CONTRAST SECURITY

STRUTS, OSS & YOU

What we learned from Equifax & what you can do now

October 25, 2017

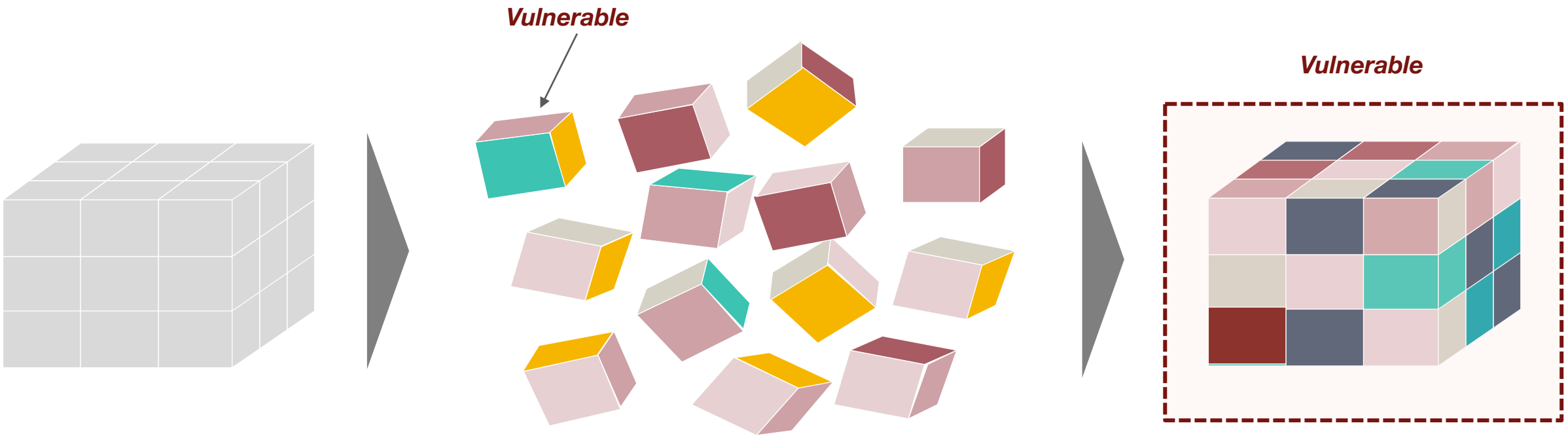
AGENDA

- The reality of software today (10 minutes)
- Struts 2 & Equifax – a case study (15 minutes)
- Best practices - remediation & secure development (15 minutes)
- A primer on runtime protection (10 minutes)

WHO'S UP HERE?

PROBLEM

THE REALITY OF SOFTWARE TODAY



Your software:
Built by you

Third party building blocks

Your software:
Assembled

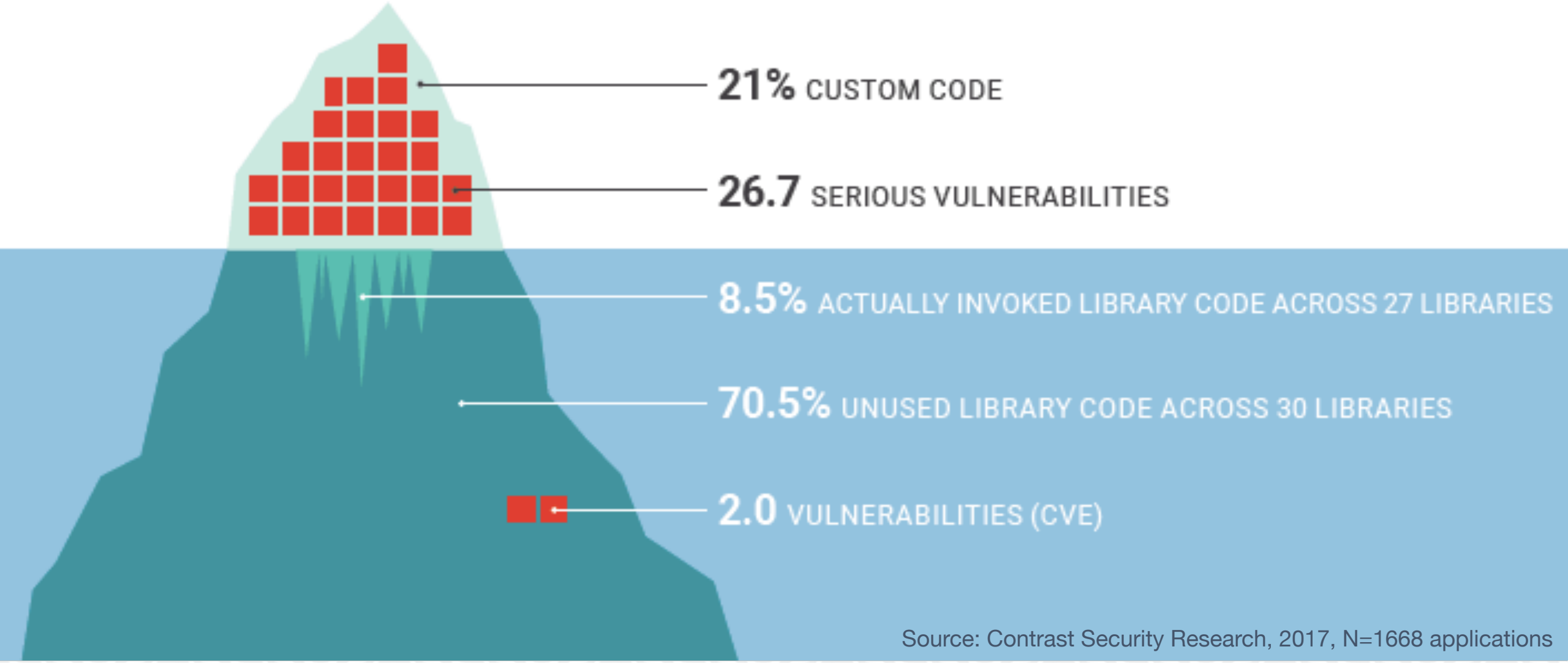
Vulnerable components = exposed software = higher risk

BUSINESS & SECURITY LOSE AS A RESULT

Option 1: Wait for updates or re-write software

Option 2: Accept the risk and launch vulnerable software

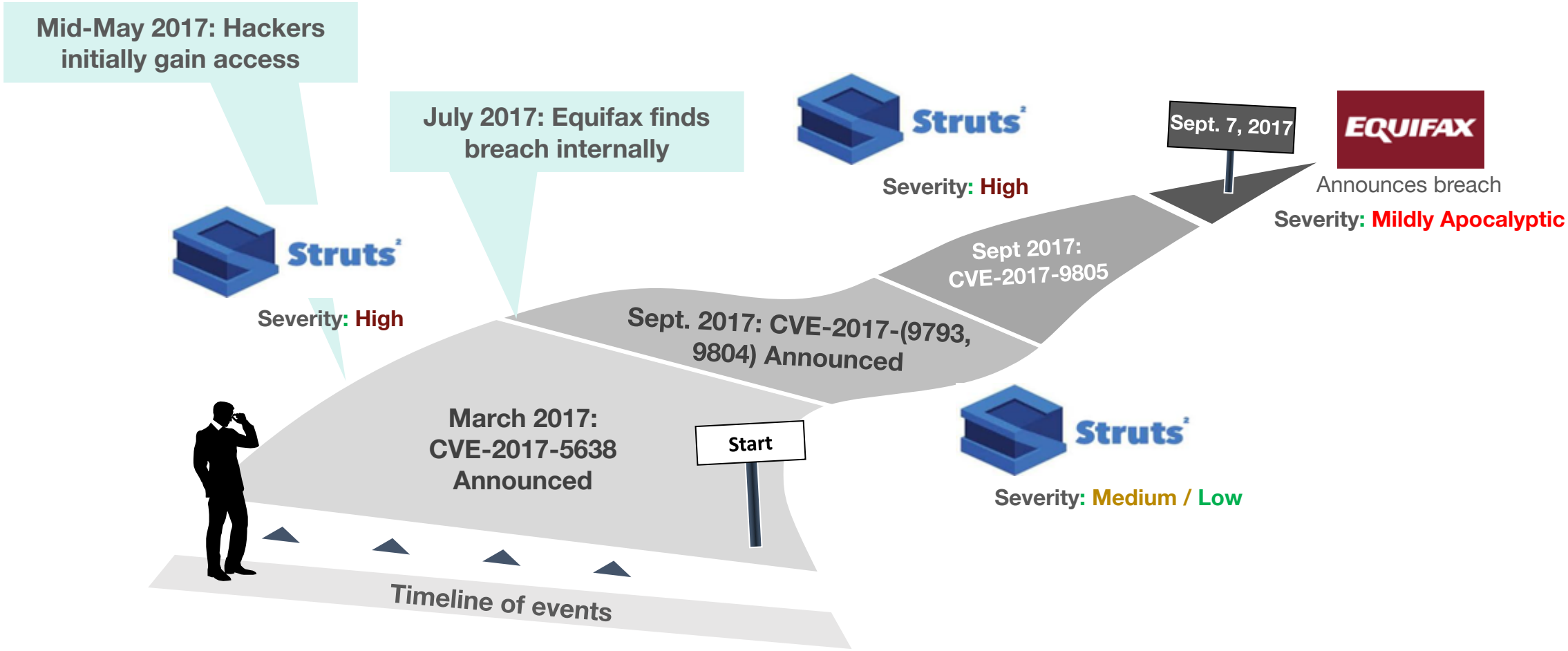
THE "AVERAGE" APPLICATION



Source: Contrast Security Research, 2017, N=1668 applications

THE EQUIFAX CASE STUDY

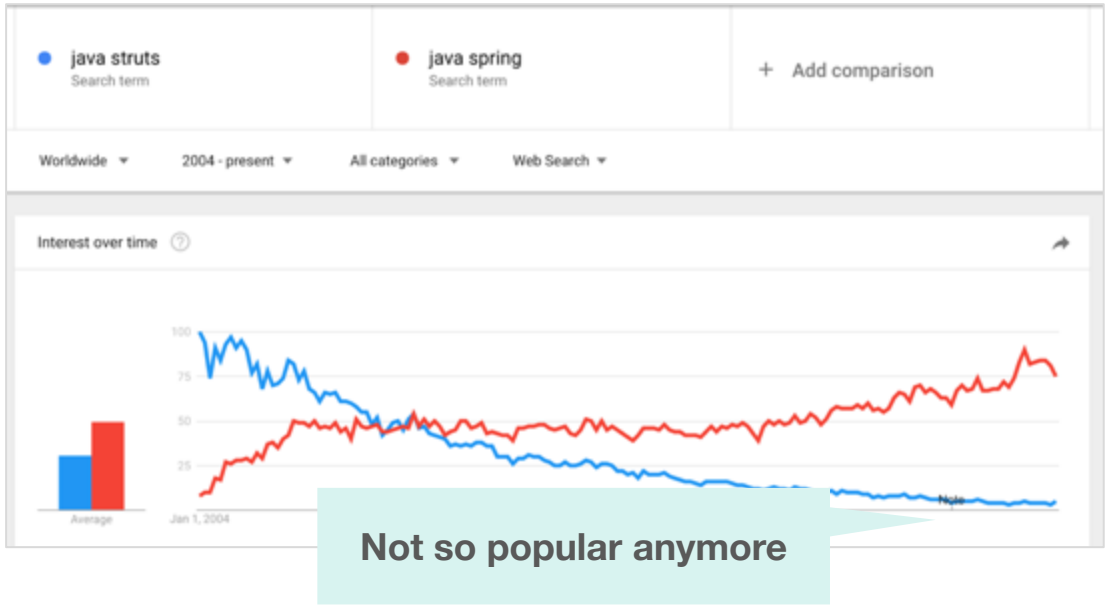
THE (INCREDIBLE) YEAR IN APPLICATION SECURITY



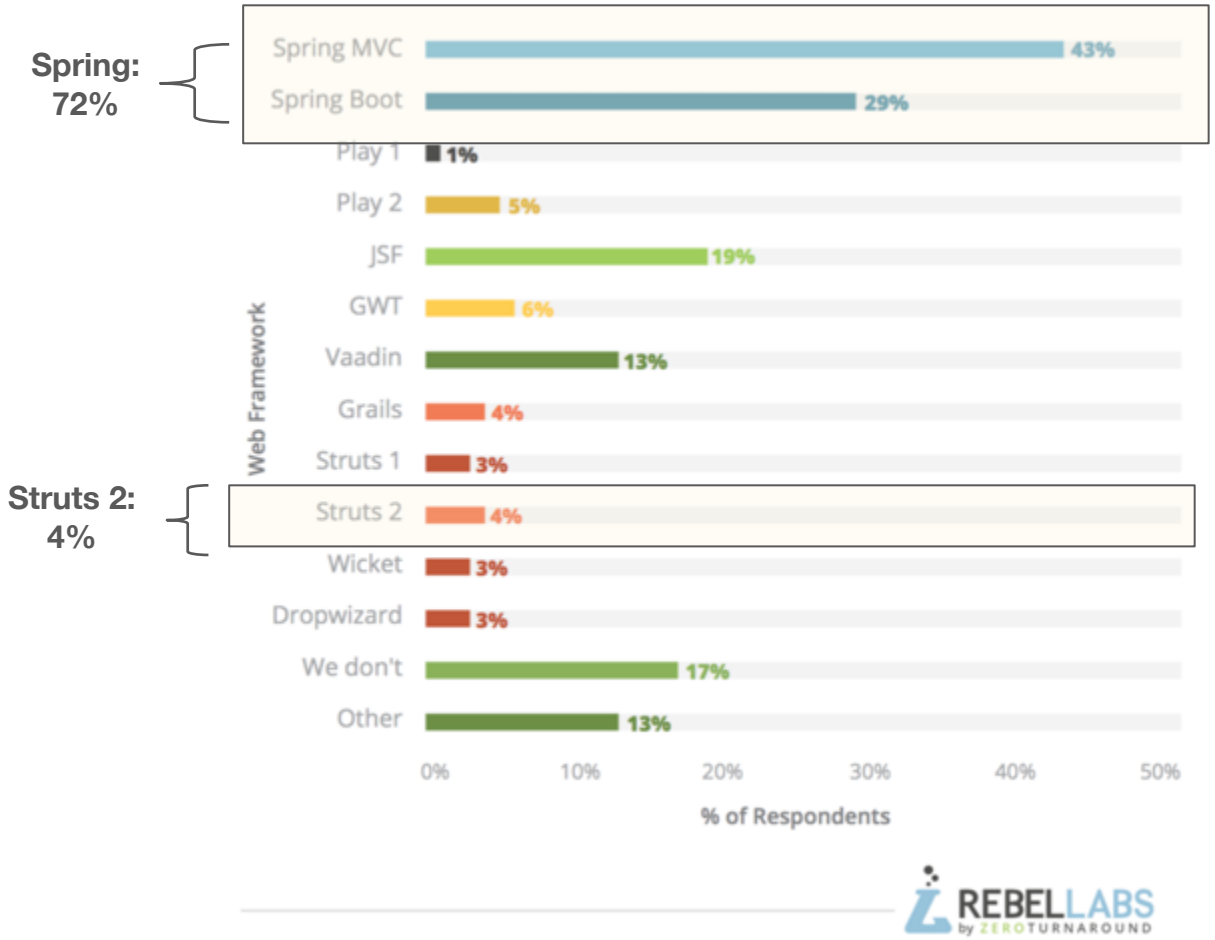
THE SOURCE: STRUTS 2- A WIDELY USED COMPONENT

Apache Struts 2

- Open-source web application framework
- Used to develop Java web applications
- History of security failures resulting from centering around expression language



Usage of Popular JAVA Frameworks



WHAT DO WE KNOW ABOUT THE VULNERABILITIES?

	2017-5638		2017-9793	2017-9804	2017-9805	2017-12611
Severity	**High**		**Medium**	**Low**	**High**	**High**
Type	Remote code execution	**6 months pass**	Denial of Service	Denial of Service	Remote code execution	Remote code execution
Target	Jakarta multi-part parser		Struts 2 URL Validator	Struts 2 URL Validator	Struts 2 REST plugin	Freemarker plugin
Cause	Multi-parser mishandles file upload Via a #cmd= string in a crafted Content-Type HTTP header		Long-running regex Via specially crafted URL	Fix for incompleteness of CVE-2017-9793 fix Via specially crafted URL	Struts 2 deserializes user input unsafely Via unconfigured XStream	Struts 2 mistakenly evaluates user input Via parameters or headers, most likely

Source of Equifax breach

STRUTS2 DEPENDENCIES

compile

The following is a list of compile dependencies for this project. These dep

GroupId	ArtifactId	Version
commons-fileupload	commons-fileupload	1.3.3
commons-io	commons-io	2.5
ognl	ognl	3.1.15
org.apache.commons	commons-lang3	3.6
org.apache.logging.log4j	log4j-api	2.8.2
org.freemarker	freemarker	2.3.23
cglib	cglib-nodep	2.1_3
commons-logging	commons-logging	1.1.3
junit	junit	4.12
org.apache.struts	struts-annotations	1.0.6
org.apache.velocity	velocity	1.7
org.apache.velocity	velocity-tools	2.0
org.slf4j	slf4j-api	1.7.12
org.slf4j	slf4j-simple	1.7.12
org.springframework	spring-aop	4.1.6.RELEASE
org.springframework	spring-aspects	4.1.6.RELEASE
org.springframework	spring-beans	4.1.6.RELEASE
org.springframework	spring-context	4.1.6.RELEASE
org.springframework	spring-context-support	4.1.6.RELEASE
org.springframework	spring-core	4.1.6.RELEASE
org.springframework	spring-web	4.1.6.RELEASE
org.testng	testng	5.14.10

test

The following is a list of test dependencies for this project. These dependencies a application:

GroupId	ArtifactId	Version
commons-validator	commons-validator	1.5.1
javax.servlet	javax.servlet-api	3.1.0
jmock	jmock	1.2.0
mockobjects	mockobjects-alt-jdk1.3	0.09
mockobjects	mockobjects-alt-jdk1.3-j2ee1.3	0.09
mockobjects	mockobjects-core	0.09
mockobjects	mockobjects-jdk1.3	0.09
mockobjects	mockobjects-jdk1.3-j2ee1.3	0.09
org.apache.commons	commons-collections4	4.1
org.apache.logging.log4j	log4j-core	2.8.2
org.easymock	easymock	3.4
org.easytesting	fest-assert	1.4
org.mockito	mockito-all	1.9.5
org.springframework	spring-test	4.1.6.RELEASE

transitive

The following is a list of compile dependencies for this project. These depend

GroupId	ArtifactId	Version	Type
commons-beanutils	commons-beanutils	1.9.2	jar
commons-collections	commons-collections	3.2.2	jar
commons-digester	commons-digester	2.1	jar
org.javassist	javassist	3.20.0-GA	jar
antlr	antlr	2.7.2	jar
aopalliance	aopalliance	1.0	jar
com.beust	jcommander	1.12	jar
commons-chain	commons-chain	1.1	jar
commons-lang	commons-lang	2.4	jar
dom4j	dom4j	1.1	jar
org.apache.struts	struts-core	1.3.8	jar
org.apache.struts	struts-taglib	1.3.8	jar
org.apache.struts	struts-tiles	1.3.8	jar
org.aspectj	aspectjweaver	1.8.5	jar
org.beanshell	bsh	2.0b4	jar
org.hamcrest	hamcrest-core	1.3	jar
org.springframework	spring-expression	4.1.6.RELEASE	jar
org.yaml	snakeyaml	1.6	jar
oro	oro	2.0.8	jar
sslext	sslext	1.2-0	jar

test

The following is a list of test dependencies for this project. These dependenci application:

GroupId	ArtifactId	Version
org.easytesting	fest-util	1.1.6
org.objenesis	objenesis	2.2

IMPACT: IT'S NOT JUST YOUR APPS!

"Multiple Cisco products incorporate a version of the Apache Struts 2 package that is affected by these vulnerabilities.

This advisory will be updated as additional information becomes available."

"Oracle has stepped outside its usual quarterly security fix cycle to address the latest Apache Struts 2 vulnerability…

[…] sprawling product set meant fixes had to be deployed across more than 20 products including Siebel Apps, Oracle Communications Policy Management, 21 financial services products, the WebLogic Server, the MySQL Enterprise Monitor, and its Retail XBRi Loss Prevention software.

Source: https://tools.cisco.com/security/center/content/CiscoSecurityAdvisory/cisco-sa-20170907-struts2

WE ARE SEEING ONGOING ATTACKS

Contrast Security

Secure https://apptwo.contrastsecurity.com/Contrast/static/ng/index.html#/3c3a73d6-78a0-46c7-944a-b07b94d557f1/attacks/events

CONTRAST | Applications | Servers | Libraries | Vulnerabilities | Attacks | Search Contrast | + Add Application | Jeff

All (121) | Find Event | 03/08/2017 02:19 pm - 04/07/2017 02:19 pm | Advanced | You have filters set. | Clear

CVE-2017-5638 Event from 222.186.34.77

PROBED | When: 04/04/2017 02:03 AM | URL: /Contrast/error/404.html | + Add Exclusion

Overview | Request | Discussion

We observed an attack against CVE-2017-5638 enter the application through the HTTP Request Header "content-type":

```
GET /Contrast/error/404.html HTTP/1.0
Accept: */*
Accept-Language: zh-cn
Connection: close
Content-Type: %{(#_memberAccess=@ognl.OgnlContext@DEFAULT_MEMBER_ACCESS).(#wmres=#context['com.opensymphony.xwork2.dispatcher.HttpServlet
Response']).(#wmres.getWriter().print("S2-045 dir--***")).(#wmreq=#context.get('com.opensymphony.xwork2.dispatcher.HttpServletRequest')).
(#wmres.getWriter().println(#wmreq.getSession().getServletContext().getRealPath("/"))).(#wmres.getWriter().flush()).(#wmres.getWriter().c
lose())}.multipart/form-data
Cookie: AWSELB=539F750F10478D4E063589242269EA3B38F3BDF0DC18B0F7A35AA369BDF525DBE006E8DA108F4FC48572AD541F9C37F85D9F6382CF8E20CC1054089C77
66B93FCB079E28F15EF3BAF264DDEB64E0691CC65B16F00F;JSESSIONID=821ABF417420EEE1EEEE9AA7F0BA4640
Host: 127.0.0.1:8080
Referer: http://54.86.199.1
User-Agent: Mozilla/4.0 (compatible; MSIE 9.0; Windows NT 6.1)
X-Forwarded-For: 222.186.34.77
X-Forwarded-Host: app.contrastsecurity.com
X-Forwarded-Port: 443
X-Forwarded-Proto: https
X-Forwarded-Server: app.contrastsecurity.com
```

- China (within 24 hours of advisory!)
- India
- Russia

HOW A 3RD PARTY VULNERABILITY DESTROYED $5B

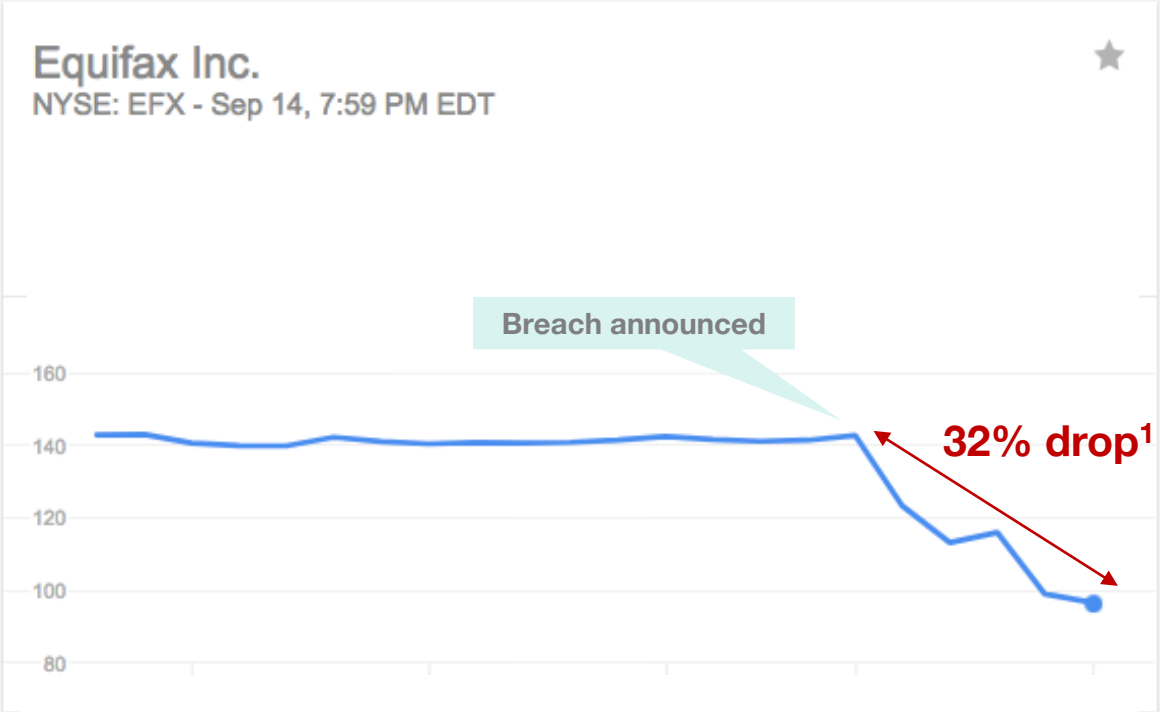
Key Facts

- **Equifax announced data breach Sept 7, 2017**
- Breach was due to a vulnerability in Apache Struts 2 (third party component)
- Apache Struts 2 CVE-2017-5638 – **March 8, 2017**

Impact on Equifax[1]

- **Customer:** 143M consumers impacted
- **Financial:** ~$5B in market cap loss ($3.4B in 2 business days)
- **Legal:** 70+ class-action lawsuits
- **Operational:** 3x increase in call center staffing, CEO & other executives terminated
- **Going forward:**
  - Slower share price growth[2]
  - Customer churn, potential regulatory fines, etc.
  - Distracted from innovation

Financial Impact of Equifax Breach: $5B market cap loss[1]



1) **Source:** Google Finance
2) **Source:** https://www.comparitech.com/blog/information-security/data-breach-share-price/

CEO TESTIMONY INDICATES EQUIFAX HAD UNREALISTIC VIEW TOWARDS APPLICATION SECURITY

Testimony from Congress

- Single person was at fault for not patching application
- Patching SLA was 48 hours for all applications
- Scanning tool did not find vulnerability

- Single point of failure
- Unrealistic policy
- Ineffective toolset

Poor understanding of how to manage application security & risk

USING TRADITIONAL TECHNIQUES, HOW WOULD EQUIFAX HAVE RESPONDED TO THE CVE?

1. Learn of the new CVE
2. Identify all the places in their software portfolio they are exposed
3. Allocate Engineering+Ops+Security resources to bring vulnerable applications down and/or put temporary workaround in place
4. Patch Applications & Re-test
5. Deploy Fixed Applications

Hackers move faster than large enterprises. This is an impossible feat. Equifax had approximately 75 days to complete all of these steps.

REMEDIATION & BEST PRACTICES

AT A MINIMUM FOR 5638..

- Upgrade your version of Struts 2
- Switch your underlying multi-part library
- Tighten up your network ACLs

WHAT YOU CAN DO NOW

	What to do
1 Understand frameworks and libraries you use	• Quickly inventory applications with vulnerable Struts 2 and dependencies • List associated vulnerabilities and prioritize • Export vulnerabilities to GRC systems
2 Quickly roll out patched software	• Update Struts 2 versions where possible • Add virtual patches in production where not possible
3 Security Policy must assume software is flawed	• Re-test all vulnerable applications - pen test for business logic, IAST for basics • Ensure coverage for vulnerabilities in security testing of upcoming releases
4 Establish security layers	• Have explicit protection against known CVEs as part of a layered defense strategy (e.g., DDoS protection, network firewall, identity & access management)
5 Establish monitoring for unusual access patterns	• Report application data flow & authentication logs to SIEM • Establish IR workflow for application attacks

Quickly identify and secure custom and third party software to reduce risk

INCORPORATE APPLICATION RISK INTO FOUNDATIONAL PROGRAMS




Program	Risk Assessment	Third Party Reviews	Incident Response & Monitoring	Identity & Access Management
Recommendation	Ensure routine application risk assessments include third party components Re-weight application risks	Require vendors to align with your application security standards	Ensure response teams and SOC have sufficient application intelligence	Establish single sign-on and multi-factor authentication on high risk applications

SECURE DEVELOPMENT – BEST PRACTICES FROM THE FIELD

	Design	Dev	Test	Production / Production with RASP
Relative Cost to Remediate	1x	6.5x	15x	100x
Residual Risks	No security requirements	Insecure libraries Poorly written custom code	No time to fix or patch – risk acceptance	Vulnerabilities not fixed Zero days
Engineering	Dedicated Security Engineer & Product Manager	Assess libraries Add vulns. to security backlog	Prioritize backlog – current vs next sprint Request runtime protection for "opens"	Deploy runtime protection for "open" vulnerabilities Business signs off on "critical" vulnerabilities only
Security	Secure Application Development Standard Architecture Review for major releases Approved secure library inventory		Approve backlogs & runtime protection requests	

A PRIMER ON RUNTIME PROTECTION

THERE ARE MULTIPLE APPROACHES TO RASP

	Overview	Implication
Filters & Plug-ins	Filter requests before they execute key functions & block signatured attacks	Perimeter/"proxy" based protection, similar to a traditional WAF. Wide language coverage
JVM Replacement	Replaces the runtime environment and hijacks calls to the underlying platform	Requires a proprietary and rigid runtime environment. Limited language coverage
Virtualization or Replication	Creates a replica of an application and learns runtime behavior	Requires learning / tuning time to ensure accuracy. Moderate language coverage
Binary Application Instrumentation	Places sensors at key junctions within the application stack to analyze behavior	Embedded into the software itself, without learning / tuning or replacement. Moderate language coverage

KEY BENEFITS OF RASP

SELF AWARE

Eliminate the need to train your protection. Immediate protection with lower costs

ACCURATE

Improved accuracy & coverage of application layer attacks

ENABLE DEVOPS

Deploy what you test, embed protection in through development into production

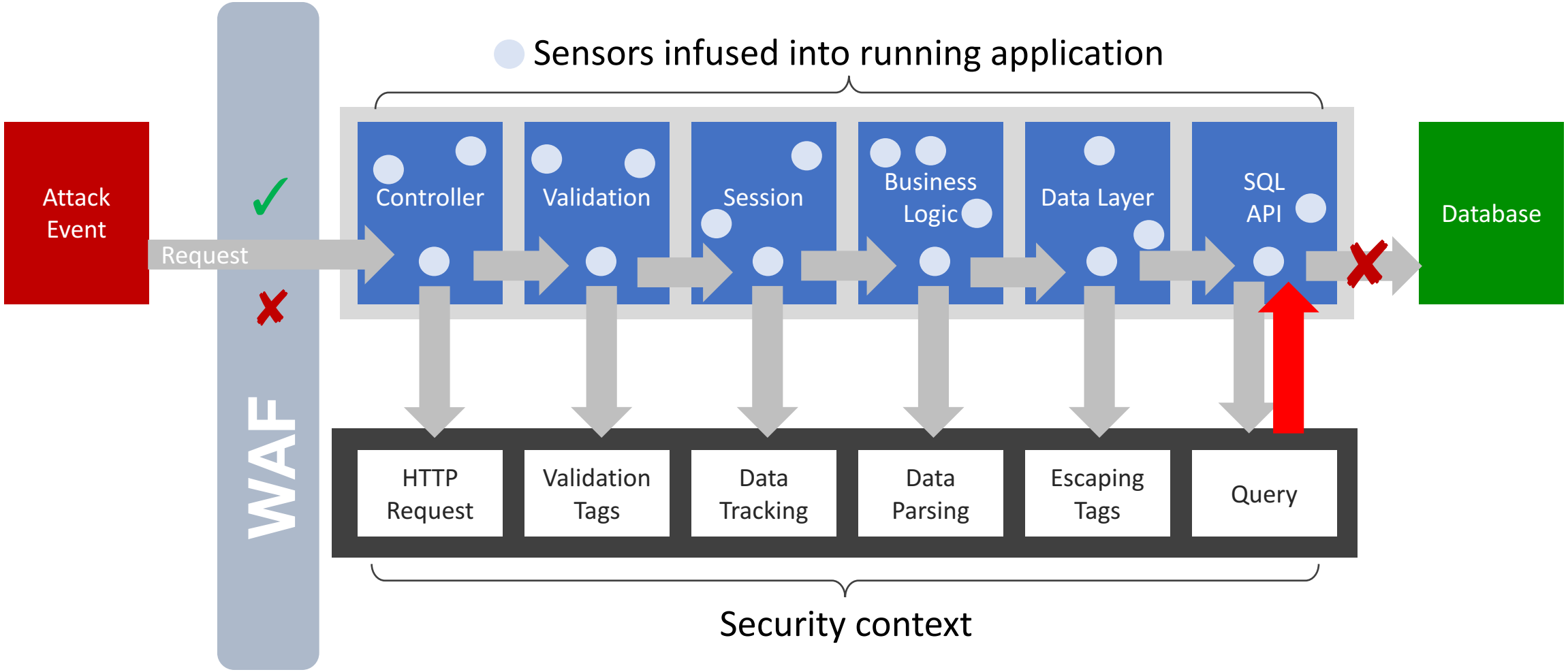
EMBEDDED

Forget the perimeter. All software is protected, regardless of where it goes

IT'S ALL ABOUT CONTEXT

RASP	WAF
• HTTP Request • Code • Data flow • Backend connections • Libraries and frameworks • Configuration • Vulnerabilities	• HTTP Request
Blocks when attack reaches a corresponding vulnerability and is about to exploit it.	**Blocks all traffic that matches attack signatures.**

HOW RASP FITS IN



WHAT TYPES OF THREATS ARE PROTECTED?

	WAF	RASP - Instrumentation
Log Enhancers	Impossible	Included
Padding Oracle	Too complex	Runtime Behavior
Complex CVEs	Too complex	Runtime Behavior
Deserialization	Too complex	Runtime Behavior
SSRF	Too complex	Runtime Behavior
CSRF	Heavy configuration	Runtime Behavior
SQL Injection	HTTP Regex	Runtime Behavior
XSS	HTTP Regex	Runtime Behavior
XXE	HTTP Regex	Runtime Behavior
Command Injection	HTTP Regex	Runtime Behavior
Path Traversal	HTTP Regex	Runtime Behavior
Simple CVEs	HTTP Regex	Runtime Behavior
DDoS	Cloud WAFs	Too Late

GOING FORWARD..

Immediate Next Steps

- Ensure the board, executive committee are aware of Application Risk
- Review & confirm you're prepared based upon Apache's recommendations
- Modernize your tool-set to address your risk as well as enable you to achieve business objectives

