SIMPLE SAAS SECURITY FOR WEB APPS IN ANY CLOUD

F5 ESSENTIAL APP PROTECT SERVICE

INTRODUCTION

Web application security is difficult—very difficult. Not to mention time consuming and costly. Building and maintaining comprehensive web security controls can consume a large percentage of the limited budget you have for developing the actual application features users need to get useful work done.

In fact, web application security is so challenging that WhiteHat Security stated in its 2017 Application Security Statistics Report that the average web application has three vulnerabilities.[1] Are we are not investing enough in penetration testing and remediation? Do we not understand the risks? Are we not deploying the right tools to mitigate these vulnerabilities?

These are persistent, long-standing problems that remain omnipresent due to the difficulty of building and rebuilding remediations into every new application that is shipped. Understanding and defending against web application vulnerabilities typically requires focused security expertise, a skillset that few developers can realistically cultivate while getting actual development done at the same time.

Fortunately, you have options. Solutions like F5's Essential App Protect Service can go a long way toward mitigating risk, while simultaneously speeding development of your applications. Application security is difficult—but protecting your apps doesn't have to be. Read on to learn about some of the most common risk areas that affect nearly all web applications, and how to simplify your approach remediating risk across all of your apps, not just the most critical ones.

[1] https://info.whitehatsec.com/rs/675-YBI-674/images/WHS%202017%20Application%20Security%20Report%20FINAL.pdf

THE OWASP TOP 10

VULNERABILITIES AND MITIGATIONS

THE OPEN WEB APPLICATION SECURITY PROJECT: **CAN EDUCATION REDUCE VULNERABILITIES?**

The pervasive nature of web application security shortcomings has not gone unnoticed. In 2001, a number of security professionals banded together to create the Open Web Application Security Project (OWASP) to educate developers and reduce these security shortcomings. OWASP is a nonprofit international group that produces publicly available methodologies, documentation, tools, and training that addresses many aspects of web application security.

THE OWASP TOP 10: A TAXONOMY OF RISK

The most well-known of the OWASP projects is the “OWASP Top 10,” which is an ever-evolving list of the biggest security problems common to web applications. The goal of the OWASP Top 10 is to provide a basic taxonomy of risk with respect to web application vulnerabilities.

Future versions of the OWASP Top 10 are slated to be more closely aligned with widely accepted risk frameworks, such as ISO 31000:2015[2], boosting the credibility and applicability of the project, and, hopefully, increasing uptake of and adherence to its philosophy.

PROTECTING YOUR APPLICATIONS: AN OVERVIEW OF THREAT

If you are responsible for the development, security, or operation of a web application, becoming familiar with the OWASP Top 10 can help you better protect that app. In this e-book we’ll focus on the three most prevalent risk areas, and describe how to address them with reliable, easy-to-implement solutions.
By understanding the most common web app security problems and learning about effective mitigations, you can boost your organizational security posture, protect your critical applications, and help ensure the confidentiality, integrity, and availability of your data.

[2] https://www.iso.org/iso-31000-risk-management.html

INJECTION

Injection is a common class of vulnerability where insufficiently sanitized input provided by external sources contains hidden application commands from an attacker. Because the web application is not properly filtering the input, it allows injected commands to be passed through to either the local system or a dependent one.

A common example is the SQL injection attack. Many applications rely upon user input to build SQL statements to fetch information or to log them in. For example:

```
select * from USERTABLE where USERID = '[userid-from-web-form]' and
PASSWORD = '[passwd-from-web-form]'
```

Under normal circumstances this could match an entry in the data table USERTABLE. In that case, the statement evaluates to "True" and the login succeeds.

But what if the user puts in "password' OR 1=1" as the password on the web form:

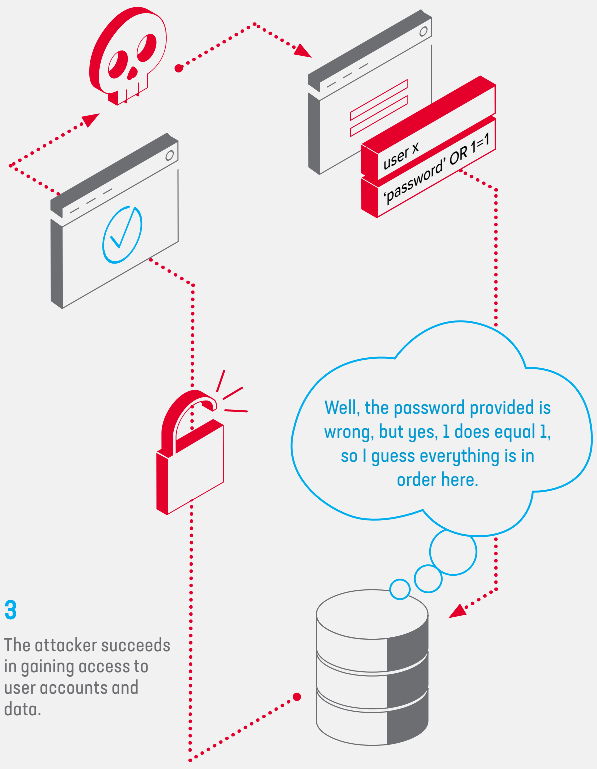
```
select * from USERTABLE where USERID = '[userid-from-web-form]' and
PASSWORD = 'password' OR 1=1;
```

Without proper sanitization and escaping, the SQL server will evaluate the 1=1 conditional as TRUE and log the user in with the username provided regardless of what password was supplied. This is a very simple and targeted example. SQL injection attacks can get far more sophisticated and malicious, and have been used successfully to delete entire databases, modify records, and exfiltrate sensitive data.

SQL ATTACKS HAVE BEEN USED TO DELETE ENTIRE DATABASES, MODIFY RECORDS AND EXFILTRATE SENSITIVE DATA.

1

An attacker sends a request with an injected command from a browser/app for a web resource.

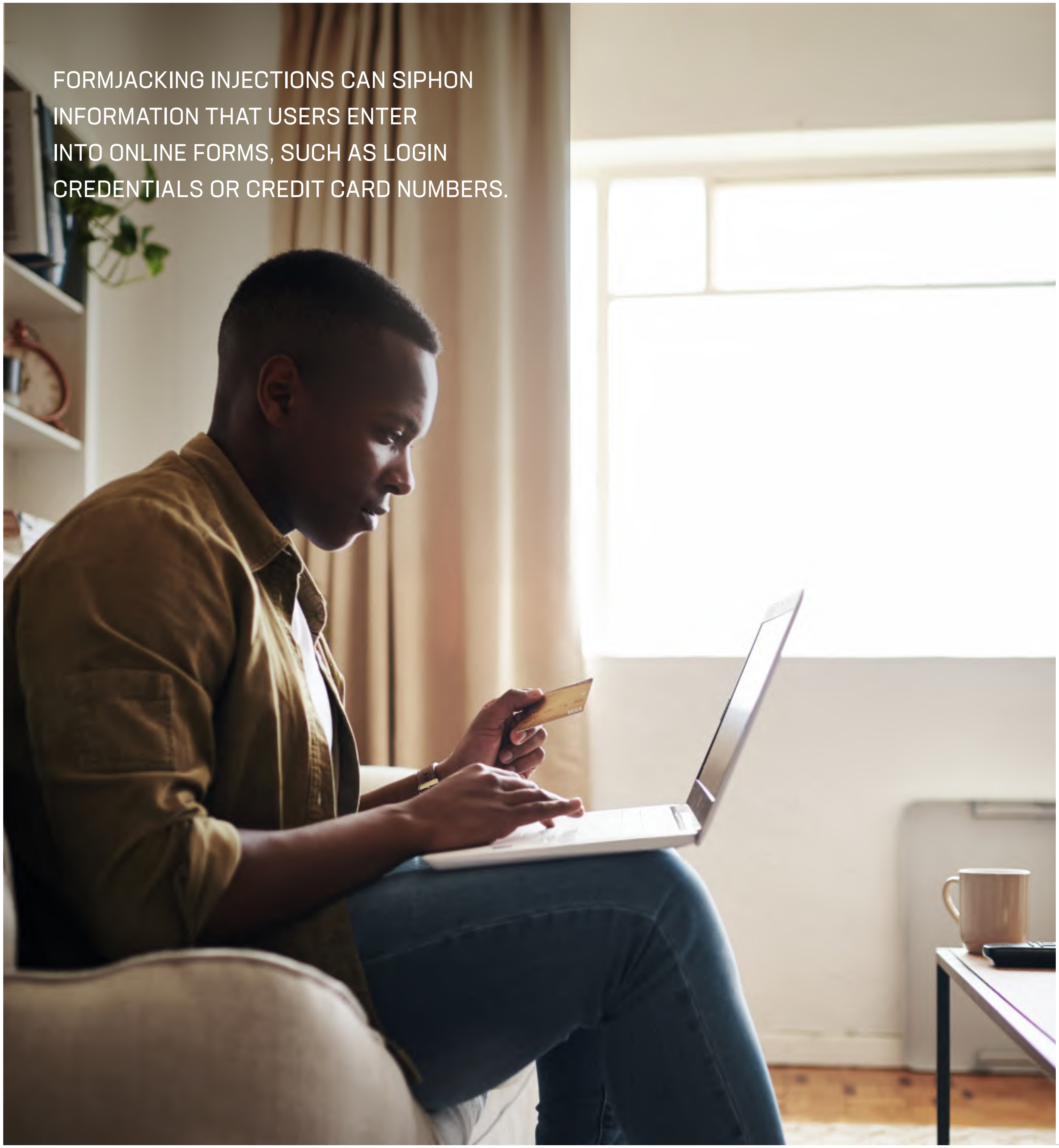

user x
'password' OR 1=1
Well, the password provided is wrong, but yes, 1 does equal 1, so I guess everything is in order here.

3

The attacker succeeds in gaining access to user accounts and data.

2

Even though the password may be incorrect, The data base agrees that 1 = 1, and the user is authenticated.



FORMJACKING INJECTIONS CAN SIPHON INFORMATION THAT USERS ENTER INTO ONLINE FORMS, SUCH AS LOGIN CREDENTIALS OR CREDIT CARD NUMBERS.

MITIGATING INJECTION RISKS

With respect to data input and security, you cannot inherently trust any data from the user. All input mustbe examined, escaped, sanitized, and filtered. Injection attacks can occur in normal user input forms as well as in hidden web fields. Leveraging parameterized SQL[3] can go a long way toward mitigating this risk by compartmentalizing input data and distinguishing it from code, regardless of user input. It is also advisable to monitor outbound responses returned to the user in an effort to detect information leakage resulting from a successful injection attack.

[3] https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet#Defense_Option_1:_Prepared_Statements_.28with_Parameteized_Queries.29

SENSITIVE DATA EXPOSURE

Sensitive data exposure is an information leakage problem. The sensitivity of what is leaked can vary, and divulging any information about how a web application is designed to an attacker is a bad idea. This kind of information is low-hanging fruit for automated scanners and ripe for exploitation.

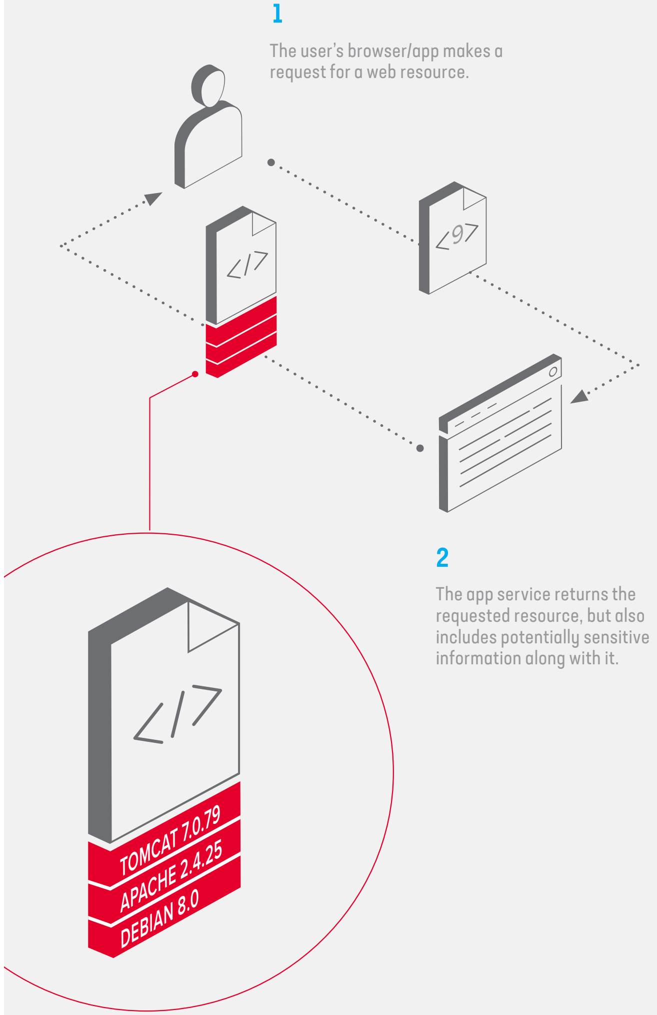
Some examples of the kinds of information commonly leaked by web applications that attackers find useful include the following:

- Error messages detailing how unexpected input is handled
- Physical locations of files on server
- Specific versions of components and libraries
- Stack traces from failed functions (can be decompiled and examined)
- “Forgot password” function error messages that reveal user ID validity, which can be used for discovery and brute-force attacks on user accounts and passwords

MITIGATING SENSITIVE DATA EXPOSURE

There are several steps you can take to minimize your risk for data leakage. It is very common for web servers to report vendor and version information, among other things.[4] Make sure that usernames cannot be validated from server response codes: an incorrect username error and incorrect password should generate the same error message. Ensure browser security directives are used to help protect sensitive data in transit. Avoid old and known weak cryptographic algorithms and methods. Transport Layer Security (TLS) is easy to use and is becoming the universal norm on the Internet.[5] And finally, all sensitive data stored within a web application should be rendered unreadable—using techniques such as encryption or tokenization[6]—in case an attacker gains access through the application.

WEBSITES OFTEN RETURN MORE DATA THAN IS NECESSARY, WHICH GIVES ATTACKERS ADDITIONAL INFORMATION TO USE AND EXPLOIT.



1

The user’s browser/app makes a request for a web resource.

2

The app service returns the requested resource, but also includes potentially sensitive information along with it.

[4] https://f5.com/labs/articles/threat-intelligence/identity-threats/phishing-for-information-part-4-beware-of-data-leaking-out-of-your-equipment
[5] https://f5.com/Portals/1/PDF/labs/R065%20-%20REPORT%20-%20The%202016%20TLS%20Telemetry%20Report.pdf
[6] https://www.owasp.org/index.php/File:Reducing_Your_Data_Security_Risk_Through_Tokenization.pptx

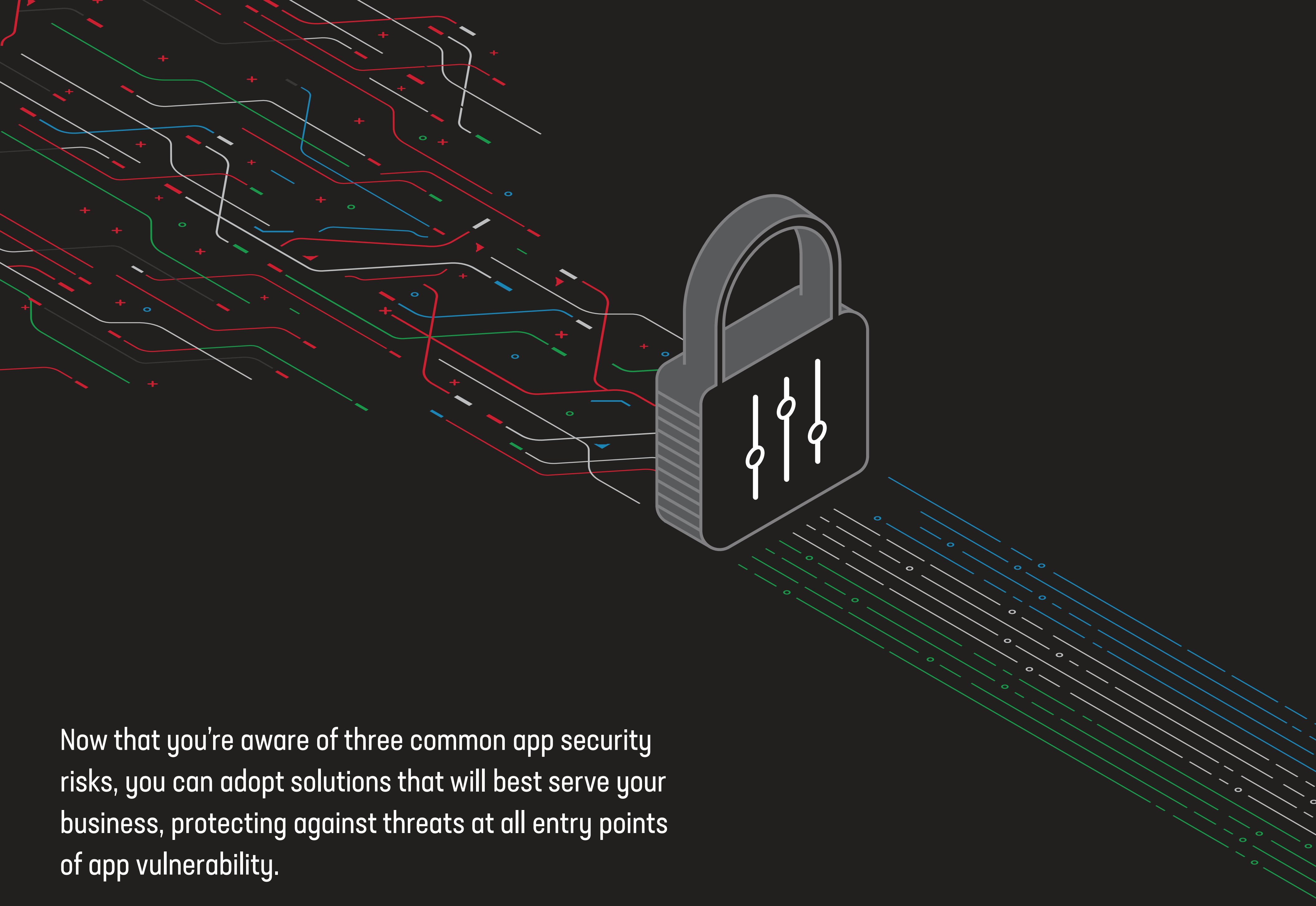
IF YOU'RE DEPLOYING APPS IN THE CLOUD, VULNERABILITIES INCREASE IN COMPLEXITY AND BECOME MORE COMPLICATED TO MANAGE BECAUSE OF ALL THE DEPENDENCIES.

USING COMPONENTS WITH KNOWN VULNERABILITIES

This is another one of those risk areas that may seem obvious, but it is worth addressing since many software components are chosen solely for their utility in fulfilling some basic operational requirement. Such components may not have known vulnerabilities when implemented, and it is common for there to be resistance to upgrades out of fear of breaking functionality or losing a valuable legacy feature. These are valid concerns, but a successful exploit against a known security vulnerability can result in a significant loss of service or breach of customer confidence, so this risk must be weighed against the perceived risks of upgrading.

MITIGATING USING COMPONENTS WITH KNOWN VULNERABILITIES

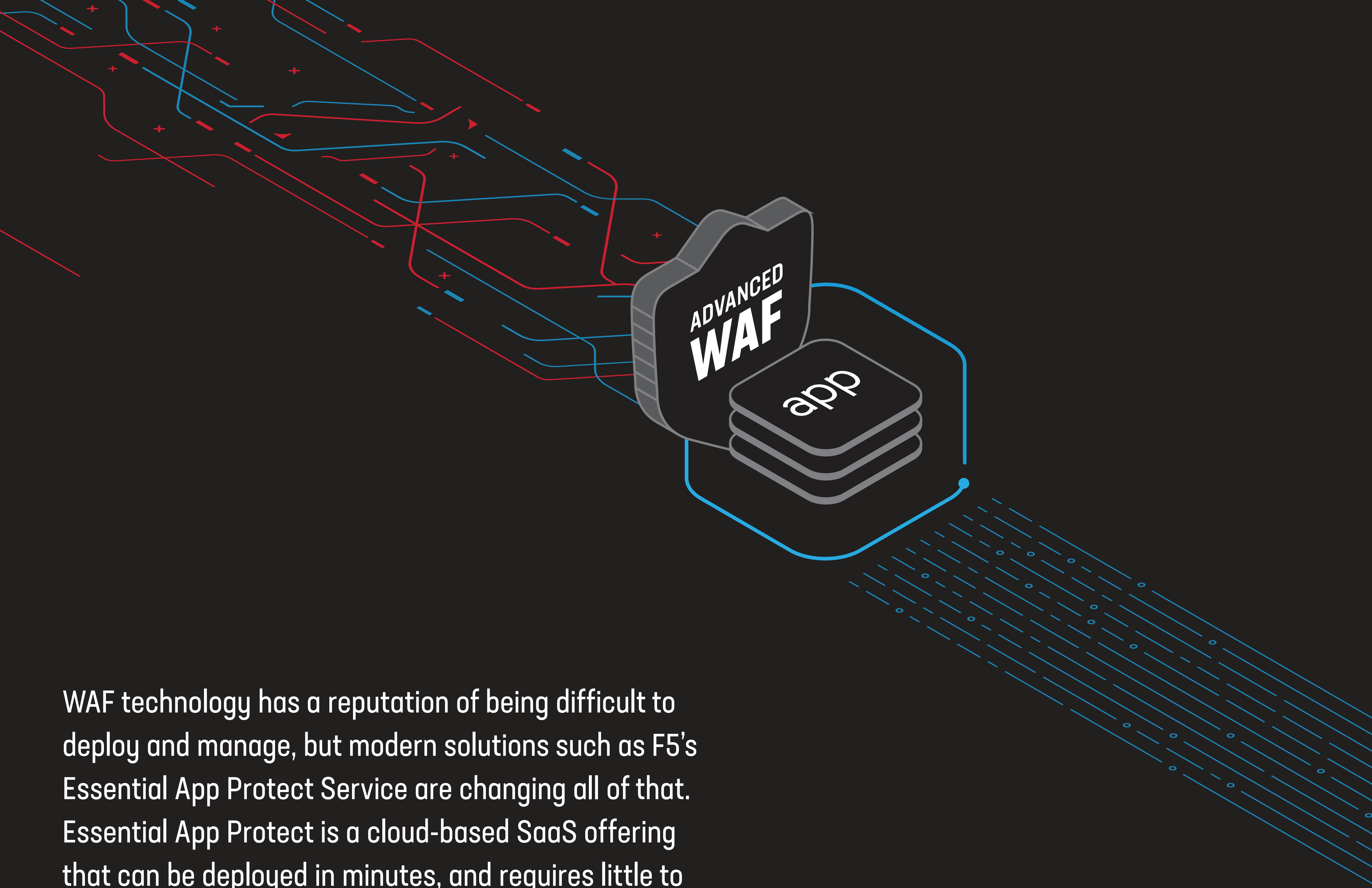
The key here is to keep components updated wherever you can. Where you can't, compensating controls such as a strong WAF will enable you to block the exploitation of known vulnerabilities while allowing you to keep your software intact and operational. A WAF can also buy you time while patches are developed and rolled out.



Now that you're aware of three common app security risks, you can adopt solutions that will best serve your business, protecting against threats at all entry points of app vulnerability.

The good news is that there are tools to help you bolster your apps against breaches by mitigating vulnerabilities and stopping attacks: specifically, web application firewalls (WAF). A WAF inspects ingress and egress application traffic to identify and block scanners, attackers, and bots, while preserving and accelerating apps for legitimate usage. Whether deployed on premises, leveraged in the cloud, or consumed as a service, WAF technology can help defend your organization against web app attacks, which are the primary entry point of successful data breaches.[7]

[7] https://www.f5.com/labs/articles/threat-intelligence/lessons-learned-from-a-decade-of-data-breaches-29035



WAF technology has a reputation of being difficult to deploy and manage, but modern solutions such as F5's Essential App Protect Service are changing all of that. Essential App Protect is a cloud-based SaaS offering that can be deployed in minutes, and requires little to no ongoing maintenance.

Although we all know we should have robust security solutions, it can be hard to justify spending money on them. The fact is, attacks can be nearly impossible to predict. Security is often regarded as a necessary evil with no quantifiable ROI, but that doesn't always have to be the case. In the world of cloud computing and big data, good security solutions can actually save you money by helping you optimize your web applications and digital properties—and they can do it while still protecting your business from attacks.

When it comes to safeguarding your applications, F5's Essential App Protect shields your web-facing apps—as a service—without sacrificing speed and agility.

Even better, Essential App Protect comes pre-configured. Using our 20+ years of networking and security expertise, we have optimized setup so that the service can be activated with a few UI clicks or API calls. There's no hardware or software to manage, and deployment can easily be dropped into a DevOps toolchain, delivering basic security controls for any app.

Developers can configure their app protection or WAF features through the rich declarative API or via an intuitive user interface. With an actionable threat map, developers have the ability to view threat origination data, immediately respond to events, or send alerts to their security operations teams.

Within minutes of activation, your app will have out-of-the-box protection against common web exploits, malicious IPs, and coordinated attacks—with no security expertise required. Tight integration with F5 Labs threat intelligence ensures that Essential App Protect Service utilizes real-time insights to defend your app from evolving threat vectors.

CONCLUSION

Web application protection is more essential than ever to maintaining service continuity for all apps. It is no longer sufficient to deploy WAF technology for mission critical applications alone.

Take the complexity and cost out of safeguarding your web-facing apps. F5’s Essential App Protect Service is simple to implement and even easier to manage. It is a true hands-off SaaS web application solution that offers simple and effective security for web apps in any cloud.

INTRIGUED?

Try EAP for free today.
f5.com/solutions

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