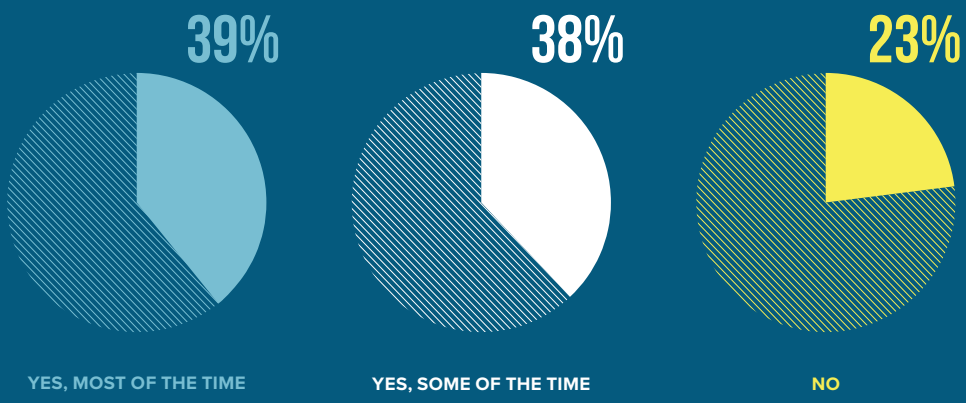


FIGURE 40: ORGANIZATIONS THAT USE ENCRYPTION FOR DATA AND APPLICATIONS IN TRANSIT

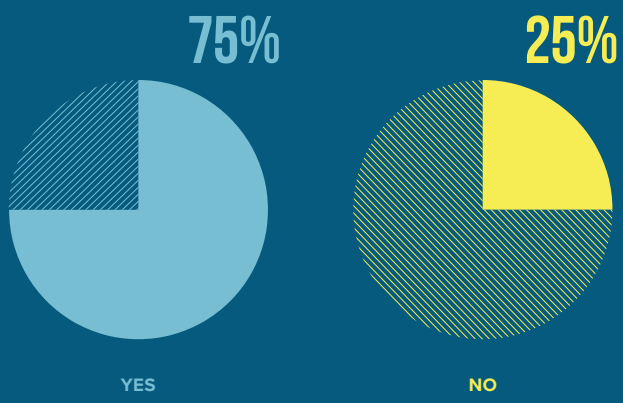


FIGURE 41: PERCENTAGE OF ORGANIZATIONS THAT KNOW THEIR LEVEL OF ENCRYPTION

When it comes to encrypting data in transit, 77% of respondents said they do, either some, most, or all of the time; 23% do not (see Figure 40).

Beyond not using encryption, the next worse case is using obsolete, vulnerable ciphers or short encryption keys. The first step in securing transport layer encryption is to be aware of what you're using so you can compare it to modern standards. We asked that question in the F5 Ponemon security survey and found a quarter of organizations don't know what encryption they're using.

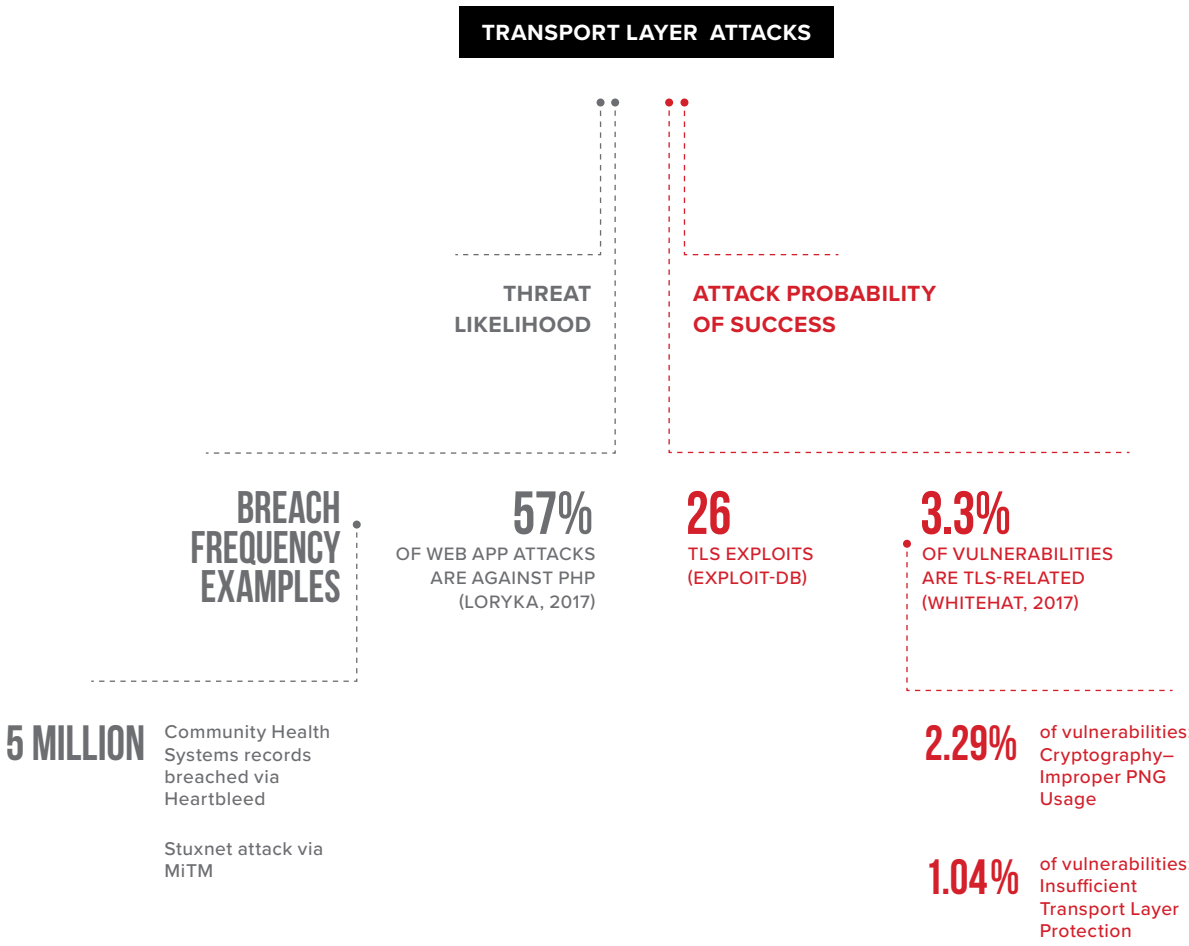
BEYOND NOT USING ENCRYPTION, THE NEXT WORSE CASE IS USING OBSOLETE, VULNERABLE CIPHERS OR SHORT ENCRYPTION KEYS.

Specifically, what is a weak configuration for TLS? The following are definitely known to be broken or easily compromised:

- All versions of SSL
- DES, RC4-40, DHE-RSA-Export, MD5, RC4 algorithms
- Keys smaller than 128 bits
- SSL certifications smaller than 2048 bits

Encryption compliance standards are always updated as new cryptographic attacks are found and better computing hardware is rolled out. It's best to keep an eye on strong encryption standards, such as the U.S. National Institute of Standards FIPS PUB 140-2, *Security Requirements for Cryptographic Modules*[25] or the standards tracking site.[26]

FIGURE 42: SUMMARY OF TLS ATTACK LIKELIHOOD AND PROBABILITY OF SUCCESS



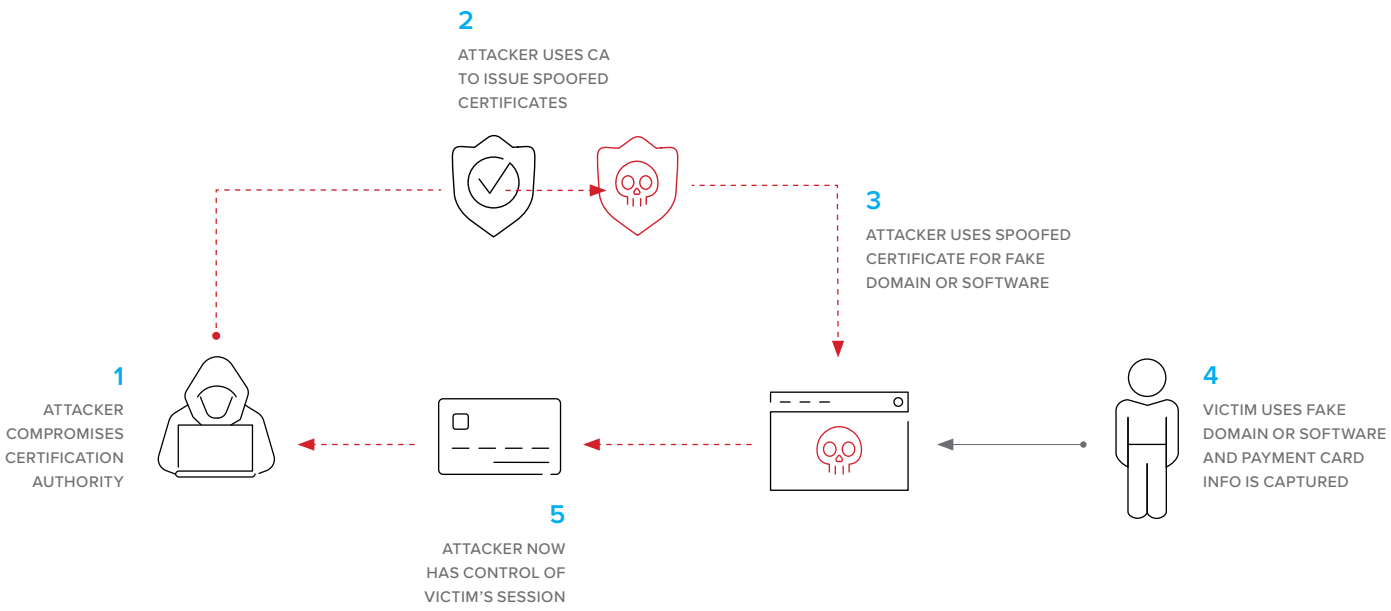
What about specific transport layer attacks and data related to potential attacks? Figure 42 shows a breakdown of data related to this threat.

New, named TLS protocol vulnerabilities are released about twice a year. However, with the exception of Heartbleed, the majority of named TLS protocol vulnerabilities are academic and rarely used in an actual breach. One of the biggest involved Community Health Systems (CHS) in 2014. CHS lost nearly five million social security numbers when an attacker used the Heartbleed vulnerability[27] to compromise a Juniper Networks SSL/VPN. At the time, it was the biggest data breach to be attributed to Heartbleed.

Perhaps the greatest TLS/SSL attack of all time was the Stuxnet malware.[28] Stuxnet targeted the centrifuge control systems in Iranian nuclear reactors; one of its infection vectors was a MitM attack against the Windows Update server. Windows Update is protected with TLS/SSL, but the nation-state that pulled off the attack was able to fool the client and use a MitM attack against the SSL/TLS connection by calculating an MD5 hash collision in real time. Such an attack had been entirely theoretical until then.

One of the most impactful but rarely mentioned vulnerabilities for TLS is the security of the random number generators that underlie the entire foundation. Truly random numbers are difficult to come by on a computer, and bad random data can lead to predictable key generation, as was discovered during the legendary Debian-SSH key debacle of 2008.[29] Similarly, researchers found that 1 in 100 SSL keys were guessable due to bad random seeds.[30] That is an unacceptable rate for an asymmetric cryptographic system whose keys are supposed to take thousands of years to brute force.

FIGURE 43: ATTACK PATH FOR COMPROMISED DIGITAL CERTIFICATES



CYBERCRIMINALS AND STATE-SPONSORED HACKERS CONTINUE TO LEVERAGE CERTIFICATE COMPROMISE FOR SPYING OR DISGUISING MALWARE.

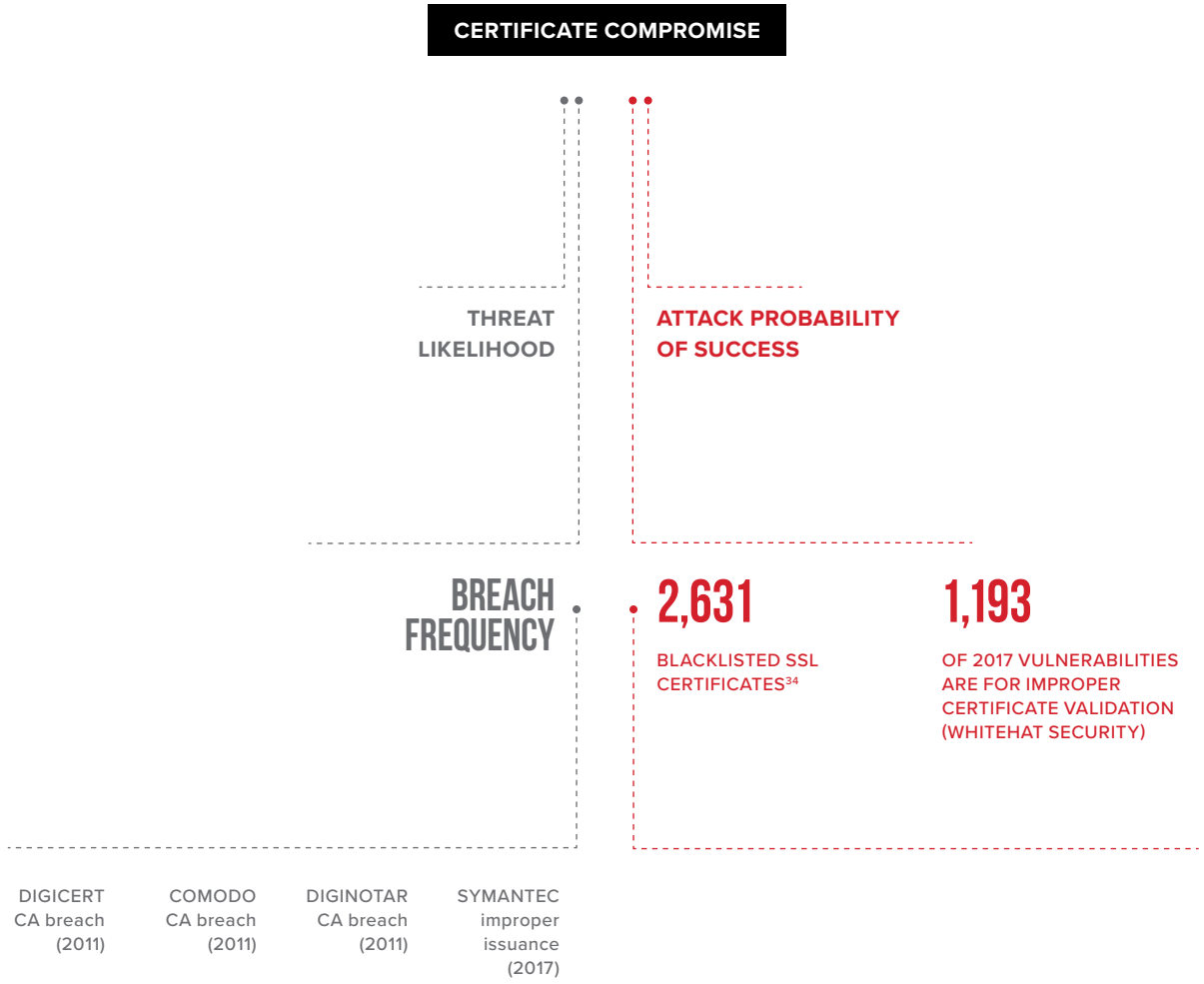
COMPROMISED CERTIFICATES

Digital certificates are the anchor for the server side of TLS/SSL that authenticates the app server for the user. A valid certificate from a trusted certification authority (CA) verifies the identity of the certificate. The key words are "valid" and "trusted" since there are cases when invalid certificates are accepted. Even more insidious are trusted CAs that are shown to be untrustworthy. Both cases can lead to victims being fooled into running or using imposter applications that, in turn, lead to malware infection or stolen credentials.

Given how much we rely on certificates, when fraud happens, it is devastating. Consider these major cases:

- In 2011, an attacker fraudulently obtained DigiCert Group certificates to spoof Google, Yahoo, Microsoft, and Mozilla.[31]
- In this case, the attack was traced to Iran. Indeed, sophisticated cybercriminals and state-sponsored hackers continue to leverage certificate compromise for spying or disguising malware.
- And again in 2011, DigiNotar CA[32] was breached, which led to the collapse of the DigiNotar company, which failed to notify Mozilla of the breach, even though DigiNotar had signed several of Mozilla's own certificates.
- In 2015, the China Internet Network Information Center (CNNIC) issued an intermediate certificate with no practice statement to MCS Holdings in Egypt. This certificate could be used to forge certificates. Google responded by distrusting these certificates in their Chrome browser.[33]

FIGURE 44: SUMMARY OF CERTIFICATE COMPROMISE LIKELIHOOD AND PROBABILITY OF SUCCESS



OVER 23,000 CERTIFICATES WERE REVOKED WHEN THE PRIVATE CERTIFICATES, WHICH HOLD THE SECRET KEYS, WERE EMAILED TO THE WRONG COMPANY.

After these major attacks, trust in CAs continues to be a problem. In 2017, Symantec's Certificate Authority business was under assault from Google for the improper issuance of Google certificates.[35]

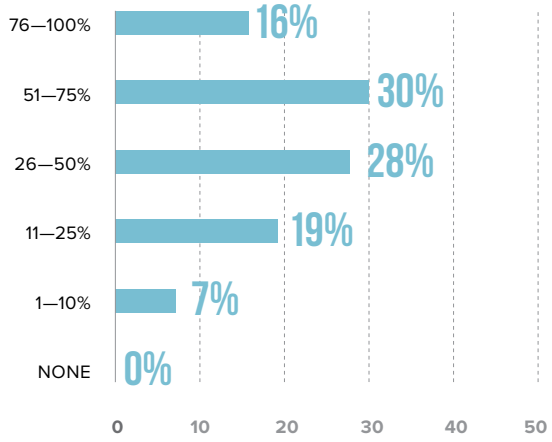
In 2018, Trustico stated their Symantec, GeoTrust, Thawte, and RapidSSL certificates were compromised.[36] Over 23,000 certificates were revoked when the private certificates, which hold the secret keys, were emailed to the wrong company.

The increased risk of using self-signed certificates

One of the crucial values of a certificate is that it authenticates the site it runs on. To do this effectively, the certificate should be signed by a trusted third party. Usually, this is a certificate authority such as Comodo, Entrust, GlobalSign, or Let's Encrypt. Without a trusted signature, users cannot verify the authenticity of the certificate, which means that the authenticity of the app is also in question.

However, there are a significant number of applications that present a self-signed certificate, which provides no proof of validity for the app. Usually a website with a self-signed certificate is an un-configured device and shouldn't be on the untrusted Internet. It's a bit disconcerting to know that as many as 46% of F5 Ponemon security survey respondents said that more than half (51% to 100%) of their web apps use self-signed certificates (see Figure 45). The good news is that F5 Labs research is showing that the prevalence of self-signed certificates is dropping quickly.

FIGURE 45: PERCENTAGE OF WEB APPLICATIONS THAT USE SELF-SIGNED CERTIFICATES



THE GOOD NEWS IS THAT F5 LABS RESEARCH IS SHOWING THAT THE PREVALENCE OF SELF-SIGNED CERTIFICATES IS DROPPING QUICKLY.

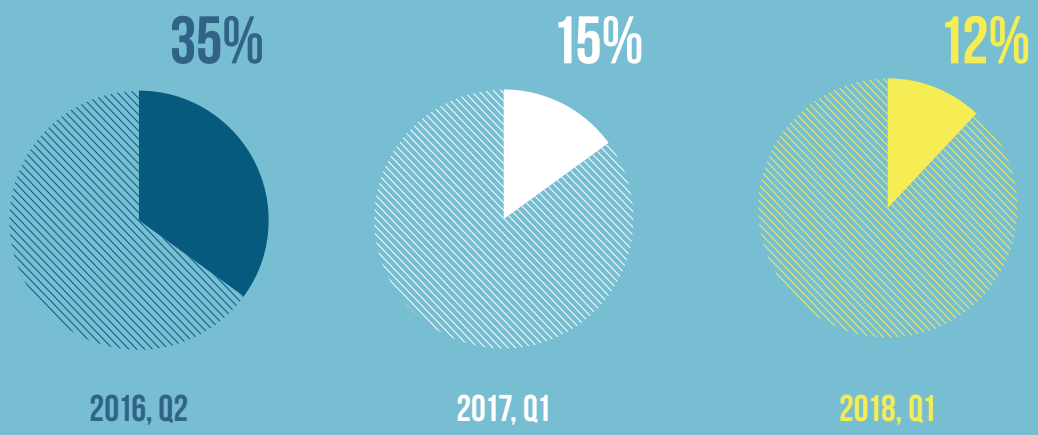


FIGURE 46: PREVALENCE OF SELF-SIGNED CERTIFICATES

THERE ARE 2 WAYS ATTACKERS TRY TO COMPROMISE DNS: THEY EITHER INTERCEPT THE DNS TRAFFIC ITSELF, OR THEY ATTACK THE REGISTRAR OF THE DOMAIN.

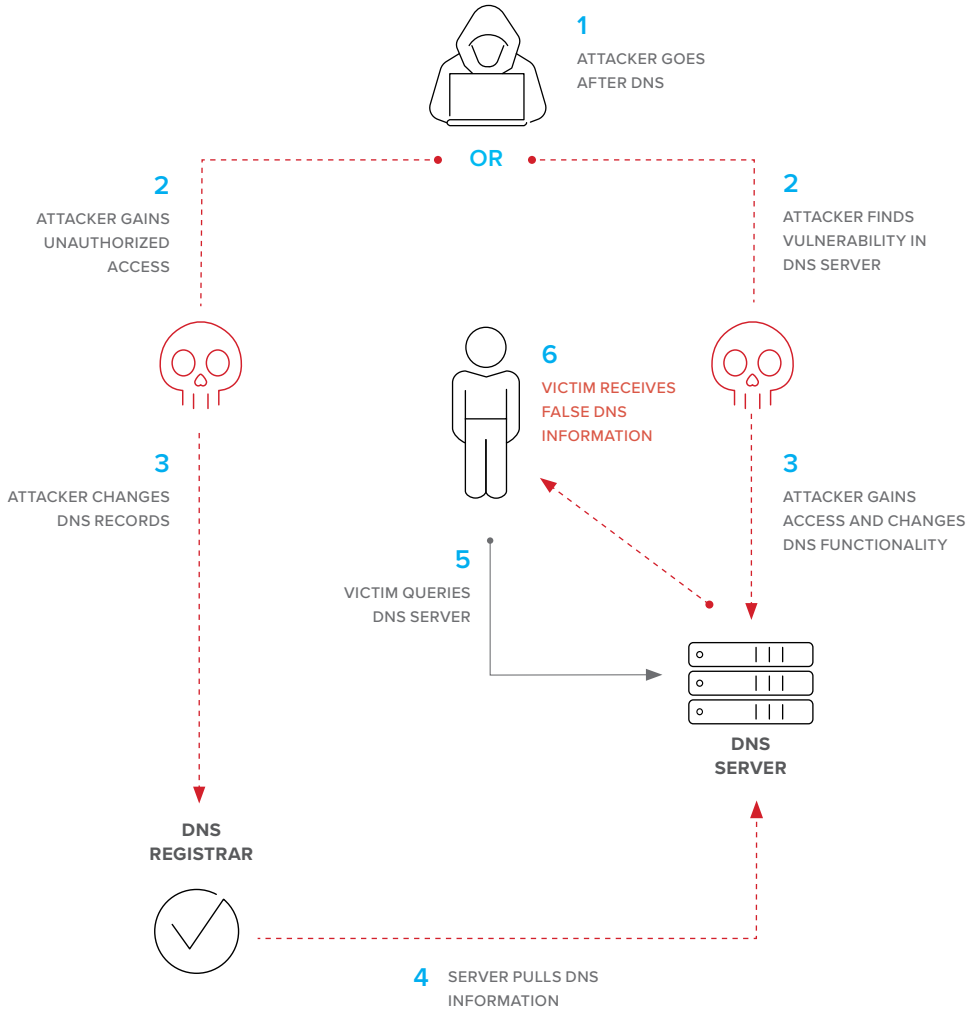


FIGURE 47: DNS HIJACK ATTACK PATH

DOMAIN NAME SERVICES HIJACKING

Another way to attack an application indirectly is through DNS. Even if an application itself were completely bulletproof, a DNS attack can still subvert or shut down an application.

There are two main vectors through which attackers will try to compromise DNS: they either intercept the DNS traffic itself, or they attack the registrar of the domain to modify or insert their own DNS records. This can be as easy as compromising the domain owner's email to obtain their login credentials to the domain registrar's app. These changes flow through the legitimate DNS servers to the users, as shown in Figure 47.

DNS is a critical piece of infrastructure for applications. The most at-risk applications are the web sites with access to valuable services or data. When DNS attacks happen to these large web applications, the affected victims can easily number in the thousands, and the damage can run into the millions of dollars.

FOLLOWING IS A LIST OF THE MAJOR DNS BREACHES OF THE PAST FEW YEARS:

2018

- BGP poisoning with DNS spoof for MyEtherWallet cryptocurrency theft[37]
- Point-of-sale malware steals credit card data via DNS queries[38]
- Domain theft strands thousands of web sites[39]

2017

- Brazilian bank domain hijacked, customers looted[40]
- Fox-It.com MitM attacks occur due to domain registrar hack[41]

2015

- St. Louis Federal Reserve suffers DNS breach[42]

2014

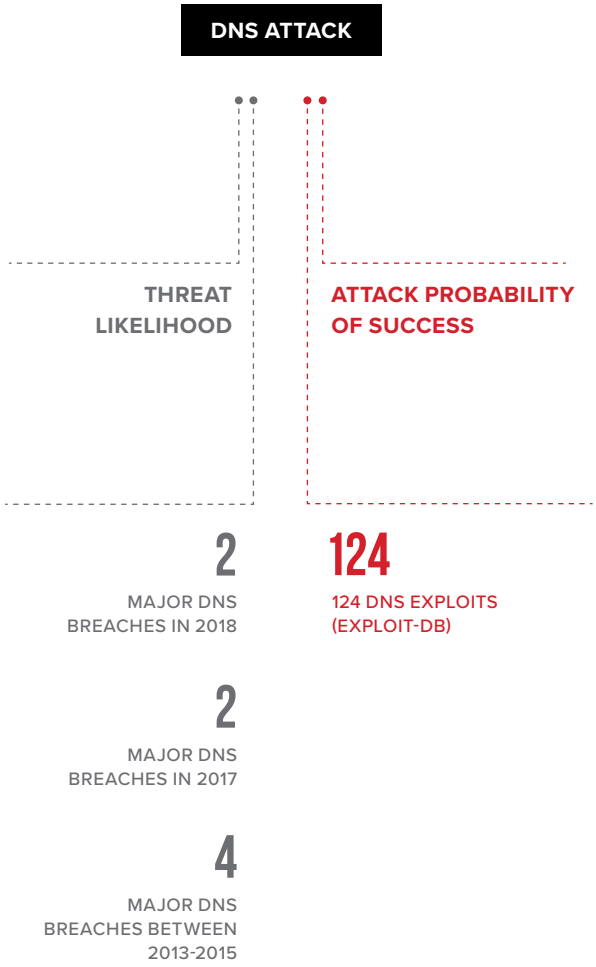
- ICANN attacked and the DNS admin system compromised[43]
- Attackers redirect the domain catholichealth.net and expose Catholic Health Initiatives patient emails[44]

2013

- Dutch DNS server hack: thousands of sites serve up malware[45]

DNS attacks are uncommon, but they are catastrophic to an application.

FIGURE 48: SUMMARY OF DNS ATTACK LIKELIHOOD AND PROBABILITY OF SUCCESS



DENIAL-OF-SERVICE ATTACKS

This section focuses on denial-of-service (DoS), and distributed denial-of-service (DDoS) attacks against applications. Here, we make a distinction between network DDoS attacks and application DDoS attacks.

$20

THE GROWTH IN IOT EXPLOITS THAT ARE BUILDING POWERFUL THINGBOTS HAS ENABLED ATTACKERS TO OFFER 300 GBPS ATTACKS FOR ONLY $20.

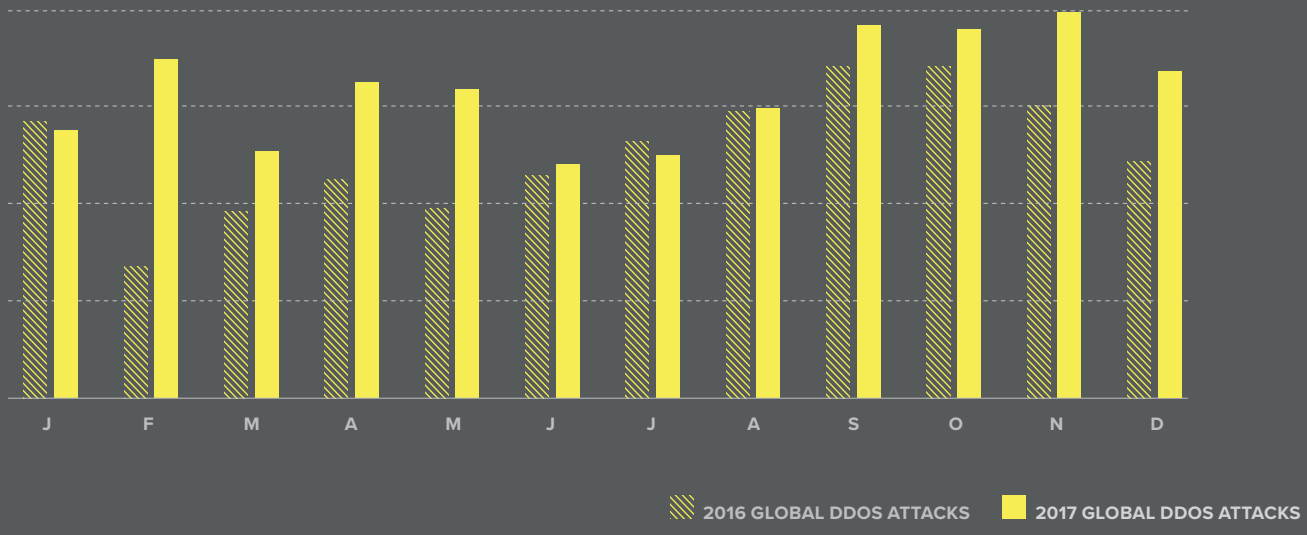
With network DDoS attacks, there are known limits of packet types such as TCP or UDP, known limits on TCP headers flags like URG or RST and, mostly, known limits on spoofing mechanisms. All of these limits make mitigating network DDoS attacks more of a science and less of an art. On the other hand, mitigating attacks against applications is more arcane, requiring custom scrubbing and talented defenders. Because application DDoS attacks are multi-host, multi-origin, and multi-vector, they are more powerful and more difficult to block. With this new trick in their arsenals, attackers are ramping up their usage of application DDoS attacks. While application attacks are often singular attacks, there are also extended campaigns that span days and weeks.

An application can be locked down tight against confidentiality and integrity attacks but still find itself very vulnerable to loss of availability from a DoS attack. It takes very little effort on an attacker's part to crush an application site under the heavy load from a botnet. DoS attacks against applications are not going away. They are not pranks or vandalism anymore, but useful tools in an attacker's arsenal.

In F5 Labs' Hunt for IoT report series, we profile DDoS thingbots—botnets composed of IoT devices—of immense size that are being leveraged for massive typhoons of DDoS attacks. Corrupting IoT devices to become DDoS launchers is a growth industry that is maturing as rapidly as we are discovering thingbots:

- DDoS is the most common attack launched from thingbots, followed by mining cryptocurrency, hosting banking trojans, and launching permanent denial-of-service (PDoS) attacks that turn IoT devices into inoperable bricks of plastic and silicon.
- The growth in IoT exploits that are building powerful thingbots has enabled attackers to offer 300 Gbps attacks for only $20.
- The discovery of DDoS thingbots began in 2008, but 64% of the DDoS thingbots we currently know about were discovered just since 2016.

FIGURE 49: F5 SILVERLINE DDOS ATTACK TRENDS BY MONTH, 2016-2017



Our internal statistics (Figure 49) from F5's Silverline DDoS scrubbing service show a slow and steady increase in DDoS attacks year over year.

In general, there are several different types of denial-of-service attacks (Exploit-DB has 5,665 DoS exploits in its database). The most basic attack uses a known exploit that will crash a running application or application-supporting service. Currently, network volumetric attacks are still very popular. These attacks come in two flavors: a direct flood of traffic from multiple sources or an asymmetric reflective attack built up from bouncing off exploitable services. However, denial-of-service attacks against an application cannot be discounted. Since first classifying DDoS attacks against applications in 2016, we have seen a steady increase in application-targeted attacks year over year.

FIGURE 50: DDOS ATTACKS BY CATEGORY, 2016 THROUGH 2017

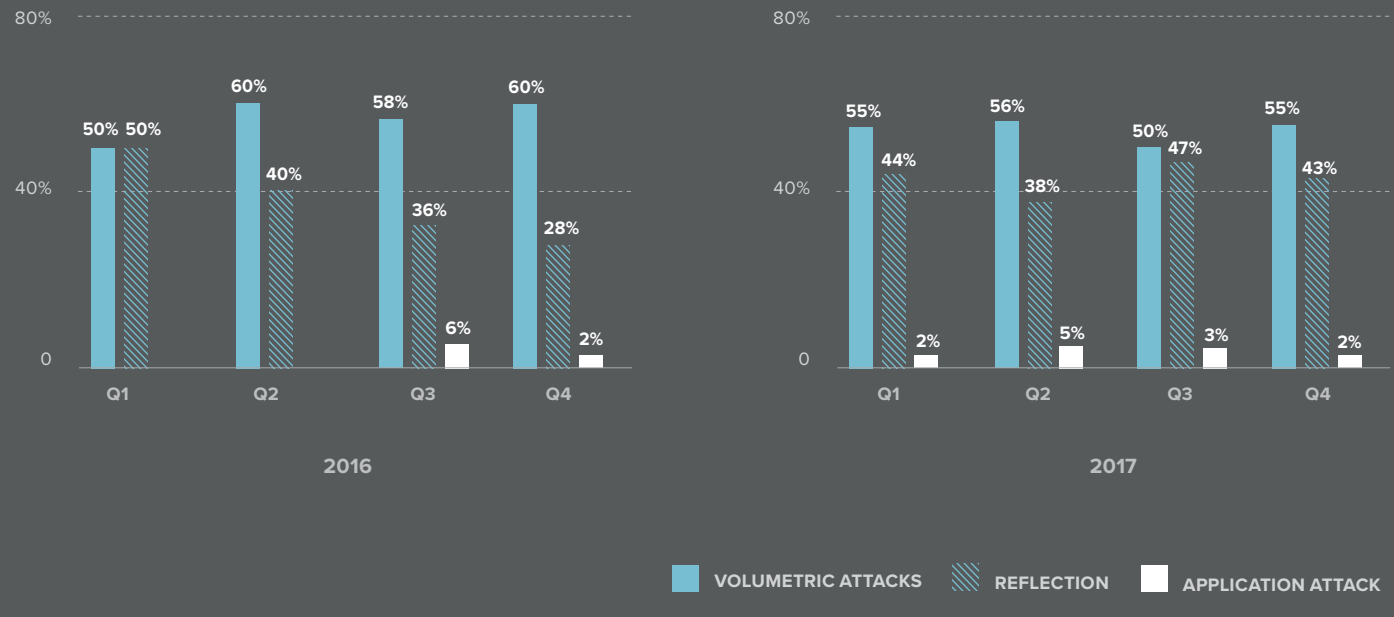
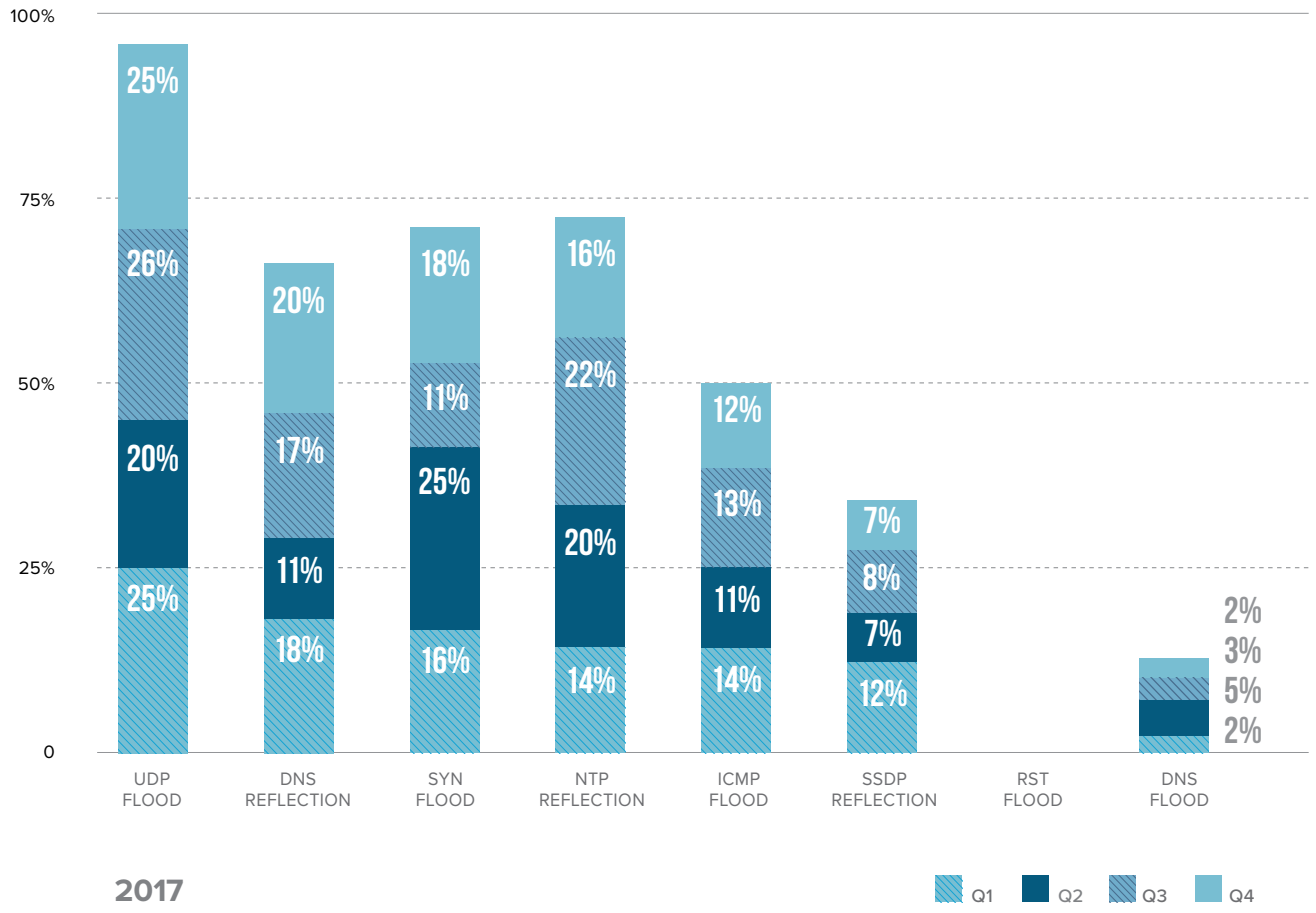


FIGURE 51: 2017 DDOS ATTACKS BY PROTOCOL



For an application that is vulnerable to resource exhaustion, a simple script running on a hijacked bot can generate a huge amount of load, from a focused request to the most resource-intensive components in the application. Attackers often leverage multithreading to amplify these kinds of strikes.

Old-school UDP and SYN network flood attacks are still popular, but defenders have learned to block these simple attacks. So, attackers have had to up their games with newer, better network strikes, as shown in figure 51. One way is with IPv6 DDoS attacks, which some anti-DDoS solutions can't block yet.[46] With IPv6 adoption closing in on 25% of the Internet address space, we are sure to see more of these attacks in the future.[47]

Another new vector for DDoS is Generic Routing Encapsulation (GRE) attacks using the popular point-to-point encapsulation protocol. Although GRE can't be spoofed, it is often allowed through firewalls and router filters because of its ubiquitous use for network tunneling. Recently, the Mirai thingbot used GRE attacks as part of its attacks against hosted DNS services at Dyn and Krebs on Security, leading to massive outages.

DDOS ATTACKERS ARE GETTING WISE TO THE FACT THAT SOME SITES POST CUSTOMER STATUS UPDATES FOR PATCHING AND MAINTENANCE.

DDoS attackers are getting wise to the fact that some sites post customer status updates for patching and maintenance. They then can target these maintenance windows for their attacks to inflict maximum operational chaos. Even worse, attackers are also timing DDoS attacks as diversions to cover data theft and fraud attacks being pulled off simultaneously while administrators are distracted.

DDoS ATTACKS CAN BE VERY COSTLY FOR ANY E-COMMERCE AND FINANCIAL INDUSTRIES THAT RELY ON THEIR INTERNET APPLICATIONS TO BE HIGHLY AVAILABLE.

In general, organizations find DDoS attacks to be a significant problem. The F5 Ponemon security survey asked about the impact of these kinds of attacks specific to application availability and found that 81% of respondents (see Figure 52) think that a DDoS attack resulting in the failure to access an application or data would be very painful.

Indeed, DDoS can be very costly for any industry that relies on their Internet applications to be highly available, such as the financial services industry or e-commerce (see figure 53). Consider the ramifications if a payment processor is unable to accept credit cards even for a few minutes.

None of the Ponemon survey respondents thought a DDoS attack impacting access to application or data would cost less than $10,000. And more than three-quarters of respondents thought a DDoS attack would cost between $500,000 and $10 million.

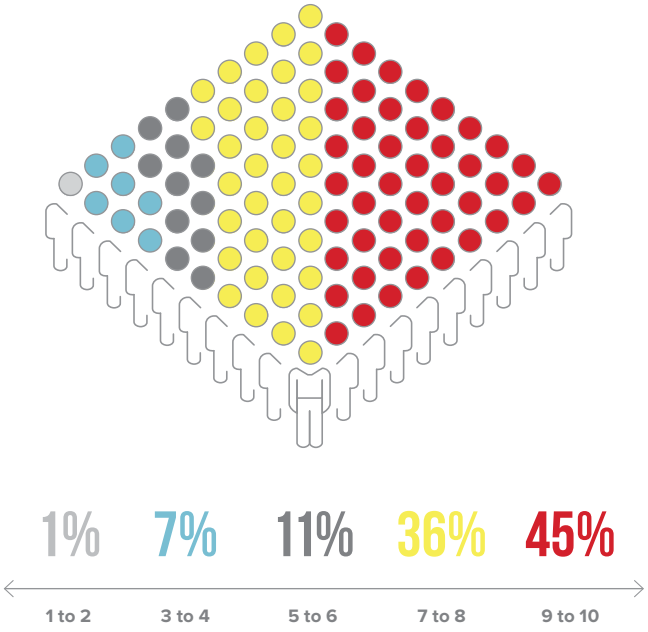


FIGURE 52: PAIN THRESHOLD OF A DDOS ATTACK CAUSING LACK OF ACCESS TO AN APPLICATION OR DATA

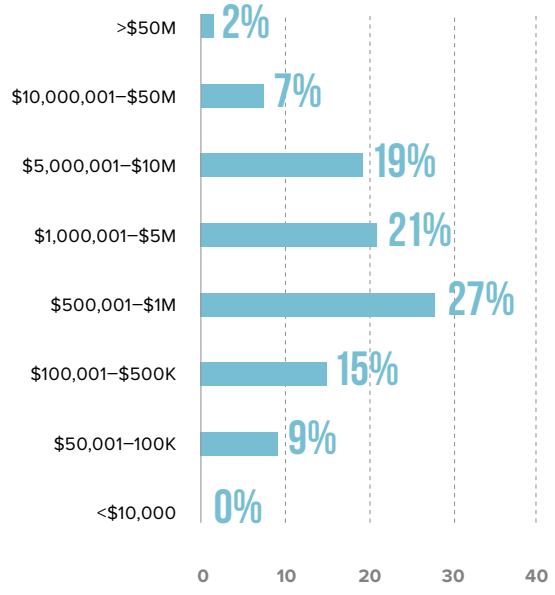
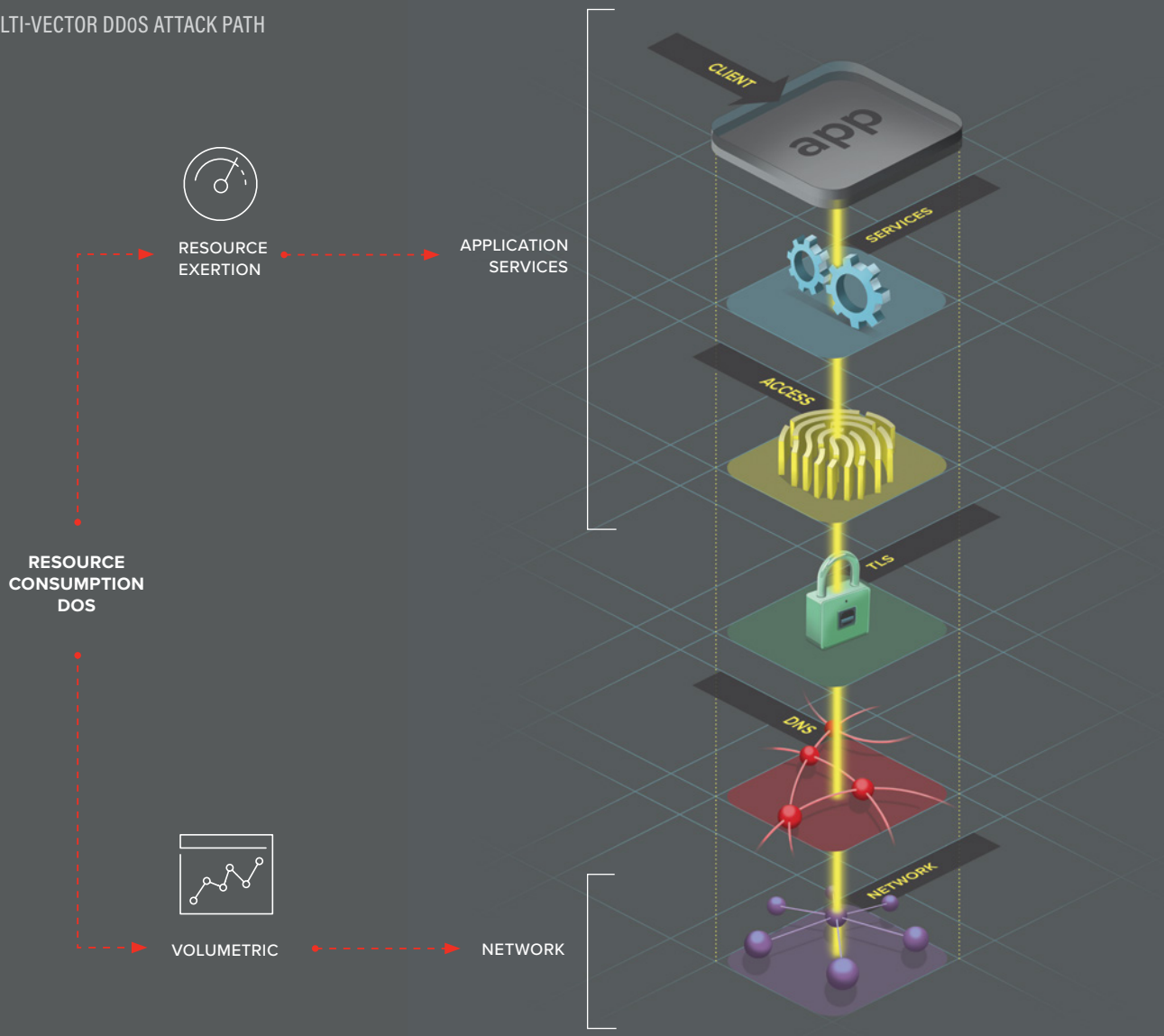


FIGURE 53: COST OF A DDOS ATTACK RESULTING IN LACK OF ACCESS TO AN APPLICATION OR DATA

FIGURE 54: MULTI-VECTOR DDOS ATTACK PATH



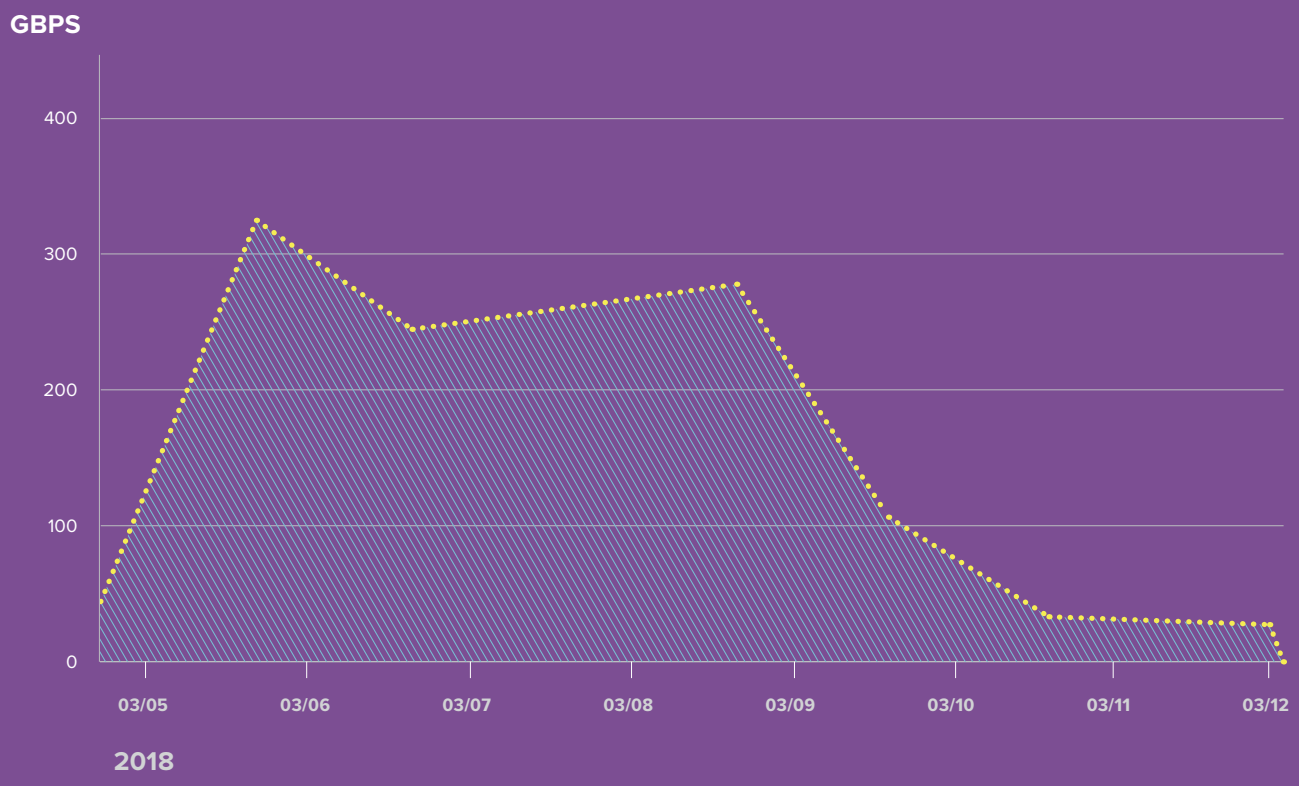
Multi-Vector DDoS

The most impactful DDoS attacks use not one attack vector, but many. Combining several different denial-of-service attack types into a sequence not only confuses mitigation solutions that are required to either scrub or drop traffic, but it can also magnify the volume of the attack, causing the most disruption possible to a web application, as we saw in March of 2018 with the 1.35 Tbps GitHub attack.[48]

Here's a common way a multi-vector DDoS attack happens (see figure 54):

1. A reconnaissance network scan occurs. This usually appears as hundreds of probes and port scans. The purpose of this attack is to find and measure targets for the later attack. Usually tens of thousands of network services are probed across the entire application infrastructure. The attacker is looking to identify network bottlenecks, backend servers, and resource-hungry application services.
2. Once the data has been analyzed, the extortion demand is delivered: "pay up or your site is toast."
3. Soon comes the DDoS attack. It begins with a traditional volumetric network flood that jams up network pipes and routing gear. This may be gigabytes in magnitude but it's just a distraction to keep the network operations team busy.
4. Then the real attack begins. It's an application-specific DDoS attack—Layer 7—against port 80. These new DDoS attacks target backend content delivery servers, overloaded routers, and resource-stressed application services. A favorite tactic is to submit web requests that trigger complex queries to bog down the whole application.

FIGURE 55: MULTI-VECTOR HIGH-VOLUME ATTACKS

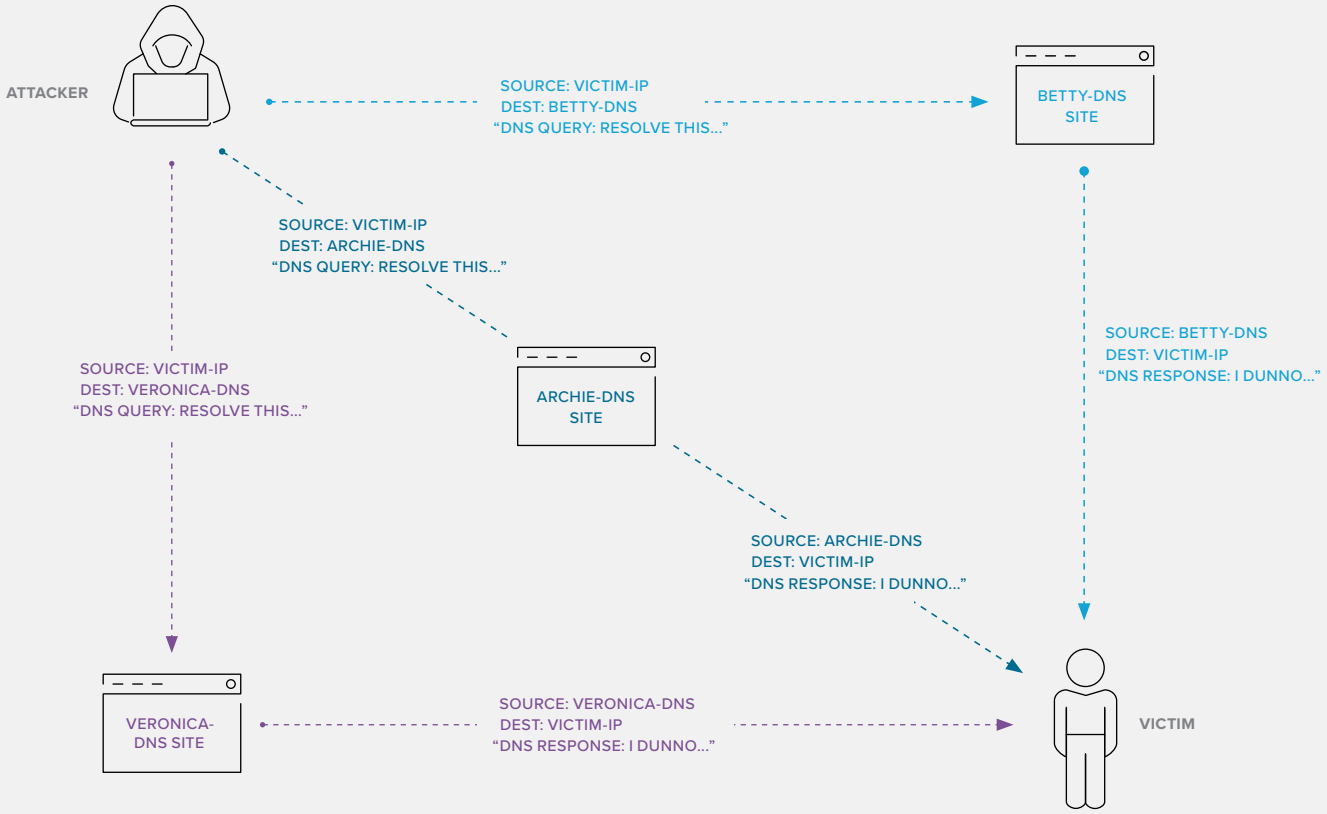


WHAT'S CRITICAL TO UNDERSTAND ABOUT THESE ATTACKS IS THAT THE ATTACKERS ARE TARGETING WHATEVER VULNERABLE POINTS IN THE INFRASTRUCTURE THEY FIND.

Attacks with the specific pattern shown in Figure 54 have been seen with alarming frequency by the F5 DDoS response team over the past year. They exhibit a specific methodology and use similar tools and attacks. Attacks have come in from hundreds of thousands of separate IP addresses, slamming in over 2,000 page requests per minute. They typically average around 170 gigabits per second and can top 325 gigabits per second, as shown in figure 55.

What's critical to understand about these attacks is that the attackers are targeting whatever vulnerable points in the infrastructure they find. This could be repeated calls to a specific URL that trigger intensive CPU load, a flood against a content distribution server feeding the application site, or even filling the pipe of a non-redundant network path. Wherever the application can be degraded or destabilized is a potential target.

FIGURE 56: REFLECTION DDOS ATTACK PATH



Reflection and amplification attacks

Attackers are also looking to improve even their most basic DDoS techniques. Yes, they are improving their attack scripts, as seen in the multi-vector DDoS attacks, but they are also subverting application services to do their dirty work.

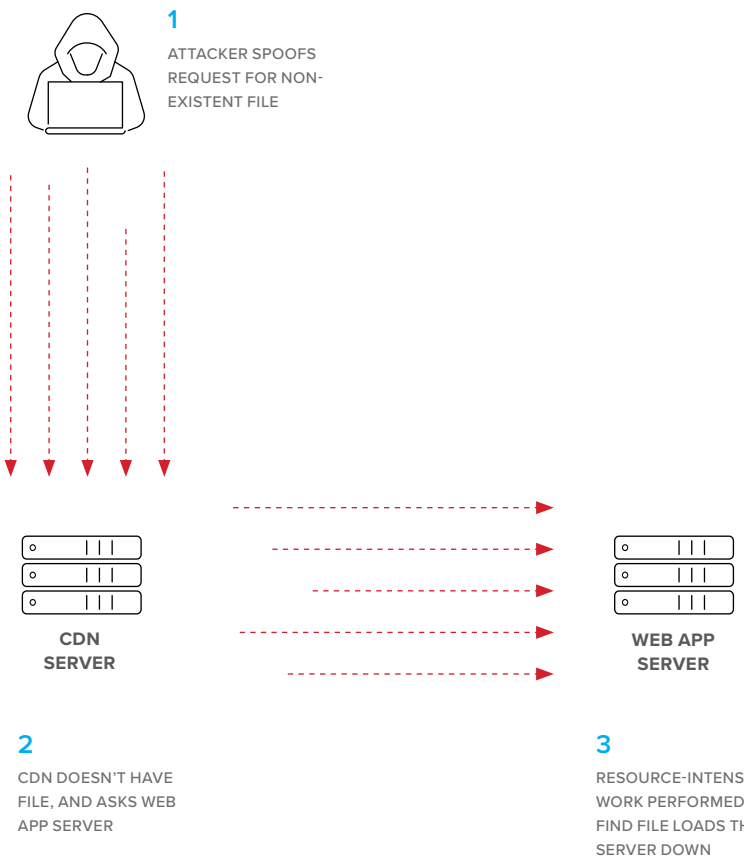
Network reflection attacks leverage your apps to attack yourself and others. Many use the simple UDP protocol which, by design, does not verify sender address, and that leads to easy spoofing. In general, UDP was never designed for trusted, mission-critical network operations. As the RFC clearly states, "Applications requiring ordered reliable delivery of streams of data should use the Transmission Control Protocol (TCP)."[49] However many useful infrastructure services have been built on UDP and thus are vulnerable to being used in a reflection attack. These include:

- Simple Service Discovery Protocol (SSDP)
- Lightweight Directory Access Protocol (LDAP)
- Universal Plug and Play (UPnP)
- Character Generator Protocol (CHARGEN)
- Session Initiation Protocol (SIP)

Internet Control Message Protocol (ICMP), another spoofable protocol, has also been used in reflection attacks such as the venerable Smurf attack.[50] Reflection attacks can also be amplified to enable an attacker to magnify the flood of network packets and also mask their true source address, as shown in figure 56.

The amplified DDoS technique is quite powerful and, although it has been around for decades, attackers move beyond network floods to pull off app-centric reflected amplification attacks. This means attackers are now undermining legitimate application services and infrastructure. A common method is to do DNS reflection attacks by sending spoofed DNS requests using UDP at open DNS servers. The spoofed requests appear to come from the DDoS victim, so that the DNS answers are returned by the server to blast the victim. A DNS attack like this has a high level of magnification; a small request can generate a hundredfold amplification of traffic.

FIGURE 57: CDN REFLECTION ATTACK PATH



ATTACKERS ARE GETTING SMARTER ABOUT LOOKING AT APPLICATION INFRASTRUCTURE AND WHAT CAN BE SPOOFED OR SUBVERTED.

Attackers are also looking to other application services for amplified reflection attacks. One attack grabbing headlines in early 2018 was Memcached DDoS, with attacks reaching the terabit level.[51] Memcached is a key component in the Services tier, but it isn't the only component exploitable for amplified reflection attack. Other components being reflected are content distribution network (CDN) devices. CDN servers hold cached popular content to help speed up web sites and apps. But a spoofed hash request for a non-existent file or image will cause a CDN to call back to the primary app server for (non-existent) data. This causes the CDN to apply additional load on the main web site instead of reducing it. With these kinds of DDoS attacks, an attacker can send a few web calls and cause an application's infrastructure to tear itself apart (see figure 57).

Reflection attacks are now taking over web clients with browser malware via XSS that launches layer 7 floods against web sites. A new WordPress Pingback DDoS attack has recently been seen in the wild. It exploits an automated notification mechanism on WordPress blog sites that sends a "pingback" POST request to a spoofed victim IP address.

Attackers are getting smarter about looking for these kinds of apps to weaponize. They're looking at application infrastructure and what can be spoofed or subverted. Expect more amplified reflection DDoS attacks bouncing off of application services in the future.

Transport Layer Security (TLS) denial-of-service attacks

There's a new set of theoretical, "brute-force, no-crypto" TLS attacks in which the client just sends random junk at a TLS stack that then tries to decrypt it.[52] The fact that the client is doing no crypto and just sending random bytes at the server restores the asymmetry of the attack. While this attack has been written about several times, it has never been seen in the wild. The TLS Internet Engineering Task Force (IETF) committee has mitigation plans for it if it ever becomes a popular attack, but it will require a change to the protocol itself.[53]

The original SSL renegotiation attack tool by the French group "The Hacker's Choice" repeatedly requested RSA key exchanges with a vulnerable server. These handshakes were ten times[54] more CPU-intensive for the server than they were for the attack client. This attack was seen at a prominent American bank and documented by one of its security analysts.[55] However, it's not seen much anymore, partly because of the global move to elliptic curve ciphers, which don't have the same asymmetry that made the original renegotiation attack so effective.

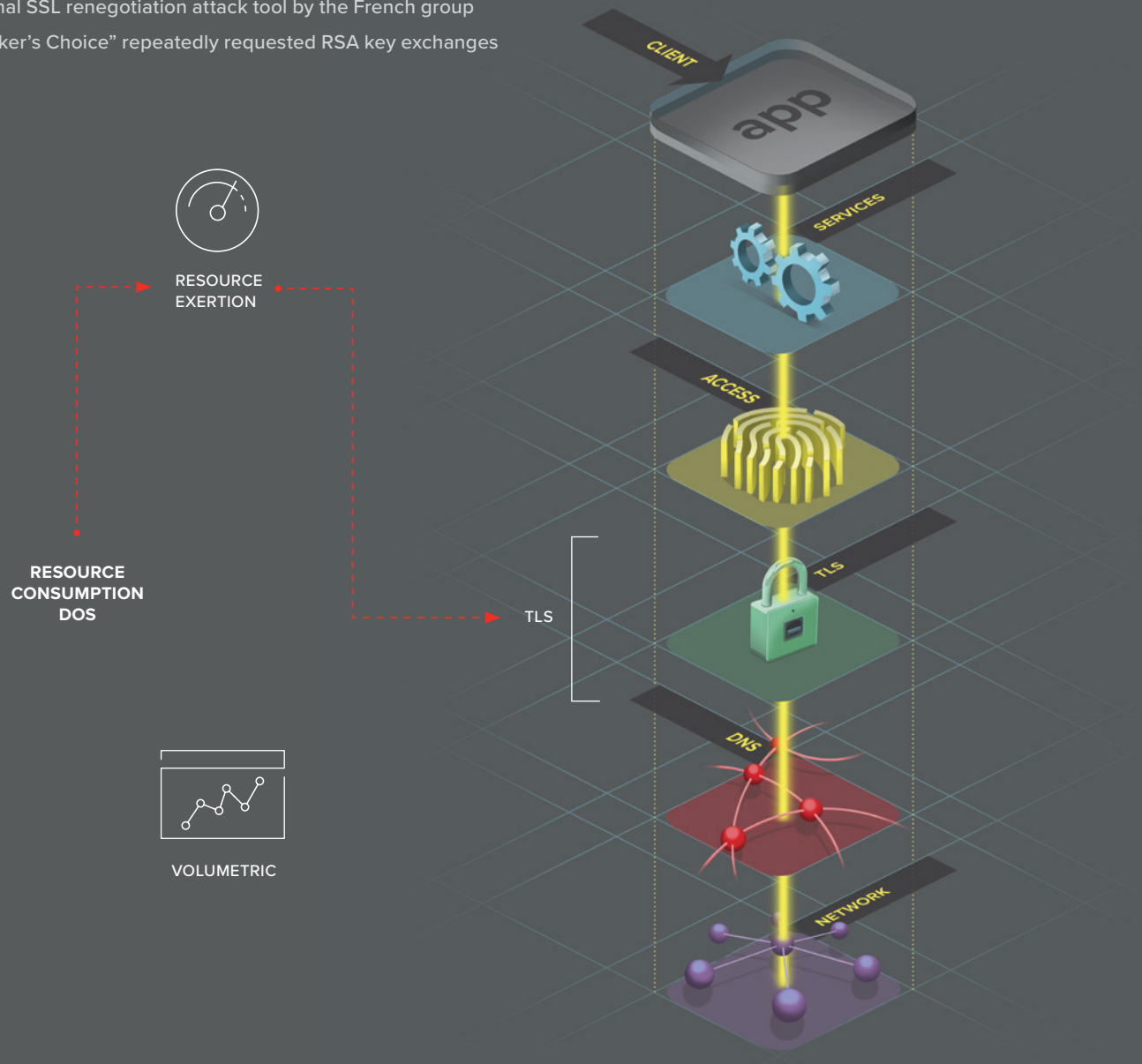


FIGURE 58: THEORETICAL TLS DOS ATTACK PATH

CLIENT ATTACKS

If attackers can't breach the application or the supporting infrastructure, then they can try to take over the app client. Most attacks against application clients are designed to steal access, either by directly stealing a user's credentials or by hijacking an authorized session in progress.

ATTACKS AGAINST CLIENTS AREN'T WELL PUBLICIZED BECAUSE IT'S INDIVIDUALS WHO ARE HIT, SO THERE'S LITTLE FANFARE OR REQUIRED DISCLOSURE.

Attacks against clients are usually not well publicized, especially compared to application breaches. This is because it's individuals who are hit, so there is little fanfare or required disclosure. However, en masse, these kinds of attacks can have significant impacts on not only the victimized user but often the application itself because it is also defrauded. That means the organization must deal with the clean-up costs.

This section explores significant application-related attacks against clients, which can involve the browser or a mobile app. In both cases, the client is communicating via the web to an application in order to retrieve, store, and process data. That authenticated connection is key to the context of these types of attacks.

In the F5 Ponemon security survey, we asked respondents which types of attacks (beyond DDoS) would be the most devastating to their organizations. The highest percentage of respondents (66%) chose man-in-the-middle, man-in-the-browser, and credential theft; followed by web fraud at 51%. Cross-site scripting, SQL injection, clickjacking and cross-site request forgery were all concerns, as well (see Figure 59).

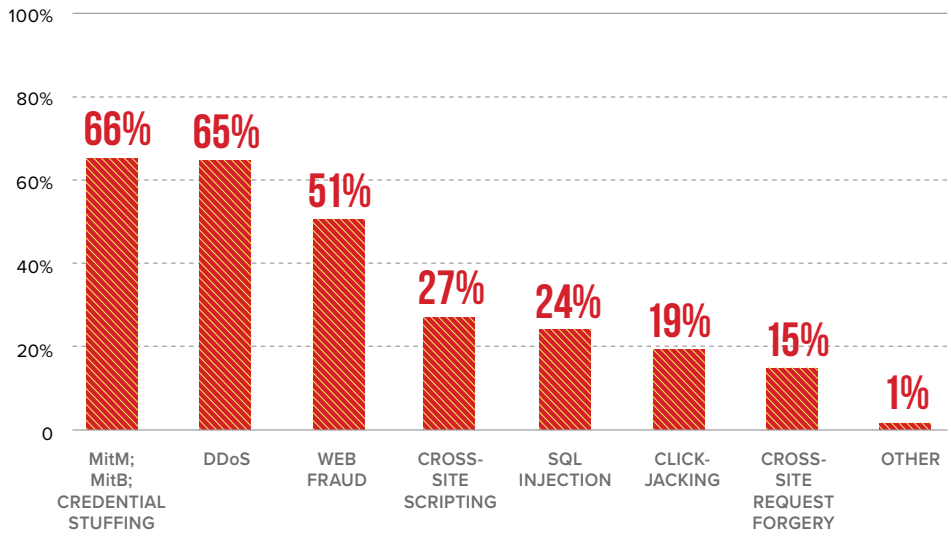
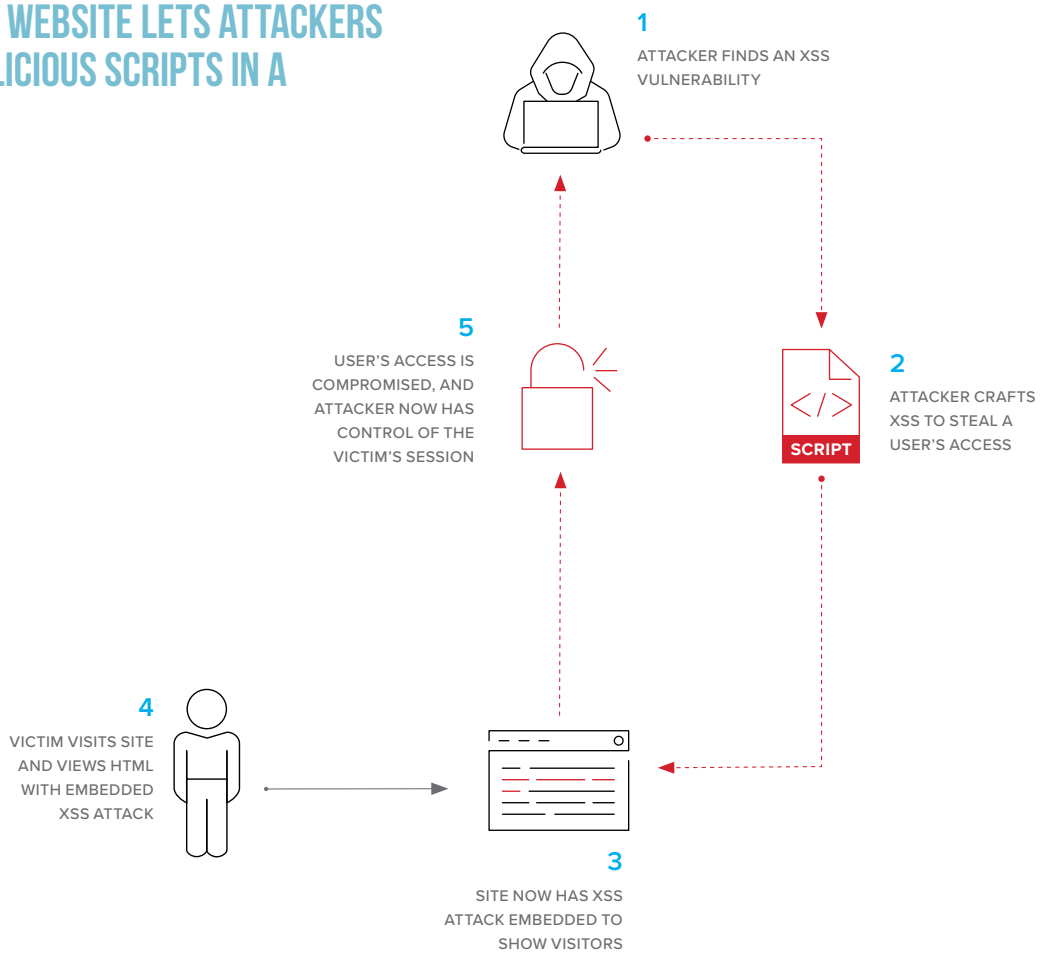


FIGURE 59: MOST DEVASTATING CYBER-ATTACKS (MULTIPLE ANSWERS ALLOWED)

FIGURE 60: XSS EXPLOIT PATH

CROSS-SITE SCRIPTING OCCURS WHEN A VULNERABILITY IN A WEBSITE LETS ATTACKERS RUN THEIR OWN MALICIOUS SCRIPTS IN A VICTIM'S BROWSER.



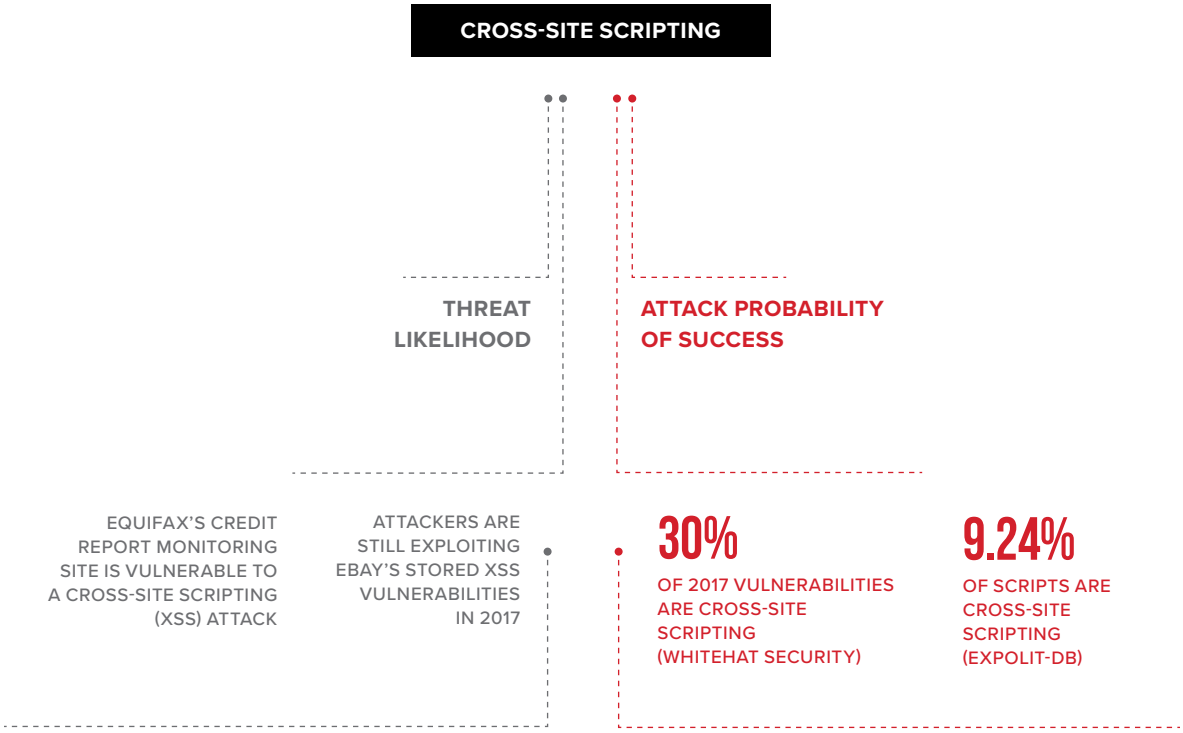
SCRIPTING ATTACKS TO HIJACK ACCESS

One of the most common client scripting attacks to hijack application access is cross-site scripting (XSS). XSS occurs when a vulnerability in a website lets attackers run their own malicious scripts in a victim's browser within the trusted context of the site they're visiting (see Figure 60). It's powerful because XSS runs against a website that the user trusts, so any commands and messages appear to be generated by that site. This attack also bypasses the same-origin defense mechanism in the browser, which prevents web pages from different origins from accessing data contained in a page.

Attackers can use XSS to steal session tokens or generate fake web pages to capture a user's credentials or other personal information. Some of the more sophisticated XSS attacks load key loggers onto the victim's computer to monitor their passwords as they type them in.

XSS can occur *anywhere* an external user can enter content to a website, which makes it one of the most common types of vulnerabilities. XSS can be hard to find and eliminate because it uses HTML commands that are often supported and required by a website to render its pages.

FIGURE 61: CROSS-SITE SCRIPTING ATTACK LIKELIHOOD



XSS IS POWERFUL BECAUSE IT RUNS AGAINST A WEBSITE THAT THE USER TRUSTS, SO ANY COMMANDS AND MESSAGES APPEAR TO BE GENERATED BY THAT SITE.

In 2017, two extremely large, high-traffic websites were found to have significant XSS vulnerabilities that attackers exploited:

- **EBAY:** Attackers are still exploiting XSS vulnerabilities that eBay has yet to fix.[56]
- **EQUIFAX:** As if the initial data breach weren't bad enough, Equifax's credit report monitoring site—the one that was meant to protect consumers who had already been victimized—is vulnerable to XSS attacks.[57]

These are just two prominent cases that made headlines. No doubt, there are many other XSS vulnerabilities currently spread across the Internet that are exploitable.

CROSS-SITE REQUEST FORGERY ATTACKS

Another scripting attack against app clients is the cross-site request forgery (CSRF), which hijacks an app client via a forged web request. If an attacker can get a user's client to submit a request without the user's knowledge or permission, the client will comply and send along the authentication. The result is that the user unwittingly carries out unwanted actions in the application they're currently using. This attack is possible because web browsers and app clients automatically submit the appropriate stored access authorization token with every web request. This attack requires that the victim be logged into the application in question so that the session token is valid at the time.

MALWARE ATTACKS AGAINST APP CLIENTS

Another way to attack an app is to seize control of the application client using malware. There are many ways malware can be injected into a client app (see Figures 62 and 63). It's commonly done via social engineering, phishing, or a Trojan horse application.

Another vector is infecting via a browser vulnerability with a drive-by-download, usually embedded in a website or web advertising. In this case, the user doesn't even have to click on anything to become infected. Once the malware is present, it can directly steal the application credentials and account information as the user enters them into the client.

Some malware, like the infamous TrickBot banking trojan,[58] uses web injection to modify the web pages shown to the user (for example, putting in additional fields that steal data as the user enters it).

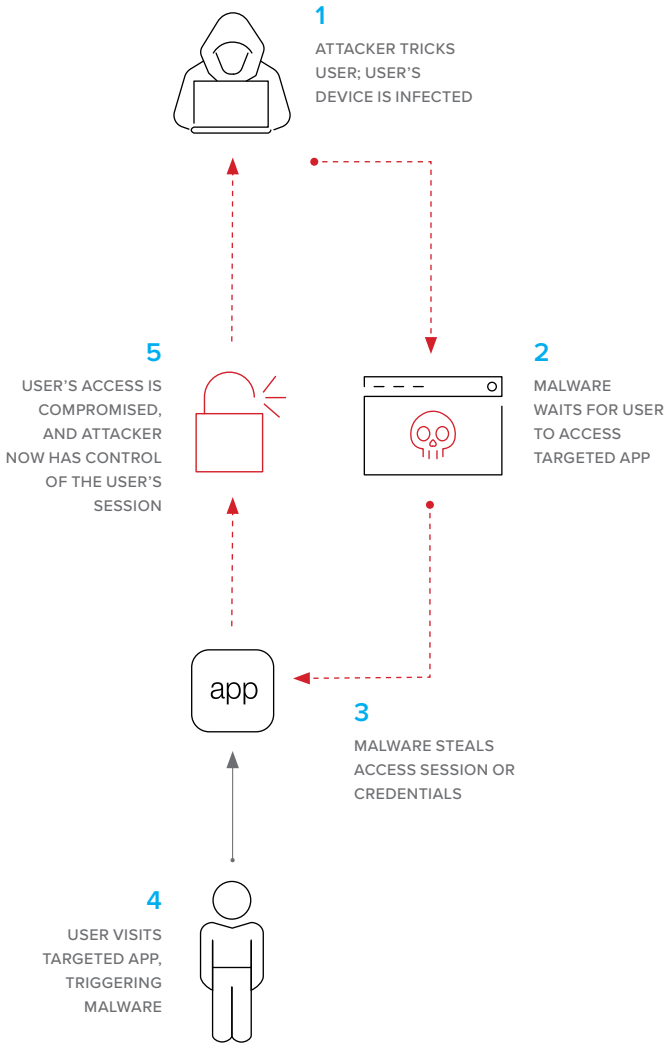


FIGURE 62: CLIENT MALWARE INFECTION ATTACK PATH

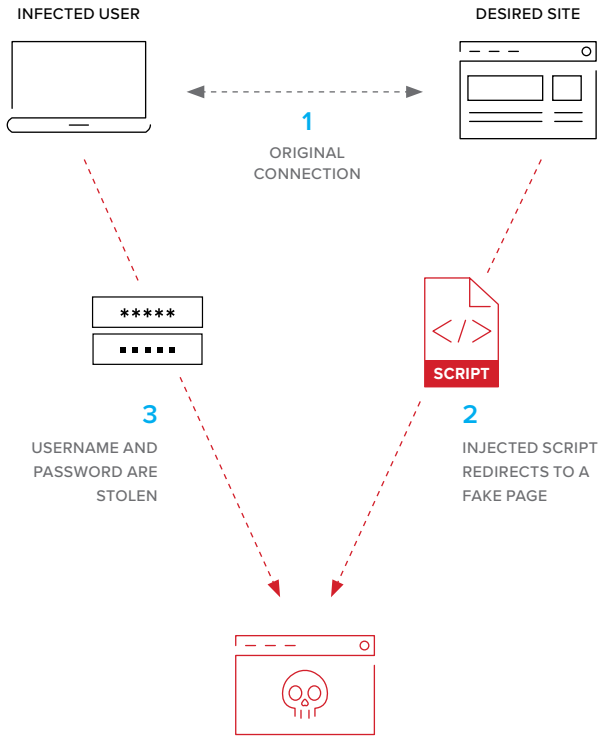


FIGURE 63: MALWARE MAN-IN-THE-MIDDLE ATTACK PATH

A common way for malware to sneak into an organization is within encrypted websites so that network intrusion detection systems can't see it. The F5 Ponemon security survey asked respondents how confident they were in their organization's ability to detect encrypted malware. The results speak to the magnitude of this problem.

It was disconcerting to learn that more than half (51%) of respondents were "not confident" or had "no confidence" at all in their ability to detect malware in encrypted traffic, while 15% were only "somewhat" confident. Just 34% said they were confident or very confident (see Figure 64).

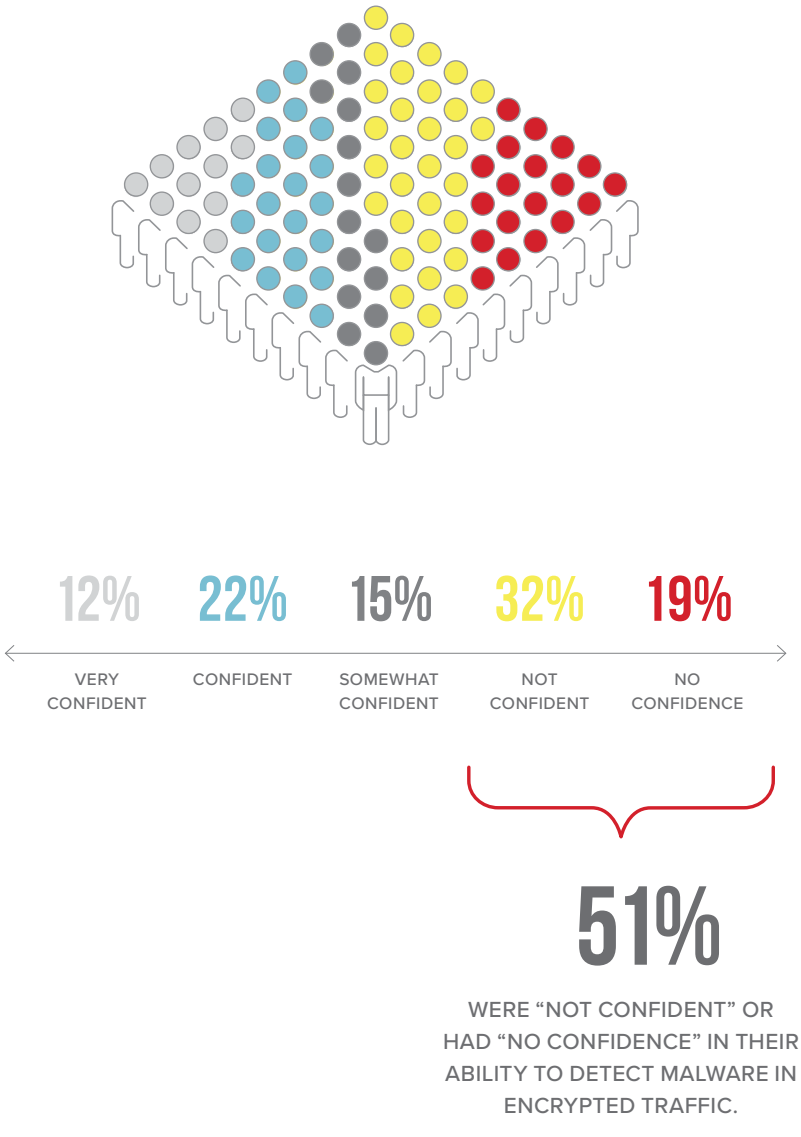


FIGURE 64: CONFIDENCE FACTOR IN ENCRYPTED TRAFFIC INSPECTION

04

PROTECTING APPLICATIONS

We know there are threats to applications, but what do we do to stop those threats? This section explores strategies for how to do just that. Before we dive into our recommendations, we want to share what our F5 Ponemon security survey respondents told us about their own organizations and how they're defending themselves against application threats.

HOW IS APPLICATION SECURITY MANAGED?

Having a single, clear owner that is responsible for the security of an app is crucial in order to make trade-offs regarding protection versus cost. Yet, in 90% of the cases, survey respondents said that application security is run outside of the security office. One wonders who is held responsible when an application is breached or suffers a DDoS attack.

Based on survey responses (see Figure 65), the primary barriers to strong application security are understanding what's going on within the application, whether that's due to lack of visibility, skilled personnel, or other factors. There are technological tools that can help this, such as a CASB for application discovery.

HOW ARE APPLICATION VULNERABILITIES HANDLED?

As we've seen in the threat sections of this report, applications have many tiers of exploitable vulnerabilities that attackers can use to crash it or break in. Organizations need to find and fix these vulnerabilities before attackers discover them, yet 46% of survey respondents disagreed or strongly disagreed that their organization had adequate resources to detect application vulnerabilities, 49% said the same about their ability to remediate them. Many organizations realize that just scanning for known vulnerabilities isn't enough. With customized and internally written applications, deeper (and more expensive) analysis, including penetration testing and code review, is needed.

Vulnerability mitigation is like weeding a garden—it should be done frequently and regularly so that new problems can be resolved before they are exploited. Based on survey responses (see Figure 66), adequate vulnerability management is still a challenge for many organizations.

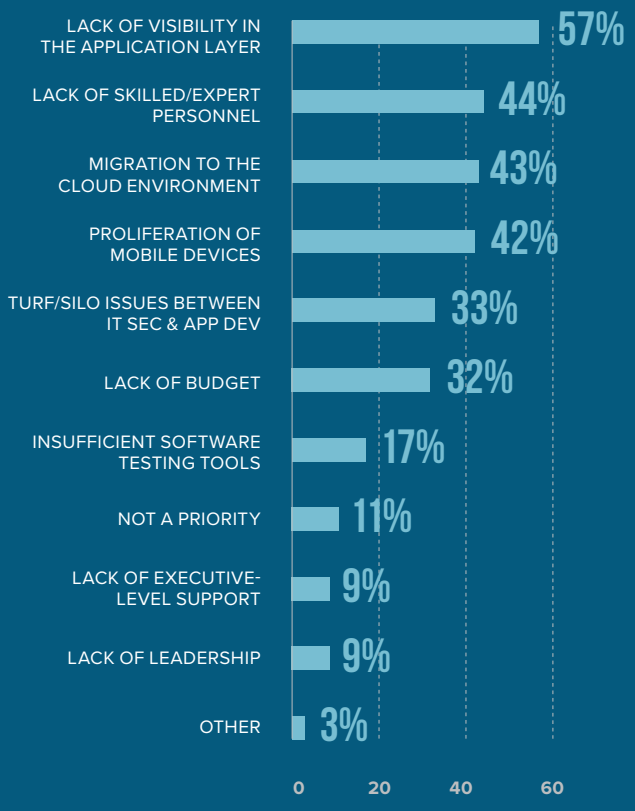


FIGURE 65: BARRIERS TO ACHIEVING A STRONG APPLICATION SECURITY POSTURE (3 ANSWERS ALLOWED)

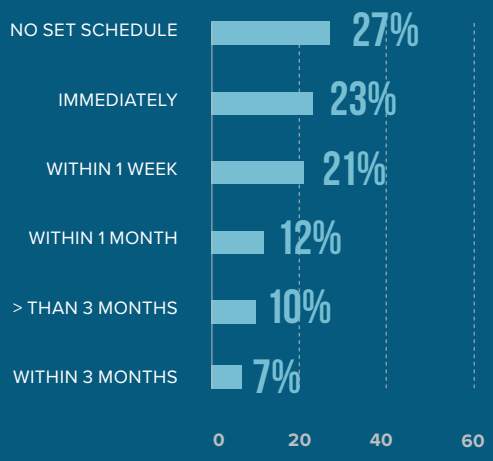


FIGURE 66: FREQUENCY OF WEB APPLICATION VULNERABILITY PATCHING

WHAT SECURITY CONTROLS ARE IN PLACE?

Beyond finding and patching holes, app defenders need to use security tools to blunt attacks. The primary tools that survey respondents rely on are web application firewall (26%), application scanning (20%), and penetration testing (19%). In addition to dedicated security tools, survey respondents said they use a number of operational techniques to help resist attacks, including segmentation (41%), Linux- and Windows-based containers (36%), and managed cloud-based application services.

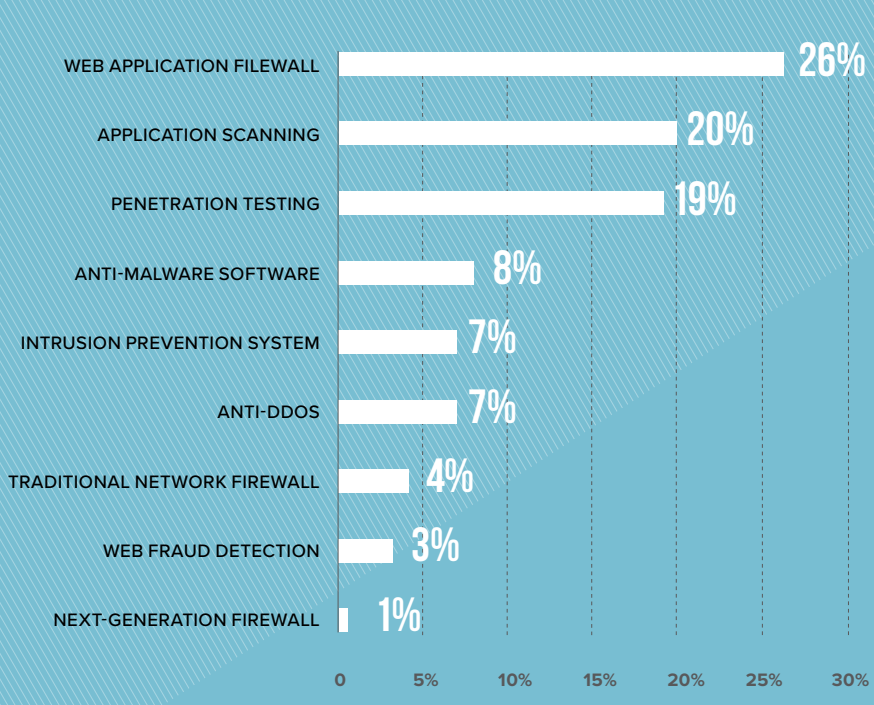
Implementing an application hardening practice before deploying applications in production is a great way to ensure you are launching secure applications, but only 36% of survey respondents stated they do this most of the time.

Because denial-of-service attacks and disasters are not uncommon, application defenders often need special tools and methods to keep their apps up and running. Comprehensive backups (68%) and strong DDoS defense (61%) are the primary controls organizations deploy for high availalbilty, followed by redundant apps (46%) and resiliance testing (43%).

Since injection attacks are common and attackers love to tamper with applications, defenders need to make sure code isn't being secretly altered. Most organizations deploy data checksums (69%) and audit change, access, and procedures logs (64%) to ensure the integrity of their applications.

Now that we know what organizations are doing to defend themselves, let's look at what F5 recommends based on threat intelligence.

FIGURE 67: APP SECURITY CONTROLS IN USE



THE TOP 3 SECURITY TOOLS THAT RESPONDENTS RELY ON ARE WAFS, APP SCANNING, AND PENETRATION TESTING.

AN APPLICATION DEFENSE STRATEGY

The question on every defender's mind is how likely am I to be hacked? It's one thing to be singled out for attack because your organization or application has been specifically targeted. For everyone else, it's really about how attractive an opportunity you present to a cybercriminal.

We know that attackers are constantly sweeping the Internet looking for weak spots they can jam their fingers into. Some use dedicated search engines, others just spider the entire web or IP space. This means that the bigger your Internet footprint (that is, the more applications you have online), the more attention you will get from attackers.

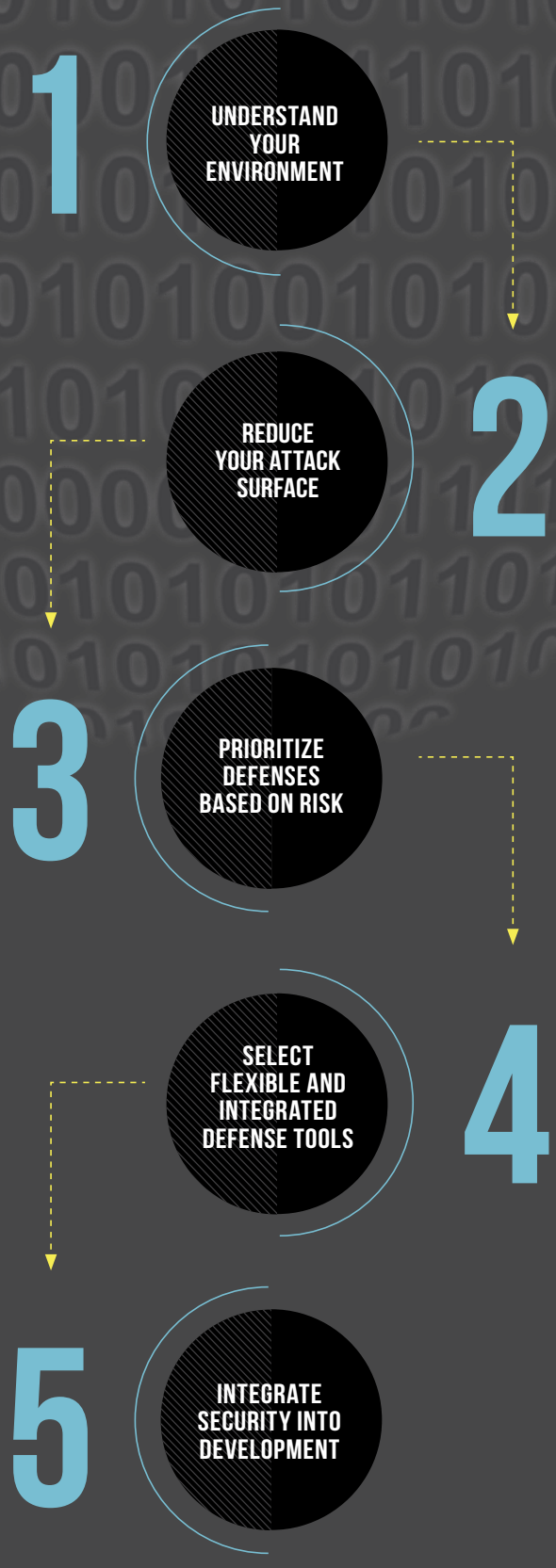
In general, there are two types of attackers: the opportunists and the attackers who target you. Both care about return on investment (ROI). They will avoid the hard way and try to go after the low-hanging fruit first. But their goals and methods are different.

Opportunist attackers keep their ROI high by keeping costs low. They use a spray-and-pray approach to sweep the Internet looking for easy pickings. It's like how the jackal picks off the weaklings in the herd. It's not personal; it's just that you were too slow that day, so you got eaten. These attackers come at you with canned exploits and known proven methods that will make them a quick buck. If rebuffed, they quickly move on to the next target. The downside is that there are a lot of them out there. If you're not seeing opportunist attacks poking at your applications every few minutes, there's something wrong with your Internet connection.

The targeted attacker picks their targets carefully. Their goal could be espionage or a high payoff, but once you're in their sights, they're likely coming after you. They maintain their ROI by going after high-value assets, which means they are willing to put in more time and effort to break in. In fact, they will persist in their attacks (which is why they're called advanced persistent threats), because once they've invested in the attack, they're unlikely to walk away after being thwarted a few times. These are the

THERE ARE TWO TYPES OF ATTACKERS: THE OPPORTUNISTS AND THE ATTACKERS WHO TARGET YOU. BOTH CARE ABOUT RETURN ON INVESTMENT.

FIGURE 68: PRIMARY APPLICATION PROTECTION STEPS



attackers that you block with some technique who will modify their tactics and overcome these defenses soon after. They're not unstoppable, but they are not easy to discourage. The good news is that they're pretty uncommon and usually only go after the big scores that will achieve their goals (usually money).

To be meaningful, your defense strategy must absolutely cover opportunist threat actors. They're as ubiquitous as rain in Seattle and if you don't plug every leak, you will get wet. Beyond that, you should have some kind of strategy for dealing with the targeted attacks. Sometimes targeted attacks are personal. Maybe one of your employees angered an elite hacker on some gaming forum. Or perhaps one of your customers is a critical government supplier and a nation-state is looking to infect the supply chain.

If nothing else, your targeted attacker defense strategy should include detection and mitigation—if you can't stop them, then at least know when they get in and how to clean things up after.

1 UNDERSTAND YOUR ENVIRONMENT

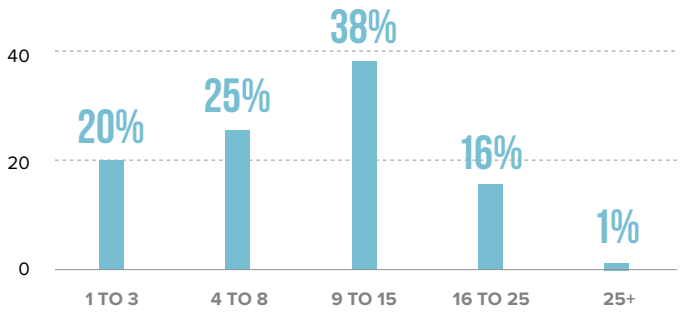
What on your network will your attackers want to steal or corrupt? First things first, you need to know what you have on your network. Unfortunately, this is not an easy job considering that the number-one cited difficulty in app protection in our F5 Ponemon security survey was "lack of visibility in the application layer (57%)." But how do you gain that visibility?

You can begin chasing down your application usage by drawing out critical business and information flows within your organization. These flows will naturally touch application and important data repositories. As mentioned earlier, vulnerability scans of your Internet presence are common tools to probe for weaknesses. They are also a great way to get an up-to-date picture of what applications are visible to the outside world.

DEVELOPMENT

If your organization develops or modifies applications (and most do), you need to interview the development and IT teams to get an idea of what programming tools and environments are being used for your applications. Tracking down this information in a dynamic organization is a challenge, often made worse by unplanned dabbling in new development tools and frameworks. In the F5 Ponemon security survey, we tried to get an idea of what the average organization is doing, beginning with an idea of how diverse a set of application frameworks organizations need to manage. The more frameworks and environments in place, the more resources, trained personnel, and scanning tools are needed to secure them (see Figure 69). Anything more than a small handful of web application frameworks or environments means a wider target profile for attackers as well as increased load on patching and security evaluation. Wise CISOs would keep a sharp eye on the growth and maturity of the app development platforms in use.

FIGURE 69: NUMBER OF WEB APPLICATION FRAMEWORKS IN USE



EXTERNAL APPLICATIONS

To chase down applications outside of your organization, a tool like a cloud access security broker (CASB) can be very helpful. CASBs sit between your users and the Internet, monitoring and reporting on all application activity. They can not only tell you what the top applications your employees use (and how they access them) but also give insight into shadow IT application usage.

Once you've got a good idea of what applications are in use and which are visible to the outside world (probably all of them), you need score them based on significance to your organization's operations. Remember step 3 is to prioritize your defenses based on risk. To do risk analysis, you need to know what's important and what's not. This should not be done in a vacuum, either. Talk to your leadership about what is truly mission-critical. It may not be what you think.

Lastly, this is not a one-and-done exercise. It may not even be enough to do it annually. It's a constant process that involves keeping an eye on what applications and data repositories are in play, monitoring what users need to do, and evaluating how your development environments are evolving.

2 REDUCE YOUR ATTACK SURFACE

Now that you have a complete list of applications in use, it's time to do a first pass of securing them. We'll look first at the opportunistic attackers, which means closing up all the obvious holes and known attack paths. All external application services are targets, either to be used against your organization or leveraged for reflection attacks against others.

It's not always easy to even keep up with basic hardening and locking down. A large, modern enterprise is going to have many complications with patching, like compatibility, support, code obsolescence, library dependence, quality assurance testing, vendor versioning, lack of operational resources, and compliance change windows.

Again, there are security tools that can help with this. A good web application firewall (survey respondents' #1 tool of choice) has the ability to buy you time to patch. It does this with "virtual patching" by linking to a static or dynamic application security test (SAST/DAST) tool that automatically creates new signatures for discovered holes. In addition, a good WAF analyzes application traffic to detect and block known exploit attacks.

FIGURE 70: WEB APPLICATION FIREWALL PROTECTION

A GOOD WAF CAN AUTOMATICALLY UPDATE ITS SIGNATURES FROM THREAT INTELLIGENCE FEEDS.

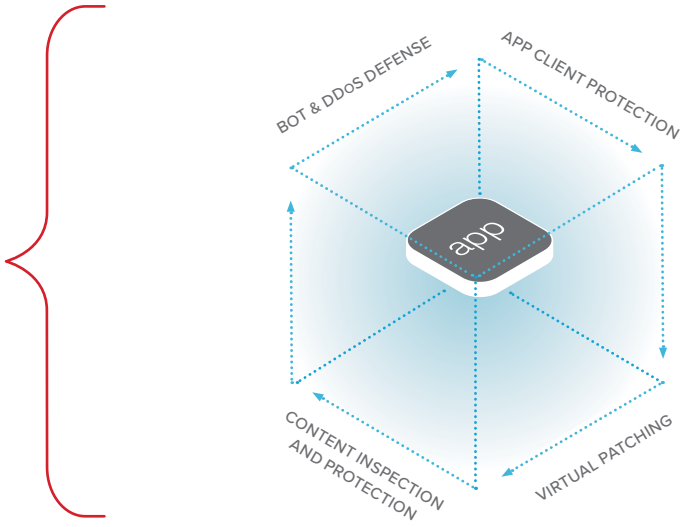
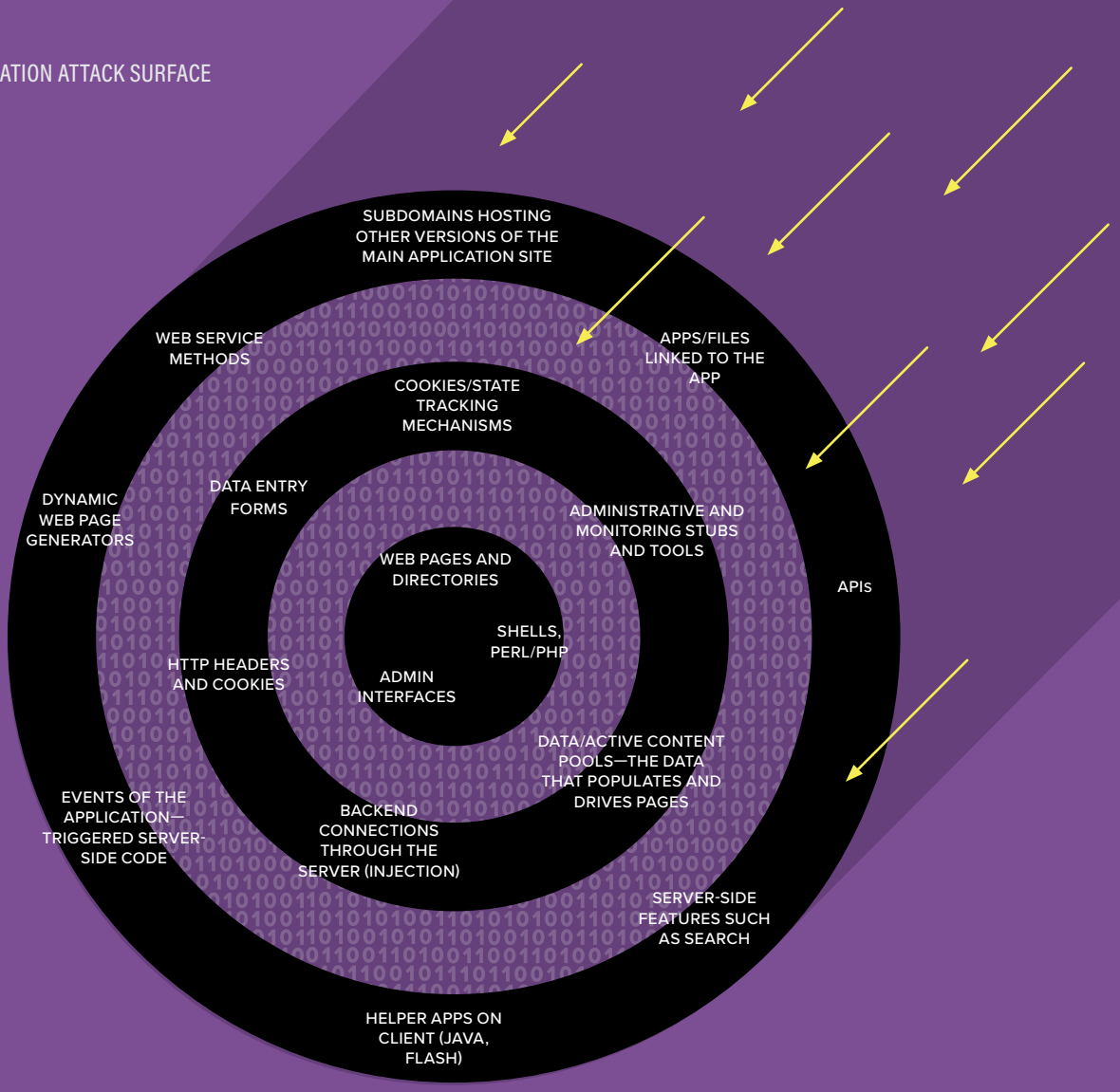


FIGURE 71: THE APPLICATION ATTACK SURFACE EXISTS AT EVERY TIER



The WAF can automatically update its signatures from threat intelligence feeds and vulnerability scans of your environment to ensure you don't have to scramble when a new exploit is released. This alleviates the time pressure of having to patch immediately and gives the ops team time to properly test and roll out fixes. This kind of WAF feature is excellent at protecting against the obvious attacks that opportunistic attackers throw at you. Just remember, your entire Internet footprint is the target, so these kinds of solutions need to be deployed across the board. We have heard of numerous preventable security incidents that occurred because security teams had not enabled the necessary security blocking features on their WAF perimeter.

Expose as little as possible

Because attackers will strike at anything you put on the Internet, it's a good strategy to only expose the bare minimum online. Everything else should live behind tight access controls and firewalls. We are not naïve enough to believe this is an easy task in a world beyond network perimeters[59]—but that doesn't mean you shouldn't try.

The likelihood of a given application or application component being attacked is key to understanding the risk of getting breached. Furthermore, not all exposed resources have equal vulnerability and impact in terms of attack potential. Overall, this is called an "attack surface" and it represents the sum total of all the listening services and methods of operation available to external users. All things being equal, app developers will introduce additional libraries and services to an application, causing it to bloat and increase its footprint. A majority of these services lie within application services all up and down the tiers, as shown in Figure 71.

SEGREGATE AND PARTITION

You've got your inventory and know what's important; now, it's a good idea to set up some internal barriers between asset groups. It's one thing to have a low-priority application get breached, but it's another if the attacker can use that as a conduit to reach the higher priority systems.

This segregation can also exist within the code of your internally developed applications. As a rule, you should reduce the amount of application functions exposed to untrusted systems and partition those functions from the rest of the systems. This is especially true for powerful interfaces like APIs and databases, which often provide full pathways into applications and data. These functions should be access controlled with the least-privilege principle applied. Any code or function that if compromised will provide unfettered access to the entire application should be insulated and monitored carefully. This can be done in code, with server isolation, sandboxes, lower privileged users, and even with firewalls.

3 PRIORITIZE DEFENSES BASED ON RISK

Having your apps hacked or disrupted is costly but you need to balance that against the operational and capital costs of defense. This is where risk comes in. The point of doing risk mitigation is to avoid unnecessary costs and at the same time make the best use of available resources. Risk analysis doesn't have to be perfect; it just has to be better than randomly choosing controls based on arbitrary "best practice" lists[60] or worrying about the headline threat of the day.[61] Your risk analysis should be based on what you know the attackers are going after and what is important to your organization. To be truly effective and efficient, you align your best defenses to reduce those biggest risks.

To do this, you drive your risk strategy with data. Figuring out what attackers would go after is a key part of your risk analysis. This report, as well as the ongoing threat research available at F5 Labs, should give you plenty to chew on as to what attackers are doing. Since every organization is different, you need to be aware of what is attractive about your attack surface.

KNOW THE RISK OF YOUR CODE

If you develop applications internally, you absolutely need to know the exploitability of those applications. Since each new application is unique to your organization, it means it's never been security tested before. There are a variety of ways to test, including internal scanners and code reviews. The OWASP Dependency-Check utility[62] is great way to see what obvious flaws might be lurking in your code from the use of external libraries. Bringing in a third party to do testing can be very revealing as you get an independent and knowledgeable perspective. With this information in hand, you can properly assess the risk of any internally developed applications. If your application is used by your customers, then the imperative to do this is even stronger as you have an obligation to protect their data as well as your own to preserve your business.

4 SELECT FLEXIBLE AND INTEGRATED DEFENSE TOOLS

To do an acceptable job of securing your applications, you don't need dozens and dozens of technical control solutions. Beyond the fixed cost of the solutions, there is the operational cost of deploying, running, and maintaining the controls.

What you need is to focus on your risks and cover controls for prevention, detection, and recovery from attacks. This can be done by deploying a handful of flexible and powerful solutions that your team can use for existing as well as emerging threats. We've already mentioned three common key technical controls: a web application firewall, a vulnerability scanning solution, and a CASB.

THREE COMMON TECHNICAL CONTROLS CAN HELP YOU FOCUS ON RISK, PREVENTION, DETECTION, AND RECOVERY.

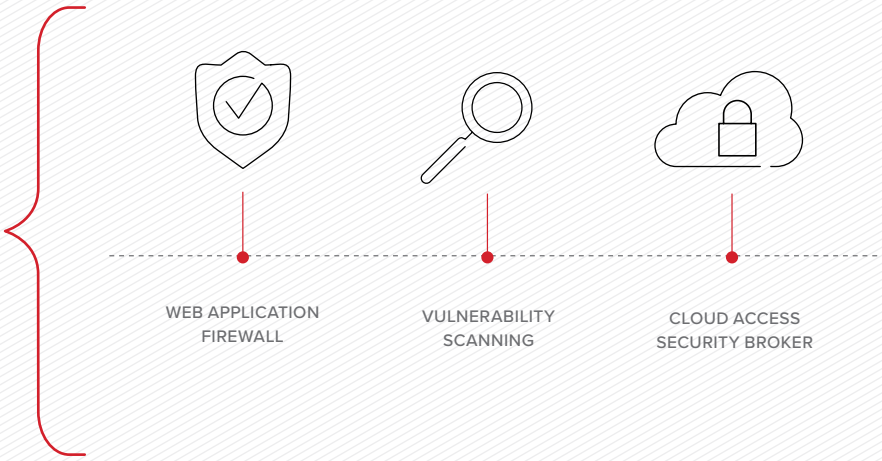


FIGURE 72: TECHNICAL CONTROLS THAT COVER PREVENTION, DETECTION, AND RECOVERY.

Provided they're powerful and flexible enough, these three tools can help with network access control, vulnerability management, inventory, risk analysis, and wrangling authentication.

There are also tools needed for the threats of DDoS, app clients, DNS, and network transport. We discuss these in more detail later in this section.

5 INTEGRATE SECURITY INTO DEVELOPMENT

It's more efficient to avoid creating security vulnerabilities in the application to begin with instead of trying to back port a fix to a newly discovered problem. To do this, you need to work with the development team. A strong first step is to make them aware of the problem by teaching everyone involved in the production of web and mobile apps about the OWASP Top 10 (see Figure 73). By having a good working knowledge of how threats and app exploits work, developers will be empowered to secure apps without the security team needing to remind them. This not only reduces holes in the final product, it also helps find and speed fixes of new holes. The F5 Ponemon security survey found that 66% of respondents felt their organization's security posture is negatively impacted by a shortage of skilled and/or qualified application developers.

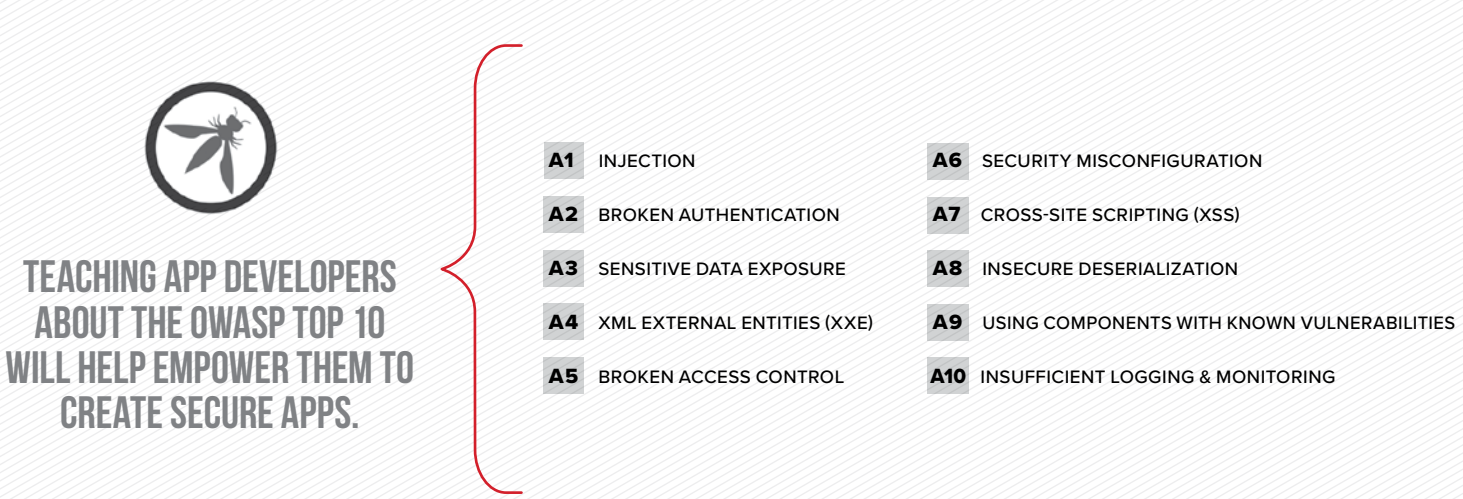


FIGURE 73: THE OWASP TOP 10

Another powerful development security practice is to ensure that confidential application data is encrypted at rest. Applications are in constant danger of being breached and the data stolen, so critical data fields need to be rendered unreadable without possession of the proper encryption key. It also is important to protect that encryption decoding key. If an application is taken over by an attacker, you don't want to make it easy for them to copy the key and decode the stolen data. There are some powerful suggestions on how to manage this in the OWASP Cryptographic Storage Cheat Sheet.[63]

PROTECTING DOMAIN NAME SYSTEM SERVICES

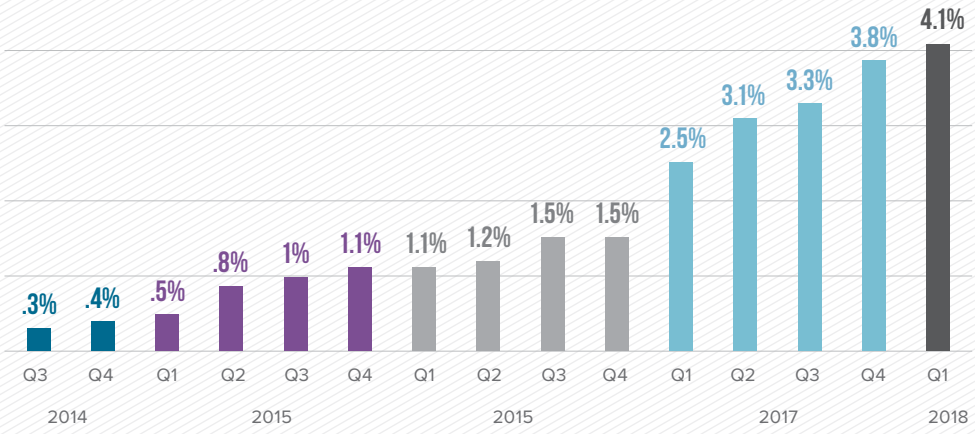
Like any critical application infrastructure service, DNS servers should be protected with access control and made highly available. Access control can take form of hardening, patching, and firewalls. Note that firewalls should not just block all ports except 53 but should also block DNS-specific exploits and DDoS attacks. DNS servers can be made highly available with redundancy and close monitoring.

PROTECTING THE TRANSPORT LAYER

Over half the web sites on the Internet are using encryption, but is it sufficient encryption? To ensure strong encryption at the transport layer, the first step is to keep abreast of the current acceptable standards and monitor your deviation from those standards.

To reduce the chance of MitM attacks on your applications, your web servers should use the HTTP security header called HTTP Strict Transport Security (HSTS). However, according to the F5 2017 TLS Telemetry report, HSTS is not widely deployed at the server level (10%).

FIGURE 74: PREVALENCE OF HTTP STRICT TRANSPORT SECURITY (HSTS)



The leading solution to address the threat of the theft of domain names or the improper issuance of certificates is certificate transparency. Certificate pinning embeds the server certificate into the client (a browser or a mobile client) such that the client then requires *that exact* certificate with serial numbers. There is a Certificate Transparency (CT) system that uses searchable certificate repositories and requires certificate authorities to publish every certificate they issue. Site owners can then search the repositories periodically to see what certificates have been issued for their site.

Try it now here: at https://crt.sh/ or https://censys.io/. The CT system has already paid some dividends, at least for Google, as it was CT that identified the improperly issued Symantec Google certificates.

PROTECTING AGAINST DDOS

For many apps, there is a certain naiveté in their design that no one will attack the site or its components. However, any app or site on the Internet is not only targetable for attack but is also being scrutinized as a possible DDoS weapon against others. Therefore, the days of brittle infrastructure shielded only by obscurity are long gone. Critical components need to be hardened, access-controlled, and capable of resilient failover.

Organizations need to assume that a DDoS attack is in the future and do the appropriate risk analysis. Based on potential impacts, applications should be protected from DDoS attacks at the network layer, application layer, and supporting infrastructure. There are many levels of anti-DDoS solutions, from on-premises scrubbing equipment to hosted solutions. The important thing is to size the solution based on the risk to your applications and the likely threats.

PROTECTING YOUR APP CLIENTS

When it comes to protecting app clients, it's helpful to think of two classes of users: your customers who access the apps you publish, and your organization's internal users who access apps on the Internet. We'll begin by talking about latter: your users.

The first step in protecting your users is by implementing reliable access control. Passwords have always been the go-to solution for authentication. They're cheap and easy—and completely weak. As we noted before, the F5 Ponemon security survey found that 74% of organizations still use a username/password unique to the application for authentication. For any application of significance, organizations are starting to turn to stronger authentication solutions such as federated identity or multifactor authentication.

74% OF ORGANIZATIONS STILL USE A USERNAME/PASSWORD UNIQUE TO THE APPLICATION FOR AUTHENTICATION.

Unfortunately, some of the applications your users may be accessing may only support username/password authentication, leaving your users open to access attacks like password guessing or stolen credentials. Technical solutions like the aforementioned CASB are a great place to start. Not only does a CASB give you a clear idea of what apps your users are using, but its primary purpose is to manage that access control by consolidating and augmenting authentication for external apps.

In addition, a CASB can help classify and filter data being uploaded into external applications from your organization to ensure that regulated or confidential data isn't being stored somewhere it shouldn't. A CASB can also lock down the client itself by ensuring that only authorized (and hardened) clients are allowed to access external applications.

PROTECTING YOUR CUSTOMERS' APP CLIENTS

Just as you protect your users, you should also be working to protect your customers' app client sessions. They are just as likely to suffer from client attacks like stuffing of stolen credentials, phishing, or malware attacks. To do this, some of the more powerful and flexible WAF systems provide app client protection which can detect bot attacks, brute-forcing, and logins from suspicious locations. This simple validation is great way to add another layer of protection for your customers as they access your apps.

THE FIRST STEP IN PROTECTING YOUR USERS IS IMPLEMENTING RELIABLE ACCESS CONTROL.

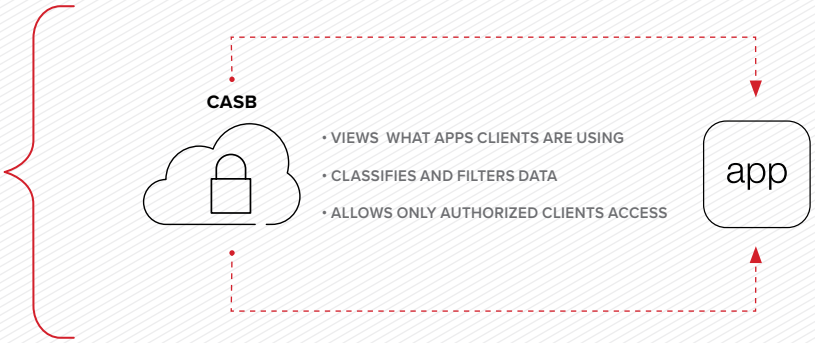


FIGURE 75: CLOUD ACCESS SECURITY BROKER FEATURES

You want to protect those client sessions with transport layer encryption as well. Having good transport layer encryption on web applications is critical not only for privacy but also to prevent MitM attacks from interfering with the transmission. To aid in verifying the authenticity of your applications, consider using extended validation (EV) certificates[64] to provide tighter assurance that your organization is a verified legal entity for the site. EVs may be a bit costlier, but for an application you're offering to customers, they're more than worth the extra cost for the trustworthiness they offer. Most modern app clients support EVs and will provide the user with notification when they are used.

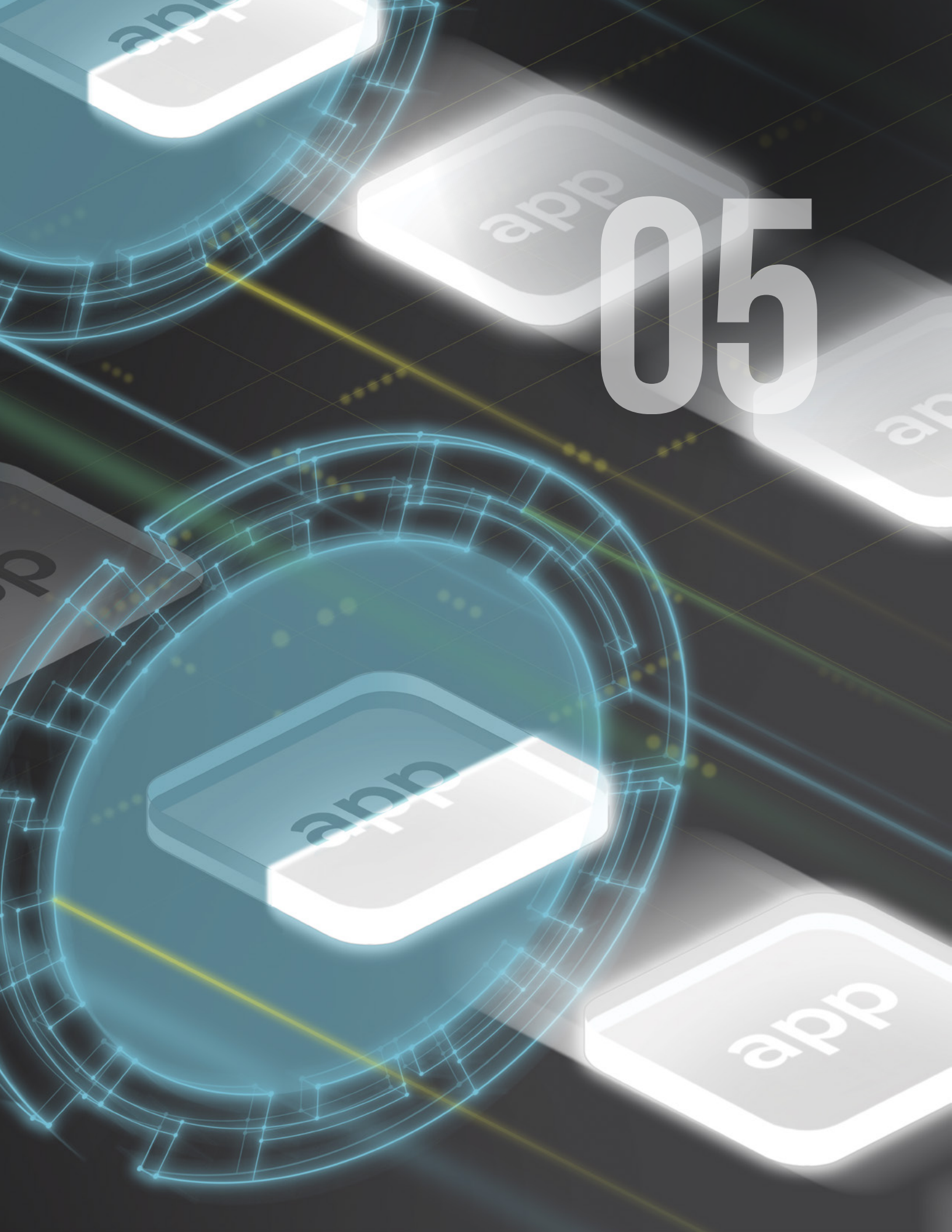
You can also enhance your transport layer protection by making sure session cookies set HTTPOnly,[65] domain restricted, and the X-frame-options to DENY. This helps prevent click-jacking and credential theft. It's an easy tweak on the web server or gateway settings to ensure this is in place for an app.

OVERVIEW OF ATTACK TYPES AND DEFENSE TOOLS

Below is a table matching defensive tools with the known application attacks they can help mitigate.

ATTACK TYPE	AFFECTED TIER	WAF	VULNERABILITY SCANNING	ANTI-DDOS	MFA	APP DEV SECURITY	DATA-AT-REST ENCRYPTION TRAINING
Injection	App services	✓	✓			✓	✓
Deserialization attacks	App services	✓	✓			✓	✓
Abuse of functionality	App services					✓	✓
API attacks	Access	✓	✓		✓	✓	✓
Account access attacks	Access	✓	✓		✓	✓	✓
TLS attacks	TLS, Network	✓		✓			✓
DNS hijacks	DNS	✓					
Hybrid DDoS	All app tiers	✓		✓			
Reflection DDoS	All app tiers	✓		✓			
TLS denial-of-service	TLS			✓			
Cross-site scripting attacks to hijack access	Client	✓	✓		✓	✓	
Malware attacks against app clients	Client				✓		

TABLE 1: MAPPING ATTACKS TO SOLUTIONS


05
app
app
app
app

THE FUTURE OF APP PROTECTION

We have examined the contemporary threat landscape for applications, but things are always evolving in the technology world. In fact, some of the changes that are coming are not just evolutionary but revolutionary. With these changes come changes to the threats and dangers to our applications. Therefore, we need to be ready with the appropriate responses to keep up.

One future pattern that we can absolutely count on is our continued growth and dependence on applications. Apps are here to stay and we need them more than ever.

APPLICATION SECURITY

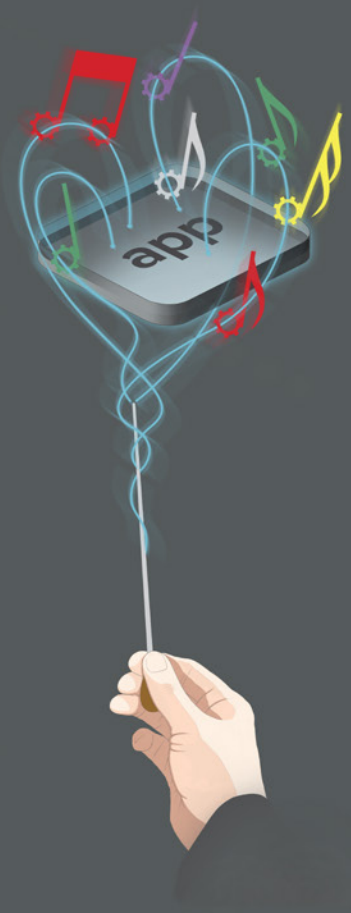
As we've pointed out throughout this report, applications are no longer monolithic software programs but instead are colony creatures composed of differing scripts, libraries, services, and devices. In the future, we hope to see our security tools catch up to this paradigm.

Expect to see application security embrace this revolution with software components and frameworks that are robust enough to stand alone against adversity. No longer will we expect APIs or code libraries to be accessed, controlled, and filtered. Instead, they will have much more security baked in straight out of the box. In the future, developers will have a broader spectrum of secure components and frameworks to choose from in a way that is dramatically different and better than today's fragile and overly dependent ecosystem.

In addition to these trends, we see changes on the horizon in serverless computing, outsourcing of application security, and inevitable improvements in TLS security.

KEY TRENDS TO WATCH FOR IN THE FUTURE OF APPLICATION PROTECTION

- Application frameworks that are "secure by default"
- Application components with the capability to report security status and events in a standardized format
- Security scanners that can continuously test application components in production in a safe but useful manner



SERVERLESS COMPUTING AND APPLICATIONS

Serverless computing is an emerging computing model that enables developers to build web applications without having to worry about the servers the applications will run on. Although serverless can be a confusing term—of course, applications still run on servers—the concept is part of a continuing evolution to help developers build just what is necessary to fulfill a business need.

Serverless applications usually focus on the user interface with events triggering functions that are otherwise sitting dormant within cloud platforms. The application code and scripts connect to APIs that directly trigger the functions and services, instead of the traditional (opposite) approach where the code runs within a server and calls other servers. The advantage of serverless development is that it enables faster, more streamlined focus on providing an app solution. Also, because serverless development decouples the app from the traditional app stack of servers and systems, it is more flexible and scalable.

However, there are downsides, too. Serverless applications do not eliminate all app security problems. In fact, a serverless application has a wider footprint of more services and functions, which means there is potentially a larger attack surface to be exploited or disrupted. With more exposed functions, the threat of abuse-of-functionality attacks also increases. And with the dependence on function calls to APIs, there is a greater need to ensure access control and transport encryption. Serverless apps do not eliminate problems like XSS, CSRF, and injection. Lastly, inventory and monitoring can become more difficult as apps become more dispersed among isolated and diverse systems.

KEY THREATS TO WATCH FOR IN SERVERLESS APPLICATIONS

- Access control attacks, especially against serverless APIs
- DDoS attacks against the much larger and dispersed, dependent infrastructure
- Core application exploits such as XSS, CSRF, and injection attacks
- Transport layer attacks against a much larger connecting network mesh

OUTSOURCING MORE OF APPLICATION SECURITY

In the F5 Ponemon *The Evolving Role of CISOs and their Importance to the Business*[66] report, 58% of CISOs reported outsourcing some of their IT security operations. In our recent F5 Ponemon security survey, 44% of respondents stated that "lack of skilled or expert personnel" was a main barrier to achieving a strong application security posture.

The same survey found that the average organization uses 765 different web applications and on average, 34% of them were considered critical. Combining these trends with the growing tendency for application attacks, we would expect to see more organizations outsourcing application security. This could mean either outsourcing application security functions, such as anti-DDoS or web application security monitoring, or even moving to hosted platforms that provide such outsourced services as part of their offering.

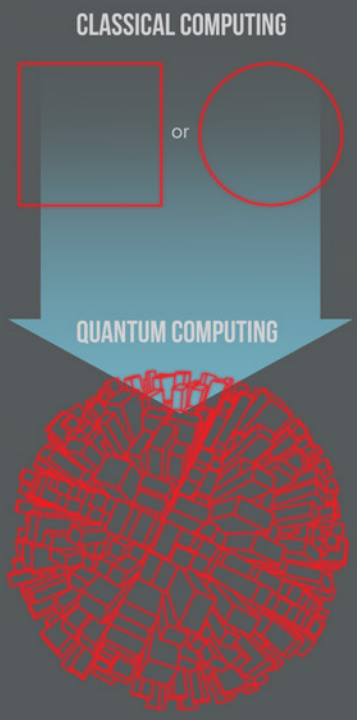
Unlike most typical organizations, outsourcing security organizations have the resources to invest in powerful security tools and highly trained security staff to operate them, 24x7x365. Since their very reason for existing is to provide security services to customers, their defensive capabilities are tied directly to customer satisfaction and revenue. This makes security improvements and compliance certifications a slam dunk for the business, as opposed to the traditional resistance that security teams encounter.

Furthermore, by managing security for many organizations, outsourcing security companies have superior threat intelligence and feedback on what controls are most effective. These organizations can spot emerging threat trends just as they begin to gain momentum and menace the larger Internet population. This is a big reason why F5 is in a position to provide timely and detailed reports like this one.

KEY POINTS TO CONSIDER WHEN OUTSOURCING SECURITY

- Your organization's security requirements versus your in house capabilities and dedication to security
- Specific application security needs that may not be feasible to obtain in-house
- The outsourcing security company's focus on and experience in providing application security related to your specific needs

FUTURE CHALLENGES FOR TRANSPORT LAYER SECURITY



In the short to medium term, the cryptographic community will struggle with the adoption of the latest version of TLS, 1.3, a more secure and faster encryption protocol. Nearly everything about the protocol has changed, including the following:

- Only forward secret ciphers are allowed. This means many more elliptic curve and Diffie-Hellman key exchanges.
- Removal of session-based server caches.
- Zero round trip (fast start) TLS, where the client sends data with the initial TLS ClientHello.
- Removal of previously stable TLS messages and fields such as *ChangeCipherSpec*.

As a result of all these changes (some of which are cosmetic), TLS 1.3 has had difficulty finding purchase in the Internet so far. Some changes are being made, such as re-introducing the fields that were removed. Complicating the move to TLS 1.3 is the fact that TLS 1.2 is completely sufficient as it stands. There's nothing really wrong with TLS 1.2, and there are no known protocol vulnerabilities against it. Therefore, there's no broad urgency to move to TLS 1.3, so the designers have some time to get it right.

The specter of quantum computing[67] is hanging over Transport Layer Security in the long term. If a real, working, quantum computer can be built with approximately 4,000 qubits, then every TLS connection in the world could be broken. Even those that are using elliptic curves and/or Diffie-Hellman key exchanges. That is a big "if" though. Current, true, quantum computers can only string together a handful of qubits. It is our opinion that Transport Layer Security will suffer more severe shocks from a different angle than quantum computing, but no one knows what those shocks will be yet.

KEY TRENDS TO WATCH FOR IN TRANSPORT LAYER SECURITY

- Browser support for TLS 1.3[68]
- Major compliance standards regarding network encryption[69]
- Advances in quantum computing[70]

CONCLUSIONS AND MORE QUESTIONS

After reading all of this, you might get the impression that web application security is intimidating. Perhaps, but there are some easy ways to get started. If you're in the minority and aren't using a web application firewall (WAF) yet, that would be a good first solution to investigate. Be sure to get proper training and make use of all the WAF features to help protect your applications. To quote our own guide to WAF configuration: "start small, but most of all, just start."[71]

Our goal in this report is to give you the information you need to answer the question, "What is the most important thing we can do now to protect our applications?" We have tried to anticipate and answer the kinds of questions CISOs would have regarding their application security. We hope you can use this report to fill in the gaps in your application security strategy—and that it will fuel productive conversations about risk and defense.

What's next? F5 Labs will continue to investigate threats and risks to applications and provide that information to you. Stay tuned.

APPENDIX

LITERATURE REVIEW

We conducted an extensive review of what has been previously published in the field of application security to help shape and define our report. Our work here is deeply indebted to the following giants whose shoulders we stood upon:

Open Web Application Security Project (OWASP) was a major source of information and inspiration for this report. There were numerous invaluable resources within the OWASP projects that provided very illuminating ideas for our work, including the following:

- Dr. Dan Geer, Application Security Matters, OWASP AppSecDC 2012 keynote
  http://geer.tinho.net/geer.owasp.4iv12.txt
- OWASP Testing Guide v4
  https://www.owasp.org/index.php/OWASP_Testing_Project
- OWASP Automated Threat Handbook Web Applications
  https://www.owasp.org/index.php/OWASP_Automated_Threats_to_Web_Applications

European Union Agency for Network and Information Security (ENISA) Threat Landscape 2016
https://www.enisa.europa.eu/news/enisa-news/enisa-threat-landscape-2016-report-cyber-threats-becoming-top-priority

Web Application Security Consortium: seminal work on the Web Security Glossary
http://www.webappsec.org/projects/glossary/

Contrast Security's State of Application Security: Libraries & Software Composition Analysis
https://www.contrastsecurity.com/state-of-application-security-libraries

Kenna Security: What You Miss When You Rely on CVSS Scores, Michael Roytman
https://blog.kennasecurity.com/2015/02/miss-when-rely-on-cvss-scores/

Advisen and Zurich, 2017 Information Security and Cyber Risk Management Survey
https://www.advisenltd.com/2017/10/25/2017-information-security-cyber-risk-management-survey/

Quantifying the Attack Surface of a Web Application, Keller & Turpe
http://www.feu.de/imperia/md/content/fakultaetfuermathematikundinformatik/pv/97-08/sicherheit2010_heumann-keller-tuerpe_neu.pdf

TABLE OF FIGURES

ENDNOTES

1 http://cve.mitre.org/data/refs/refmap/source-EXPLOIT-DB.html
2 https://www.exploit-db.com/about-exploit-db/
3 https://www.reuters.com/article/us-equifax-cyber/equifax-says-15-2-million-uk-records-exposed-in-cyber-breach-idUSKBN1CF2JU
4 https://cve.mitre.org/cgi-bin/cvename.cgi?name=cve-2015-5477
5 https://f5.com/labs/articles/cisotociso/trends/cisos-striving-toward-proactive-security-strategies
6 http://carnegieendowment.org/specialprojects/ProtectingFinancialStability
7 https://www.us-cert.gov/bsi/articles/best-practices/architectural-risk-analysis/architectural-risk-analysis
8 https://f5.com/labs/articles/threat-intelligence/malware/from-ddos-to-server-ransomware-apache-struts-2-cve-2017-5638-campaign-25922
9 https://www.exploit-db.com/
10 https://www.helpnetsecurity.com/2017/07/12/magecart-monetize-stolen-payment-card-info/
https://blog.sucuri.net/2015/06/magento-platform-targeted-by-credit-card-scrapers.html
11 https://en.wikipedia.org/wiki/Point-of-sale_malware
12 https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf
13 https://www.owasp.org/index.php/Top_10_2017-A9-Using_Components_with_Known_Vulnerabilities
14 https://www.owasp.org/index.php/PHP_Security_Cheat_Sheet
15 http://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=CVE-2017-5638
16 https://www.owasp.org/index.php/Top_10_2017-A5-Security_Misconfiguration
17 https://nvd.nist.gov/vuln/detail/CVE-2017-9805
18 https://www.owasp.org/index.php/Deserialization_of_untrusted_data
19 https://www.owasp.org/index.php/Top_10-2017_A8-Insecure_Deserialization
20 http://www.nytimes.com/2011/06/02/technology/02google.html
21 https://moxie.org/software/sslstrip/
22 http://webpolicy.org/2015/08/25/att-hotspots-now-with-advertising-injection/
23 https://medium.com/@nicklum/my-hotel-wifi-injects-ads-does-yours-6356710fa180
24 https://www.infoworld.com/article/2925839/net-neutrality/code-injection-new-low-isps.html
25 https://csrc.nist.gov/Projects/Cryptographic-Module-Validation-Program/Standards
26 https://www.keylength.com
27 https://www.reuters.com/article/us-community-health-cybersecurity/u-s-hospital-breach-biggest-yet-to-exploit-heartbleed-bug-expert-idUSKBN0GK0H420140820
28 https://www.wired.com/2014/11/countdown-to-zero-day-stuxnet/
29 https://jblevins.org/log/ssh-vulnkey
30 https://www.usenix.org/system/files/conference/usenixsecurity12/sec12-final228.pdf
31 https://www.schneier.com/blog/archives/2011/03/comodo_group_is.html
32 https://www.pcworld.com/article/223760/article.html
33 https://techcrunch.com/2015/04/01/google-cnnic/
34 https://sslbl.abuse.ch/
35 https://arstechnica.com/information-technology/2017/03/google-takes-symantec-to-the-woodshed-for-mis-issuing-30000-https-certs/
36 https://www.digicert.com/blog/digicert-statement-trustico-certificate-revocation/
37 https://www.forbes.com/sites/thomasbrewster/2018/04/24/a-160000-ether-theft-just-exploited-a-massive-blind-spot-in-internet-security/#28dce3a45e26
38 https://thehackernews.com/2018/02/pos-malware-dns.html?m=1
39 https://krebsonsecurity.com/2018/02/domain-theft-strands-thousands-of-web-sites/
40 https://www.wired.com/2017/04/hackers-hijacked-banks-entire-online-operation/
41 https://www.fox-it.com/en/insights/blogs/blog/fox-hit-cyber-attack/
42 https://krebsonsecurity.com/2015/05/st-louis-federal-reserve-suffers-dns-breach/
43 http://www.theregister.co.uk/2014/12/17/icann_hacked_admin_access_to_zone_files/
44 http://www.kiro7.com/news/catholic-hospitals-suffer-second-data-breach-year/81976907
45 http://www.zdnet.com/article/dutch-dns-server-hack-thousands-of-sites-serve-up-malware/
46 https://www.scmagazineuk.com/first-true-native-ipv6-ddos-attack-spotted-in-wild/article/747217/
47 https://www.google.com/intl/en/ipv6/statistics.html
48 https://f5.com/labs/articles/threat-intelligence/ddos/the-global-playing-field-is-leveling-out-as-europe-and-asia-take-on-more-ddos-attacks
49 https://tools.ietf.org/html/rfc768
50 https://en.wikipedia.org/wiki/Smurf_attack
51 https://www.wired.com/story/github-ddos-memcached/
52 https://devcentral.f5.com/articles/mitigating-sslsqueeze-and-other-no-crypto-brute-force-ssl-handshake-attacks
53 https://tools.ietf.org/html/draft-nir-tls-puzzles-00
54 https://devcentral.f5.com/articles/ssl-renegotiation-dos-attack-ndash-an-irule-countermeasure
55 https://www.ietf.org/mail-archive/web/tls/current/msg07553.html
56 https://news.netcraft.com/archives/2017/02/17/hackers-still-exploiting-ebays-stored-xss-vulnerabilities-in-2017.html
57 http://www.zdnet.com/article/equifax-freeze-your-account-site-is-also-vulnerable-to-hacking/
58 https://f5.com/labs/articles/threat-intelligence/malware/trickbot-focuses-on-wealth-management-services-from-its-dyre-core?tag=TrickBot
59 https://f5.com/labs/articles/cisotociso/trends/wait-dont-throw-out-your-firewalls-25982
60 https://adam.shostack.org/blog/2009/10/are-security-best-practices-unethical/

ENDNOTES, CONT.

[61] https://en.wikipedia.org/wiki/Availability_heuristic
[62] https://www.owasp.org/index.php/OWASP_Dependency_Check
[63] https://www.owasp.org/index.php/Cryptographic_Storage_Cheat_Sheet
[64] https://en.wikipedia.org/wiki/Extended_Validation_Certificate
[65] https://www.owasp.org/index.php/HttpOnly
[66] https://f5.com/labs/articles/cisotociso/trends/cisos-striving-toward-proactive-security-strategies
[67] https://f5.com/labs/articles/threat-intelligence/ssl-tls/what-happens-to-encryption-in-a-post-quantum-computing-world
[68] https://caniuse.com/tls1-3
[69] https://blog.pcisecuritystandards.org/are-you-ready-for-30-june-2018-sayin-goodbye-to-ssl-early-tls
[70] https://f5.com/labs/articles?tag=quantum+computing
[71] https://support.f5.com/csp/article/K07359270

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