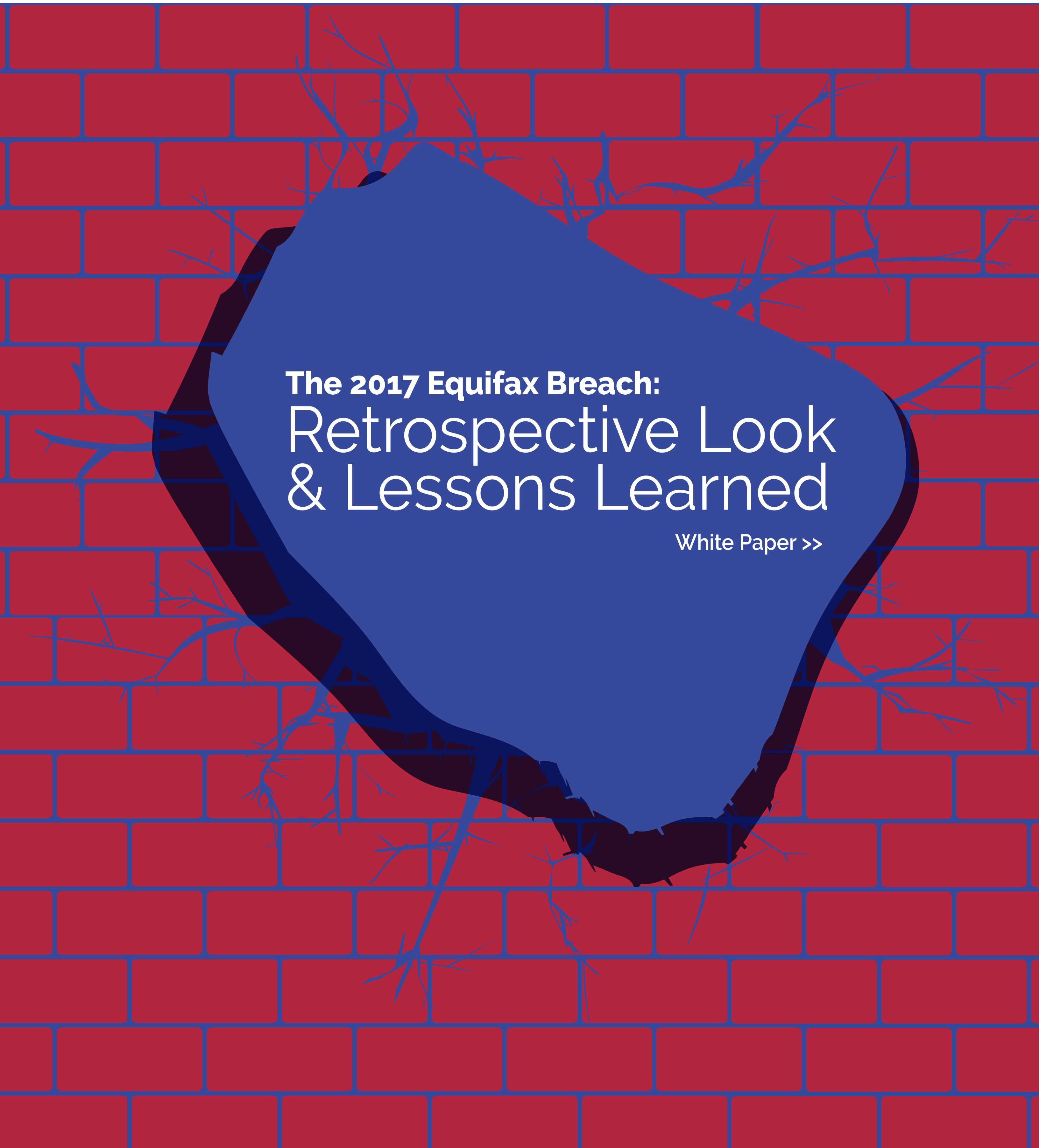
WhiteSource

Microsoft



The 2017 Equifax Breach: Retrospective Look & Lessons Learned

White Paper >>

INTRODUCTION

The Equifax breach that was announced in September 2017 is considered to be the biggest breach in history with hackers having stolen the personally identifiable information (PII) of nearly 147.9 million of the company's customers, primarily located in the United States.

PII from the Equifax database and dispute records included names, Social Security numbers, birth dates, addresses and driver's license numbers. Upwards of 200,000 credit card numbers were also stolen, adding even more damages to the theft..

The 2017 breach is characterized by a succession of errors which began with the company's slow response to the Apache Struts 2 vulnerability. Equifax failed to detect the vulnerable open source component in their web application and implement its fix in a timely manner.

The vulnerability was made public March 7 2017 and a fix was issued that same day; two months before Equifax officials claim that the first criminal activity occurred in their database.

Following detection, the company waited six weeks to go public with the security breach. Equifax discovered that it had been hacked on July 29, 2017, and only announced it publicly in early September 2017.

An additional set of errors were made from the public relations perspective. Before the company went public with news of the breach, a number of Equifax executives sold off close to $1.8 million worth of their company stock in expectation of the foreseeable drop in stock price. As a result of public pressure following a series of catastrophic managerial mistakes, the company's CISO, CIO, and CEO resigned from their positions.

Given the magnitude of the Equifax breach, the objective of this white paper is threefold. Primarily it is to uncover the technical malfunctions of this case so that companies become better equipped to respond to open source vulnerabilities from the earliest stages. Secondly, the objective is to outline the lifecycle of an open source vulnerability, showing how hackers might use it as an attack vector. Lastly, to lay bare the managerial errors made by Equifax executives so that C-suites can learn how not to behave in moments of corporate crisis.

CONTENTS

CHAPTER 01

EQUIFAX IN A NUTSHELL

Equifax is one of the big three credit reporting bureaus in America. Along with TransUnion and Experian, Equifax holds personally identifiable information (PII) of all American citizens, acting as mediator between creditors and customers applying for services such as loans and credit.

Equifax collects PII from firms that issue credit, such as banks, credit card companies, and credit unions. All three credit reporting agencies (CRAs) are permitted by federal law to compile personal data of US citizens independently and without consent. This means that by being part of the financial system — having a bank account or a credit card — one enters the databases of the three credit bureaus without taking any direct action on their part.

In addition to passive enrollment into the agency's database, many Americans actively seek Equifax's services for the purpose of assessing their creditworthiness. Consumers will often register with one or all three credit bureaus before applying for a loan as a means to assess loan eligibility, or at other times for a more general financial sanity check.

Equifax Database

The Equifax database consists of personal information including full names, Social Security numbers, driver's license numbers, birth dates, and home addresses. It also includes detailed financial information about existing and past credit files, loan history, bankruptcy history, payment history from a variety of financial institutions including banks, credit card companies, mortgage providers, and other creditors.

Payment records from utility companies are collected as well, as is information regarding tax liens, judgements and delinquent debt, charge-offs, and bankruptcy filings.

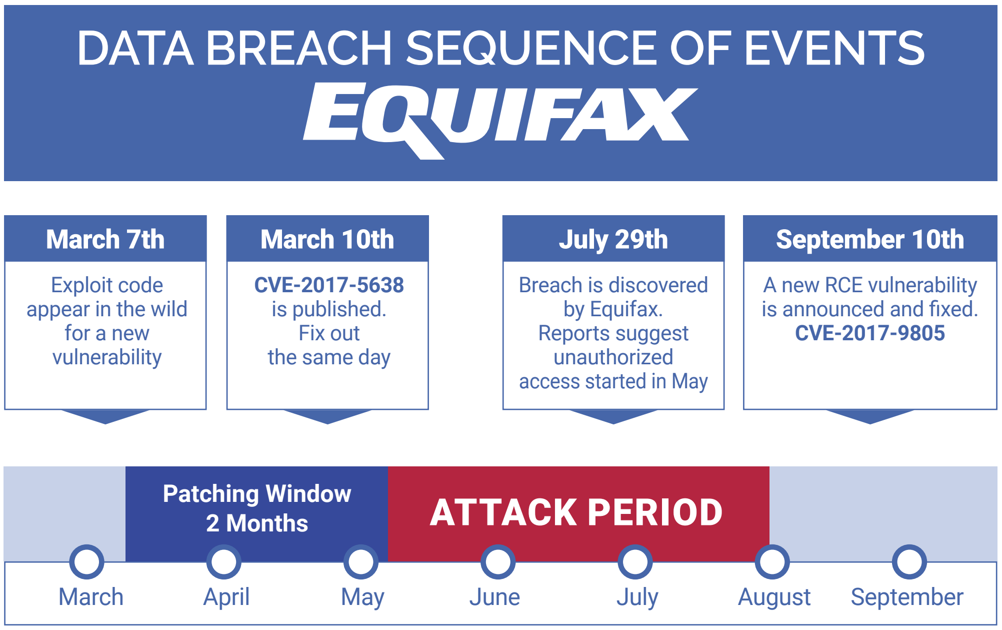
If not properly secured, a database that holds sensitive PII inevitably puts those featured in the database at risk. Once vulnerable, Equifax's database became a premium source for identity theft.

CHAPTER 02

THE BREACH

The Play By Play

Hackers found their way into Equifax's poorly secured database through a vulnerability in the Apache Struts 2 open source component used in their customer portal web application. The vulnerability was first made public by the Apache Foundation on March 7, 2017 and posted on the National Vulnerability Database (NVD) under the CVE-2017-5638 identification. On the same day a patch had already been issued and was available for use.


DATA BREACH SEQUENCE OF EVENTS
EQUIFAX
March 7th
Exploit code appear in the wild for a new vulnerability
March 10th
CVE-2017-5638 is published. Fix out the same day
July 29th
Breach is discovered by Equifax. Reports suggest unauthorized access started in May
September 10th
A new RCE vulnerability is announced and fixed. CVE-2017-9805
Patching Window 2 Months
ATTACK PERIOD
March
April
May
June
July
August
September

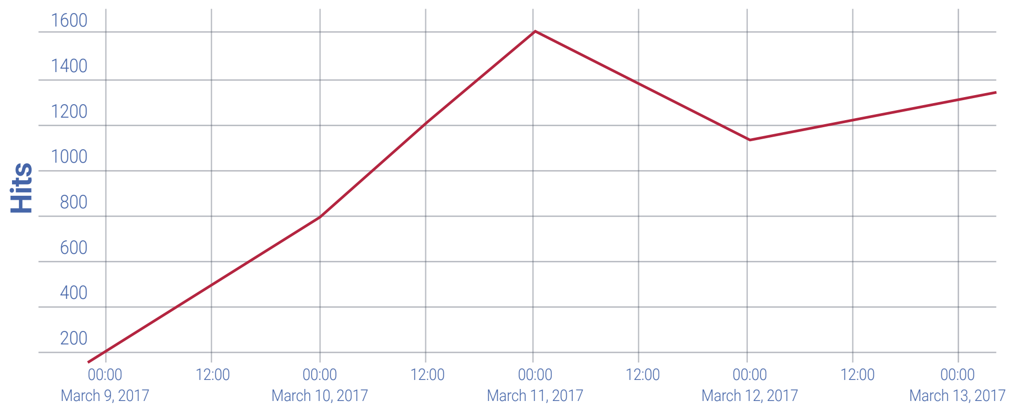
The March 7 2017 publication of the vulnerability led to a massive surge in the attempts to exploit users of the component. The onslaught of attacks in the first week of going public made it clear that hackers were feeding off the media frenzy, and caused many companies to take immediate active measures to protect their software.

[1]Apache Struts released the following statement September 14, 2017: "This vulnerability was patched on 7 March 2017, the same day it was announced. ... In conclusion, the Equifax data compromise was due to their failure to install the security updates provided in a timely manner."

Hackers are still exploiting the bug to install malware on high-impact sites.

Dan Goodin - 3.14.2017, 9:30 PM

CVE-2017-5638 Exploitations Attempts



Despite extensive media coverage, Equifax did not follow suit and did nothing to check for the vulnerability in the time surrounding its publication. In fact, it was not until July that Equifax first noticed that their system may have been compromised. It was this lack of diligence on the part of Equifax's development and security teams that ultimately led to the exploitation of sensitive data belonging to 147.9 million consumers.

Equifax officials have stated their belief that the attack began around May 13, 2017. By that date, a fix for the vulnerability had already been available for over two months.

It was only on July 29, 2017, four months after the vulnerability was first published and a fix was made available, that Equifax noticed suspicious activity on their online dispute portal and decided to dig deeper.

By early August, with the help of FireEye-owned cyber security firm Mandiant, Equifax realized that the breach had infiltrated far into their systems and hit the most sensitive of its databases where PII was being stored.

It took another month for Equifax to publicly announce the breach on September 7, 2017 and for the patch to be put in place[2].

[2]Timelines of the breach vary from one source to another. Inconsistencies in timeline may be the result of updated information published as the dust settled and more data came to light. It is worth noting that varied accounts are generally characteristic of high profile events covered by hundreds of outlets and relying on a multitude of sources.

Following Equifax's announcement, on September 9 Apache Struts issued a statement responding to the Equifax data breach that included details on its response process to reported vulnerabilities and also provided recommended security guidelines.

By September 13, 2017, Equifax issued yet another statement, this time confirming that the vulnerability was Apache Struts CVE-2017-5638. A vulnerability which was made known already March 7, and for which a patch was made available that same day.

Announcement and Action Taken

Following the announcement, many of the company's C-suite executives embarked on what seemed like a resignation frenzy. Amongst those to step down were the company's CISO, CIO, and CEO.

As a means to support those affected, Equifax followed the textbook path of offering one year identity theft protection service free of charge. The company also put up a site under the address hxxp.equifaxsecurity 2017.com assisting those affected to assess their personal damages.

Equifax announced that it expects to invest $275 million in breach related costs in 2018. The costs mainly reflect data security upgrades and the integration of open source best practice policies to prevent more breaches in the future. This number does not include legal fees, free identity theft protection and credit monitoring packages to the 147.9 million consumers affected by the breach.

Corporate Etiquette

It was retroactively discovered that a small number of Equifax executives who had known about the breach before its official September 2017 release had secretly sold off nearly $1.8 million worth of their stock in anticipation of the dip in stock price.

Equifax's damage control mechanisms were also criticized regarding the impromptu support site the company put up to help those affected gain insight into the state of their violated identities. They were roundly pillaried for this site, as it was not on their regular domain and could be easily used by scammers for phishing operations.

This was followed by rumors that the company had retracted their offer of free identity protection and credit monitoring services to the breach's 147.9 million victims.

Equifax is facing a 50-state class-action suit following the breach. On October 3, 2017, former Equifax CEO Richard Smith testified before Congress, and is expected to appear before the Senate in more hearings throughout 2018. By late October 2017, Equifax was at least partially saved from more legal woes by the Senate's repeal of a rule that allowed consumers to file class-action suits against banks and other financial institutions, forcing Equifax victims to go the route of private arbitration.

Small financial institutions have also filed multiple class action lawsuits against Equifax to recover financial harms related to the breach. The first of those has been by Summit Credit Union, followed by Bank of Louisiana, Aventa Credit Union, and First Choice Federal Credit Union lawsuits claiming that Equifax had violated federal law.

CHAPTER 03

DETECTION & PATCHING

Apache Struts 2 - The Popular Kid

Apache Struts 2 is an open source web application framework used to create applications in Java. Its widespread use powers both front and back end applications. A top-level Apache project (initially started under the Jakarta project), Struts is a model 2 framework that allows developers to bundle up multiple technologies for building, extending, and maintaining Java web applications. Struts falls under the Apache Software License (ASF), serving as the free framework of choice for Java developers across the world.

With more than 1,500 repositories on GitHub, hundreds of commits, and an abundance of versions, it is estimated that at least 65% of Fortune 100 companies actively use applications built with the Apache Struts framework.

Apache Struts has a notably vibrant community of contributors and owes its popularity to the enterprise quality code that its community produces. As compared with competing open source Java frameworks such as Spring MVC, JSF, or Hibernate, Struts is known for its high functionality and the ease of integrating its components.

Struts 2's eyeball advantage that stems from its community of contributors, means that vulnerabilities more readily rise to the surface and fixes are more swiftly and effectively made available.

Simply, bug detection is a labor intensive and ongoing process. As code is developed and revised, as extensions and improvements are made, coding mistakes are inevitably made as well. The best way for vulnerabilities in code to be detected, is to have multiple developers engaging with the code on an ongoing basis. For this to be possible, an active, contributing community, like the one responsible for Apache Struts, must be in place.

[3] According to Apache Foundation projects database, Struts 2 comes in at number 35 out of 190 active projects in measures of popularity and number of committers.

Many Eyes Make All Bugs Shallow

The claim made famous by Linux Kernel's principal developer Linus Torvalds rings true in the case of Apache Struts too.

Code developed by the Struts community enjoys the benefits of a large enough developer base that flaws are picked up on and fixes offered quickly and efficiently. That is the great advantage, argued Torvalds, of multiple developers and contributors to a single project. It is, in other words, the advantage of redundancy and overlap.

For Struts 2 users, all that was left to do was remain attuned to news coming from this bustling community. Once adopted, it was in the hands of the developers using the framework to keep track of their use of the Struts components, be aware of vulnerability updates as these came to light, and be ready to integrate fixes in real-time.

Developers using the popular Apache framework needed to be aware that as its popularity rises so does its risk of attack. As seen in the case of the CVE-2017-5638, detection of the vulnerability and its publication led to a dramatic rise in hits (from 200 hits before publication to 1600 just days after the vulnerability went public).

Vulnerability Detection

The Apache Struts CVE-2017-5638 flaw was designated as an expression language vulnerability. The Jakarta Multipart parser in Apache Struts 2 led to a mishandling of content-type value. The signature of the vulnerability was the presence of #cmd= or #cmds= strings in the Content-Type, Content-Disposition, or Content-Length HTTP headers.

To exploit it, all the attacker needed to do was send an HTTP request containing a particular syntax expression to the server. The Equifax Struts 2 application received the request and granted the perpetrator access to all operating system commands. This is a well known vulnerability, easily exploitable and the magnitude of damage it causes can be severe.

In layman terms, the Apache Struts CVE-2017-5638 vulnerability allowed attackers remote execution of arbitrary commands. This is the holy grail of vulnerabilities because it provides attackers with an extensive window of opportunity to access the server ultimately achieving a privilege escalation breach.

Privilege escalation breaches are fairly commonplace, especially in unpatched systems. They allow hackers to execute commands remotely, leading to complete control of the entire server.

The Fix

Most vulnerability fixes require either downloading and installing a patch or updating a version. In the case of the Apache Struts CVE-2017-5638, the Apache Foundation issued a patch to remedy the vulnerability.

The fix required that each web application that was developed with a vulnerable version of Apache Struts 2 be recompiled using a patched version. All versions of Apache Struts since 2008 were deemed affected (Struts 2.1.2 – Struts 2.3.33, Struts 2.5 – Struts 2.5.12) and all web applications using the framework's REST plugin were said to be vulnerable as well.

That placed Equifax's code firmly inside the brackets of what was clearly defined as "at risk." Like so many other companies, Equifax had made use of the vulnerable Struts 2 components and, like too many other companies, it was unaware that it had done so.

Given the labor intensive work involved in patching up the CVE-2017-5638 vulnerability, it would have earned Equifax brownie points had the company been actively working on applying the March 7 patch before they got hit and could argue they just didn't get it rolled out soon enough. However, the company clearly stated they only noticed a problem in their system in July, four months after the Apache Foundation went public with the vulnerability and a patch was made available. There is no running away from the fact that Equifax was simply not aware of the vulnerability in its system until after it was compromised.

So What Now?

The question on the minds of many following the 2017 breach has been whether vulnerabilities in Apache Struts make it too risky a framework for widespread use, suggesting that migration to alternative frameworks is needed.

The simple answer to this question a resounding no. Primarily because, as any developer using open source components knows, the potential for bugs is part and parcel of all open source usage. Period. Of any project, in any library, in any component.

Secondly, because Apache Struts' eyeball advantage makes it a safer rather than riskier choice in terms of the potential for vulnerability non-detection. Going back to the arguments made earlier with regard to Struts' popularity amongst developers, the more a piece of code is looked over, the more attention it receives from multiple contributors, the greater the likelihood for vulnerabilities to rise to the surface, and the larger the pool of developers to offer a fix.

As the Apache Foundation said in a statement immediately following the Equifax breach "Most breaches we become aware of are caused by failure to update software components that are known to be vulnerable for months or even years."

The Equifax vulnerability was uncovered in March 2017 and a patch to remedy the situation was issued that same day. That Equifax took until September of 2017 to patch up their system merely exemplifies how the credit agency was just another one in a long line of institutions to fall prey to the same issue the Apache community deemed their most recurring problem; a negligent attitude to vulnerability warnings and a lack of visibility of the components in one's system.

CHAPTER 04

DOWN IN FLAMES; WHAT WENT WRONG WITH EQUIFAX?

Slow response rate. Lack of diligence on the part of Equifax's development and security teams. The 2017 breach was an avoidable one. Not by migrating to other platforms. Not by adopting components that had less known vulnerabilities, but by staying on top of the open source components that were being used in their environment for their products.

The problem was made well known in the public arena, it was known to any developer who took the time to keep track and know that they were using Apache Struts in their code. And the remedy was made available. The problem was also a hot commodity for hackers who were actively attacking Struts components in any system they could find them, working against the clock to infiltrate as many systems as they could before the patching was implemented.

To that end, Equifax is far from unique. In many ways, the credit reporting agency "took one for the team". The open source users team, that is. The data breach that happened in their systems could have happened, and in fact has happened, to many companies in the past for the same reason. Failure to properly manage the open source components in use, and lack of clear open source security measures. It will happen again in the future too.

Why Was Equifax Using Open Source?

In today's agile, performance-driven, world software companies are under ongoing pressure to deliver quality products in record time. In an effort to meet the ever-growing pressures of the marketplace, developers regularly turn to open source components to cut development times.

Open source enables companies to share the burden of development, thereby speeding up production times and cutting costs. Open source has become the development norm in modern organizations and, to that end, Equifax was no different.

Responsible Vs. Negligent Open Source Usage

As companies from enterprise to startups regularly leverage open source in mission-critical aspects of their software development, two elements become absolute necessities in ensuring the safe use of open source code.

Visibility

Visibility is a company's ability to see of the open source components that it uses. To measure the amount of open source used relative to the whole of their product and map out its location in the codebase.

Developers are often oblivious to the number of open source libraries they are using. They often do not "count" smaller libraries, seemingly trivial components that serve to activate connective functions in the code. Developers are also often unaware of the dependencies that exist between code layers, between the open source components they introduce and the greater product they are working on (which includes proprietary code and possibly also third party commercial code).

They certainly do not think about their use of transient dependency management tools like Maven (Java), Bower (JavaScript), or Bundler (Ruby) that pull in third party dependencies automatically.

If such uses are not readily marked under the title of 'open source usage', it is clear why companies cannot document their open source usage manually. Simply, true visibility cannot be arrived at manually. Not at this level. Not with thousands of pieces of open source components dispersed throughout the codebase. This level of visibility necessitates an automated documentation system.

Connectivity

Having a detailed account of the open source components in a system is half the battle, but it is not enough. To ensure vulnerability detection happens in real-time and fixes are implemented at the speed they are released, a steady, ongoing feed from the open source community is needed. Which brings us back to the failure of manual tracking and the need for automation.

Only an automated system is able to send and receive data from the open source ecosystem in real-time, with frequent updates and consistent cooperation.

Open Source Vs. Proprietary Code

Open source and proprietary code differ not only in their production methods but also in the security measures taken to protect them.

While proprietary code is often deemed the “special sauce” of a company, the unique selling point which the company goes to great lengths to secure and safeguard from prying eyes, open source is the “imported step-child”, collaboratively and openly developed and arguably requiring little security measures be invested in its maintenance.

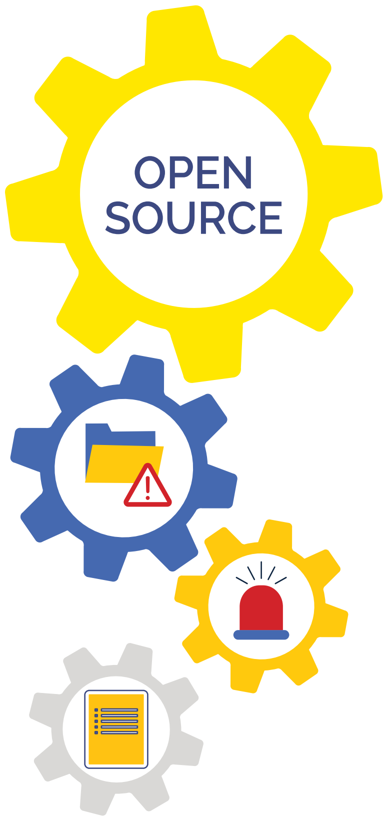
Despite this perception, reality reflects a far different situation. Open source makes up 60% to 80% of the codebase used in most companies. Inbred step-child or not, it accounts for the majority of a company’s mission-critical functionality. It seems a grave imbalance, therefore, that little efforts are directed towards open source security and that governance processes regarding open source use are not properly enforced.

Even if companies are taking steps to keep their special sauce in-house code safe with tools aimed at finding vulnerabilities, if they are not directly addressing their open source, then they are totally missing the mark.

Therein lay the problem that cost Equifax hundreds of millions in breach related costs. It is a liability that continues to loom large for all companies that make open source use their norm.

CHAPTER 05

OPEN SOURCE USERS, KNOW THIS



Open source is a powerful force multiplier for development teams, helping them to succeed in their mission. However, we know that it has its own rules and requires different treatment than proprietary code.

Use this checklist to use open source components confidently and avoid some of the common pitfalls.

Establish a Security Policy that Specifically Addresses Open Source

For starters, unlike proprietary code, open source code enjoys little to no security testing. Far too many organizations have invested in securing their in-house code but have failed to implement protections for open source.

A Ponemon Institute study from 2016 found that only 36% of those surveyed reported performing any kind of security testing on their open source components. Most organizations are only now beginning to invest in reliable means of vulnerability detection.

Consumers do not care if a breach is caused by a vulnerability in the proprietary code or in an open source component. Whatever code a company uses, it is expected to have a policy in place to protect customers' data.

Utilize an Automated Centralized Reporting System

Open source's pluralistic nature makes it so that there is no centralized authority, no single listing, wherein all vulnerabilities are reported. MITRE, the non-profit government backed corporation that operates the Common Vulnerabilities and Exposures (CVE) list, is the largest and most comprehensive listing of open source vulnerabilities.

However, not all vulnerabilities are reported to MITRE. Some open source project managers send their vulnerability findings to project specific security advisories like Node Security, RubySec, or Linux Security. In other cases, project managers will post vulnerabilities on their own repositories and issue trackers.

With no centralized location to report vulnerability updates from all repositories and in all projects, it becomes impossible to manually track vulnerabilities in real-time. Only an automated system can track vulnerabilities across multiple listings, in real-time, and with the ability to match fixes to flaws.

Understand the Depth of Your Dependencies

Open source components are built in a stratified form, with new components built atop existing ones to which they are linked for functionality. With companies failing to keep accurate inventory of their software dependencies, it is virtually impossible to manually detect vulnerabilities everywhere they may exist within the codebase.

Developers are usually oblivious to the number of open source libraries they are using and the kind of dependencies supported therein. They simply are not keeping proper (or in most cases any) records of their open source inventory. This is especially true if they are using dependency management tools like Maven (Java), Bower (JavaScript), or Bundler (Ruby) which pull in third party dependencies automatically.

While some vulnerabilities, such as Heartbleed, ShellShock, and the DROWNattack have garnered so much attention that development and security teams may have heard of, most bugs found in dependencies go unnoticed until a company is hit.

Know what your components are built on and protect them.

CHAPTER 06

SOFTWARE COMPOSITION ANALYSIS FOR OPEN SOURCE HEALTH

The situation can feel pretty dire if we exmine the situation merely through the prism of open source vulnerabilities. Thankfully, there are some positives to help lift our spirits. First off is the fact that 87% of known vulnerabilities have a fix. This was certainly the case with the Apache Struts 2 vulnerability that hit Equifax.

Then why aren't 87% of all known vulnerabilities being fixed? Why are preventable exploitations still rampant? To answer these questions we would need to consider the existing state of open source security.

Manual monitoring is a labor intensive, time consuming and error prone process which companies cannot sustain over time. That sentiment is often followed by the belief that, if need be, companies will be be able to monitor their open source usage manually.

Manual monitoring is an error prone and time consuming process which companies cannot sustain over time. Mainly because nobody in the organization, from executives to developers, knows exactly the amount of open source used. This reason alone is enough to argue the necessity of an automated monitoring policy. But even if companies are adamant about their ability to manually document their open source usage, any attempt to manually sync up a company's open source portfolio with all issue tracking databases in an attempt to shore up vulnerabilities in real-time, is an endeavour doomed to fail.

It simply cannot be done manually. Not in terms of breadth of data. Not in terms of depth of data. Not in terms of real-time remediation. This effort can, possibly, be done one a one-time basis but it is not a scalable model. Especially in agile markets, where time to market is short and product quality is high, there is no way to manually and independently monitor open source usage.

Enter Software Composition Analysis

A tool for automating open source monitoring, software composition analysis (SCA) tracks all open source components in the code. SCA tools bring to the fore vulnerabilities as soon as they are made public in any open source project. Only automated SCA tools with direct and ongoing monitoring of all open source vulnerability listings will be able to alert of new vulnerabilities in real-time, and match up fixes as these are made public.

SCA provides valuable data to executives, development, security and legal teams with the capability to generate inventory reports for visibility. Providing users with much needed visibility, SCA tools take the guessing out of the open source game.

A recent study presented in the Gartner Security & Risk Management Summit 2017, estimates that 20% to 50% of the open source community already implements SCA tools. This number is steadily on the rise as Equifax type hacks drive home the need to take accurate inventory of open source usage. Simply put, you cannot patch what you do not know you have. Automated SCA has fast become the industry standard for code security, offering a four-fold solution that covers all risk-taking aspects of the production lifecycle:

Checking Open Source Components Before Use

01

SCA aids developers in selecting a non-vulnerable, license-compliant component from open source repositories and projects. This step can be implemented at the very earliest stages of component selection, before a programmer has made a pull request. By knowing if a given component has any pre-existing security or compliance issues, developers can cut out the time that would be otherwise spent implementing fixes down the line when it becomes far more costly to remediate.

Visibility of Open Source Components Already In Use

02

The way to get the full picture on what is in the code is by exposing the deepest layers of its makeup. As discussed above, dependencies between open source components in the code and other code pieces is a major culprit in developers not knowing the extent of their open source usage. SCA offers the kind of deep-dive needed to ensure developers and company executives have a good handle on the amount and placement of open source components in their code.

Visibility of Open Source Components Already In Use

03

If knowing what you are using is the first rule of open source usage, then standardizing what you allow into your products and what you want to keep out is the next logical step. This means using SCA tools for enforcing security policies across your Software Development Lifecycle, attaining a level of governance that would be unachievable without the ability to automate. It is the next stage in truly owning your product.
SCA tools allow for this customization, enabling users to determine what level of vulnerabilities they permit use of and what level of vulnerabilities cannot be introduced into their environment. This is true for licenses too. SCA tools can be set to allow usage of certain licensed components and not others based on the governance policy a company sets for itself.

Alerting When New Vulnerabilities Hit

04

SCA was the answer to the Equifax case. Had an SCA tool been in place, Equifax developers would have been made aware of the Struts 2 vulnerability when it was first published. They would have also been offered the patch to fix the vulnerability when it was made available that same day.

A perfect example of how an automated software composition analysis could have avoided a breach and spared the 147.9 million records that were compromised.

CHAPTER 07

CONCLUSION: EVERYONE WAS DOING IT, EQUIFAX WAS JUST THE ONE TO GET CAUGHT

In the aftermath of the data breach, Equifax's missteps are laid bare for everyone to probe at and criticize. It is certainly a lesson learned, and the ease with which it happened has hopefully taught onlooking companies something about the heavy price paid for saving a few pennies on code visibility.

The 2017 Ponemon Institute's Cost of Data Breach Study[3] has found that, the current cost of a single data breach is estimated at $3.62 million. On a more granular level, the average cost of a stolen record containing PII comes out to $141 per capita. With the study finding that breaches are compromising 24,000 records on average and are continually on the rise, the costs involved for companies can be expected to climb in the coming years.

With breach proportions and effects ever increasing (the Equifax case representing a landmark case that sets it apart from most breaches in scale and magnitude), directly correlating to the rise in open source usage, what is becoming clear is that neither the use of open source nor the hacking attempts that follow will die down anytime soon. On the contrary. As time to market is cut down even further and companies are expected to produce quality products in record time, the reliance on collaborative third party components only increases. Hand in hand with that comes the growing need to safeguard products with smarter, more secure open source component usage. On the heels of the Equifax breach, the Apache Software Foundation offered some advice to businesses using Apache Struts 2 and, really, any open source library. It was a refresher course of best practices and a reminder to err on the side of caution.

The salient points were these:

- Know which supporting frameworks and libraries are used in your software. Also make sure you know which versions you are using.
- Keep track of security announcements and vulnerability updates affecting components you are using. In this case too, be version compliant.
- Don't sit on it. Roll out security fixes as quickly as possible.

To these points we would add:

- Educate your C-level execs. More often than not, the C suite has strikingly little knowledge of the extent of open source use in their development cycles.
- Continually be on the lookout for suspicious activity that could point to lateral movement or exfiltration of large amounts of data.
- Monitor for unusual access patterns.
- Make security layering part of your engineering blueprint.
- Make encryption a priority
- Invest in security audits
- Use segmentation so that when a breach does occur, that the scofflaws will only have a more limited trove of stolen goods

[3]Data derived from IBM® sponsored 12th annual Cost of Data Breach Study, conducted by Ponemon Institute. .