WhiteSource

WHY BANKS ARE OVERLOOKING APPLICATION SECURITY

CONTENTS

INTRODUCTION

APPLICATIONS VS. OPERATIONS 4

THE PITFALLS OF APPLICATION SECURITY 5

- PITFALL #1: SECURE BY DESIGN 5
- PITFALL #2: THE INEVITABILITY OF VULNERABILITY 6
- PITFALL #3: THE MANY FLAVORS OF PATCHING 7
- PITFALL #4: DEVICE SECURITY IS NOT THE (FULL) ANSWER 8

OPEN BANKING INTRODUCES NEW CHALLENGES 9

APPENDIX 10

HOW APPLICATION SECURITY TOOLS COULD HAVE PREVENTED THE EQUIFAX BREACH

INTRODUCTION

Cybersecurity is quickly becoming a top concern for banks, and it is easy to understand why. Ransomware alone is thought to have caused damages exceeding

$5 BILLION IN 2017

and that is only one category of cyber attack. Many cybersecurity vulnerabilities faced by banks are self- inflicted, but luckily, they can be easily addressed if one knows what to do.

Banks have to contend with the operational security of their own internal solutions. They also need to support a growing array of customer-facing solutions ranging from ATMs and card readers to mobile banking applications and websites. Somewhere in there, most banks are also writing applications for internal use as well as creating applications for use by suppliers, partners, and most importantly, their customers.

It is safe to say that modern banks are completely and utterly reliant on information technology to function and applications for interacting with their customers. As a result, cybersecurity problems simply cannot be ignored.

Attempting to address cybersecurity is daunting. There are already a nearly incomprehensible number of cybersecurity threats, and the threat landscape is diversifying daily.

Organizations that are unsure about their security posture are likely feeling overwhelmed by pressure from customers and regulators to solve all aspects of security simultaneously.

Unfortunately, the end result from this confusion is often that key areas of security get overlooked, opening the door for an ambitious attacker.

So, where is the low hanging fruit?

APPLICATIONS VS. OPERATIONS

When most people think about cybersecurity, they think about traditional IT operations security concerns. Access and authentication, anti-malware solutions, endpoint and wireless security, encryption, and monitoring are the current, top IT security spending trends.

Traditional IT operations security concerns attract the lion's share of spending and are viewed by most organizations as highly effective. At the same time, application security and compliance are the most in-demand skill sets, with organizations ranking their application security efforts as among their least effective endeavors.

While there is a correlation between money spent and reported effectiveness, this does not imply causation. Traditional IT operations security concerns have more mindshare. This is in part because, for the past several decades, operations-focused IT security has been a lucrative market, with companies pumping billions of dollars into marketing and awareness.

Application security is another story. For decades, application security has largely relied on education efforts aimed at training developers in secure application development. Efforts at an organizational level have focused on "culture," or adoption of various nameplate methodologies. Building applications that are "secure by design" comes quickly to mind as well as the idea of shifting left development to catch issues early in the hopes of preventing tear and replace ops that can push off release schedules.

Unfortunately, these efforts have not been as successful as we might have hoped, due in large part to a lack of education in the market or serious consequences to the offenders. Organizations in the application security space have established guidelines to help push the software industry in the right direction.

Thankfully, solutions like Static Application Security Testing (SAST) and Dynamic Application Security Testing (DAST) are being incorporated into the development process to scan through the in-house code. These methods use sets of rules in their algorithms to seek out potential bugs within the code. While still far from a perfect solution, it has gone a long way in helping developers to write better proprietary code.

However, dealing with open source is a different issue that developers and security teams need to contend with. By some estimates, applications have an average of 24 components with known vulnerabilities inherited, something that malicious actors can now exploit automatically.

Heavily relied on for building applications across multiple organizations, open source components face the challenge of having their vulnerabilities publicized so that developers can know to go in and patch their products. The problem is that just as the developers hear about the exploitable vulnerabilities, so do the hackers who can use the information for ill-gotten gains.

This is not a new problem. The alarm has been sounded about third-party application components making organizations vulnerable for many years. Unfortunately, application security remains a neglected aspect of security.

THE PITFALLS OF APPLICATION SECURITY

For banks, addressing application security issues may represent the biggest bang for the security buck currently available. To understand why, we need to look at the common issues that banks encounter in providing secure solutions for their customers.

SECURE BY DESIGN

The European Union's General Data Protection Regulation (GDPR) becomes enforceable on May 25, 2018. It is the poster child for a new generation of data protection regulations that include critical — albeit often vague — concepts, such as security by design and by default.

One of the practical results of these laws is that application developers have no choice but to implement better security practices to their development process. Even if we put all other considerations aside, designing a secure application requires the use of encryption, and the golden rule of cryptography is to never roll your own crypto.

Encryption underlies much of modern software security. TLC encryption is used to secure communications between applications (or web browsers) and back-end servers. VPNs are used to provide more complete, secure tunnels that allow applications or devices to securely connect to a network. In many jurisdictions, it is legally mandated that data storage for sensitive data be encrypted. This makes encryption hard to avoid for many application developers.

The cryptography field is broken into two broad groups: those who create new crypto, and those who implement crypto. Many individuals implement cryptography in one form or another. They either incorporate it into an application (or application component), or they apply those applications from an IT operations standpoint.

As for the other group of individuals within the cryptography field, there are so few cryptographers in the world with the skills and experience to create new, viable cryptographic algorithms that in many cases, their employers will not let more than one of them travel on the same plane. These individuals are so highly sought-after that enterprises and even governments carefully consider the bus factor of having too many key individuals congregate in one place before authorizing travel.

As a result of the shortage of cryptographers, banks are unlikely to have the right individuals on hand to implement even the bare minimum legally mandated cryptography within their applications without using third-party application components. The world's cryptographers work together to design not only the algorithms we all use, but also the application components that everyone uses to make those encryption features available.

Enshrining "secure by design" into law is thus — in practice at least — enshrining the use of third-party application components in law. One cannot be accomplished without the other.

PITFALL #2

THE INEVITABILITY OF VULNERABILITY

All code has bugs. Software bugs create security vulnerabilities. Third party components that provide cryptographic functionality are not immune.

Common encryption components have had some famous vulnerabilities over the years. Heartbleed, Shellshock, Krack, and Log Jam have all made headlines around the world. Some of these have been easily patchable, and some of them are more fundamental, requiring abandoning older cryptographic solutions for newer ones.

Even OWASP mentions this issue in their famous "Top Ten List" of application security risks, which highlights the most common errors that occur and may be easily exploited.

In 2013 OWASP added "A9: Using Components with Known Vulnerabilities" and warned organizations that the risk of using a third party components with a known vulnerability is becoming widespread and sometimes critical as both open source usage is rising exponentially year over year, and as a result more vulnerabilities are being published.

Dealing with vulnerabilities in common encryption solutions requires patching, changing the default algorithms in use, and every so often, it means changing the encryption solution entirely. But most organizations are currently blind to the fact that they are using vulnerable components, while hackers can easily exploit it.
(see note regarding Equifax)

THE MANY FLAVORS OF PATCHING

When most people think about patching, they think about patching their device's operating system. Every computer user has encountered Windows Update, MacOS update, or the update mechanisms on their smartphones. These ubiquitous patching systems are well known, but they are not the only place where patching has to occur.

Third-party components are frequently included in compiled applications directly. This means that those components need to be patched before the application building process begins in order for those applications to contain the most recent components.

Unfortunately, there is not an easy way to tell which components are patched by the operating system and which need to be patched during the application build process. Consider for a moment the Heartbleed bug. Heartbleed was a flaw in numerous versions of the OpenSSL cryptography library, a common open source component.

OpenSSL is frequently downloaded, installed, and managed by package management solutions on Linux operating systems.

Many Linux applications are designed to call and use the copy of the OpenSSL libraries that are managed by the operating system's package manager. Here, regular, automated updates to the operating system would update OpenSSL, and any application using it would make use of the new version at application restart without having to recompile or redeploy the application.

OpenSSL is, however, also built directly into innumerable applications, especially where those applications are destined for use on Windows. Windows' package manager does not install or manage any version of the OpenSSL libraries. In this case, the applications need to be recompiled (or at least repackaged) every time a new version of the OpenSSL library comes out.

This same discussion could be repeated for numerous other common categories of components. There are a number of frameworks and libraries that have become so common that many modern developers cannot write applications without them.

DEVICE SECURITY IS NOT THE ANSWER

Although it consistently ranks amongst the top security expenditures, the problem of applications that make use of open source components with known vulnerabilities cannot be solved through the use of device security. No anti-malware solution, application behavior profiling tool, endpoint security suite, or mobile device management solution can make an application with a known vulnerability not be vulnerable.

Building one's applications to take advantage of a device's advanced security features will not solve the problem either. Biometrics, NFC, and multi-factor authentication are all just buzzwords if the application they are meant to secure is itself vulnerable. In addition, incorporating each of those security features into one's application is likely to require a number of additional, third-party components, each of which will also need to be kept up-to-date.

From a bank's point of view, this should be worrisome. It means that the burden of security cannot be pushed back onto the customer or any of the vendors that exist in the supply chain between the customer and the bank.

If a bank deploys an application that includes open source components with known vulnerabilities — or if the bank does not update and immediately deploy a new version of its applications as soon as vulnerabilities are discovered in components it uses — the bank is liable. Pointing the finger at the customer for lax security practices, or at the device manufacturer or operating system vendor will not help.

If the vulnerability exists as part of the code deployed with the application, the onus is on the company that built that application to keep it up-to-date.

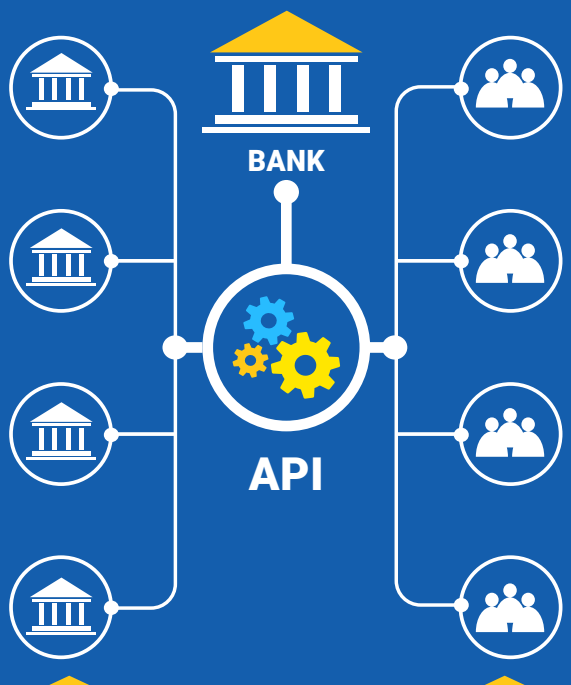
OPEN BANKING IS INTRODUCING NEW CHALLENGES

Adoption of open banking is becoming a necessity, but it is also adding new challenges when it comes to securing your customers' data. Depending on how one parses the risks associated with application development, open banking can seem like the perfect solution to risk management, or a disaster waiting to happen.

Open banking shifts the application development burden to a third party. In open banking, a commercial entity that is completely unaffiliated with the bank develops an application.

This application communicates with Application Programming Interfaces (API) published by the bank.

Open banking applications can typically work with multiple banks, providing a single user experience for end customers, regardless of the bank they choose to use.



With the application being designed, published, and maintained by a third-party developer who is not contracted to the bank, the liability for keeping that application up-to-date rests on the open banking application developer, not on the bank.

While that may be desirable from a liability standpoint, it also means that the bank has little control over whether or not their customers are using applications with known vulnerabilities, requiring banks to redouble their back-end anti-fraud efforts.

Trust is key. Trust needs to exist between the bank and the developer of the open banking application. It must also exist between the customer and the bank, as well as between the customer and the developer of the open banking application.

Trust in the open banking application world is often complicated by changes in the customer relationship. Retail banks are used to owning the customer relationship for decades at a time. However, with open banking, the developer of the open banking application owns much of the customer's day-to-day banking experience.

As the usage of open banking applications proliferates, the market is becoming increasingly competitive. Many open banking application developers are making an effort to compete on trust, attempting to nurture relationships with the banks their applications interact with, and pursuing certifications and formal code reviews of their applications.

Banks that embrace open banking can foster increased security through the use of partner application security initiatives. They can also boost their security by insisting upon audits that prove developers of open banking applications are employing all of the tools necessary to ensure that the deployed applications are kept safe by updating them in a timely manner.

APPENDIX

When banks and developers work with components that they take from outside sources, they need to trust that they are using code that will help them put out a superior product, but verify for themselves that those components are are in fact safe.

Verifying that an open source component does not contain any known vulnerabilities takes using the right tool for the job. If SAST and DAST can lend a hand on dealing with proprietary code, then it would reason that there be a specific tool that works with open source components to secure the applications.

HOW APPLICATION SECURITY TOOLS COULD HAVE PREVENTED THE EQUIFAX BREACH

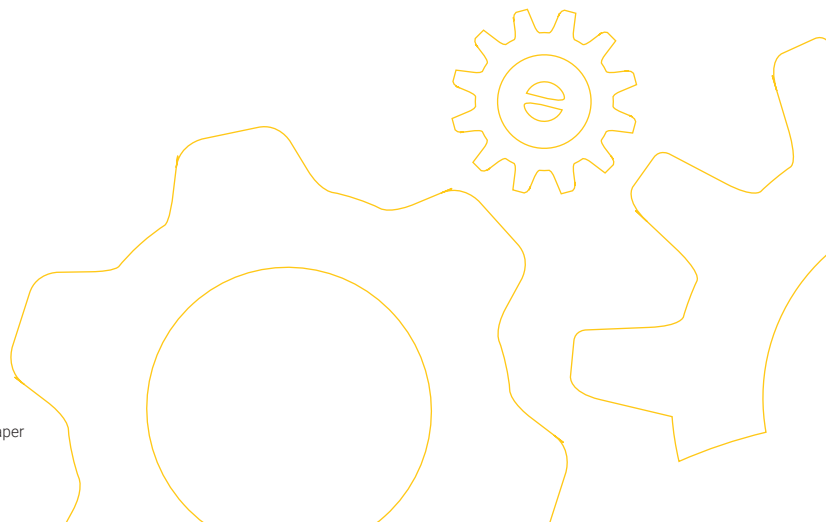
Whereas SAST and DAST can help sift through lines of code, Software Composition Analysis (SCA) tools have a much easier lift since open source components can be easily identified through their hashes, also known as SHA-1s. These hashes are posted on the vulnerability databases, and can be matched with a component in an inventory or repository.

The problem is that developers are not always diligent about keeping track of which components they are adding to their products, grabbing and commiting the code. This means that if a new vulnerability is announced, a developer team will not necessarily know that their product contains the exploitable bit of code, and are left unprotected. We witnessed this story play out last year when a vulnerability was announced in March for the Apache Struts2 project, a very popular piece of open source code that is helpful for building web applications.

Through this component, hackers were able to break into one of the big three credit rating agencies, Equifax, and make off with the personally identifiable information belonging to 145 million people.

The biggest breach in history did not come from a clever social engineering or from someone pouring over code for months. The attackers used a well publicized exploit in a popular project, hitting the company two months after the vulnerability and patch had already been announced.

From Equifax's testimony and statements, we understand that they were using various testing solutions, but clearly did not have an SCA tool that could have identified for them that they had the risky code, alerting them as soon as it was announced. The painful lesson here was that you can't patch what you don't know that you have.



Despite the widespread nature of the problem, application security initiatives remain in their infancy in most markets. Banks cannot afford to be part of those statistics, especially when simple solutions exist.

SCA tools such as application component vulnerability scanners must become part of everyday security for all banks. Their use by internal development teams, contract application developers, and third-party, open banking application providers must be insisted upon.

**In banking, and everywhere else,
modern cybersecurity must include application security.
It is an unavoidable part of security by design.
It is not just best practice — in nation after nation,
it is increasingly becoming the law.**