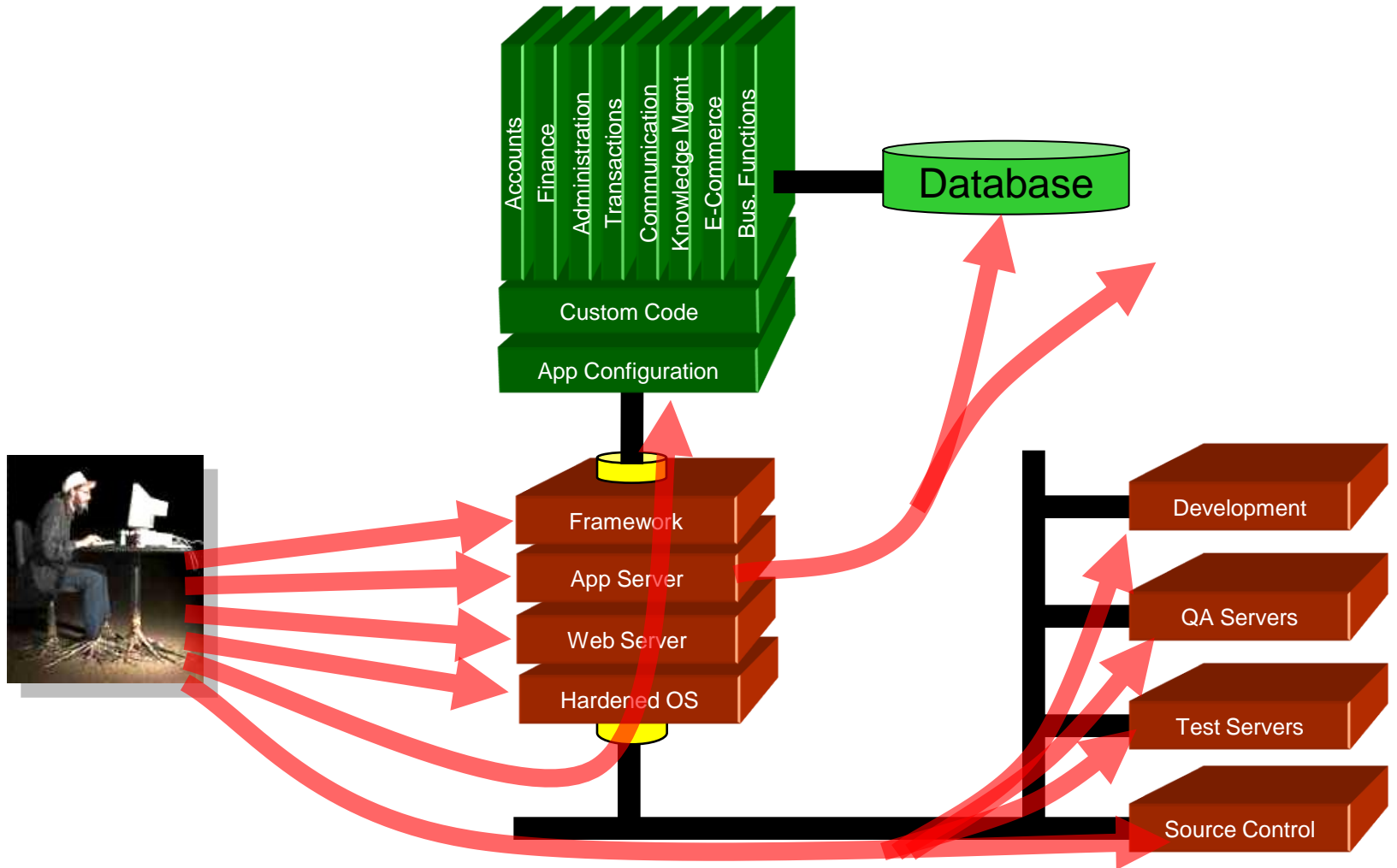


A5: Security misconfiguration

A5: Security Misconfiguration

- **Web applications must rely on a secure foundation...**
 - Everywhere from the OS up through the application server
 - Throughout its entire lifetime (from development to production)
 - Especially in the age of agile development, deployment and operations (DevOps)

Security Misconfiguration Illustrated



Examples

- **Not properly reducing privileges of services**
- **Not disabling all unnecessary functionality in OS, web framework, web application**
- **Not hardening the configuration of vulnerable frameworks (PHP)**
 - Not disabling `eval()`, `passthru()`, or `system()`
 - Not removing unused modules/plugins and minimizing dynamic extensions
 - Not hiding errors from site visitors (`display_errors`)
 - Not turning on `safe_mode`
 - Not limiting or disallowing file uploads
 - Not controlling POST size
- **Not removing credentials in source code control**
- **Not changing default credentials (Mirai)**
- **Improperly configured networking**
 - Use of deprecated TLS/SSL protocols and encryption schemes (Poodle)
 - Not enabling HSTS (HTTP Strict Transport Security)

A5-Prevention

A5 - Prevention

- **Secure configuration “hardening” guideline covering entire platform and application**
- **Automate checks of application configuration in development and deployment process**
- **Verify**
 - Scan to find any credentials improperly stored
 - Remove credentials from code repositories via SQL Safe Mode in PHP or .gitignore

HTTP's Strict-Transport-Security:

- **HTTP response header to force the use of HTTPS**

- Informs client to automatically redirect all HTTP requests to HTTPS for domain

- **Example**

```
$ curl -I http://facebook.com | head -10
HTTP/1.1 301 Moved Permanently
Location: ...
```

- Server set up to redirect HTTPS version (an improvement)
- Note, assumes response is not hijacked by adversary
- So, after redirection, use header to force client to use HTTPS in the future (to avoid MITM)

```
$ curl -I https://www.facebook.com/ | head -10
Strict-Transport-Security: ...
```

- Now, if client goes onto open WiFi, adversary can not perform MITM as client browser automatically redirects

<http://facebook.com> to <https://facebook.com>

HTTP's Strict-Transport-Security:

- How can we avoid this initial request in the first place?

```
$ curl -I facebook.com
```

```
HTTP/1.1 301 Moved Permanently
```

```
Location: https://facebook.com/
```

- Hard-coded list of domains (HSTS preload list) shipped with browser that are HTTPS only
- Check and add site to list
 - <https://hstspreload.org>

HTTP's Strict-Transport-Security:

- Configuration

- Within Apache,

- Set up redirection of unencrypted requests

```
<VirtualHost *:80>
```

```
    ServerName example.com
```

```
    Redirect permanent / https://example.com/
```

```
</VirtualHost>
```

- Set up Strict-Transport-Security header

```
<VirtualHost *:443>
```

```
    Header always set Strict-Transport-Security "max-age=63072000; includeSubdomains;"
```

```
</VirtualHost>
```

- **nginx** server {} block

```
add_header Strict-Transport-Security "max-age=63072000; includeSubdomains;";
```

HTTPS and Rogue CAs

- **Certificate Authorities (CAs) lynchpin of TLS (https)**
 - Sign certificates of sites
 - Browsers packaged with code that can validate certificates signed by each CA (several hundred)
 - Used by web browser to signal users that they can “trust” web server
 - Prevents hijacking secure connections via proxy
 - Browser detects MITM
 - Apply not only to web site, but also for all API calls (Amazon Echo hijacking via Burp Suite)

HTTPS certificate pinning issue

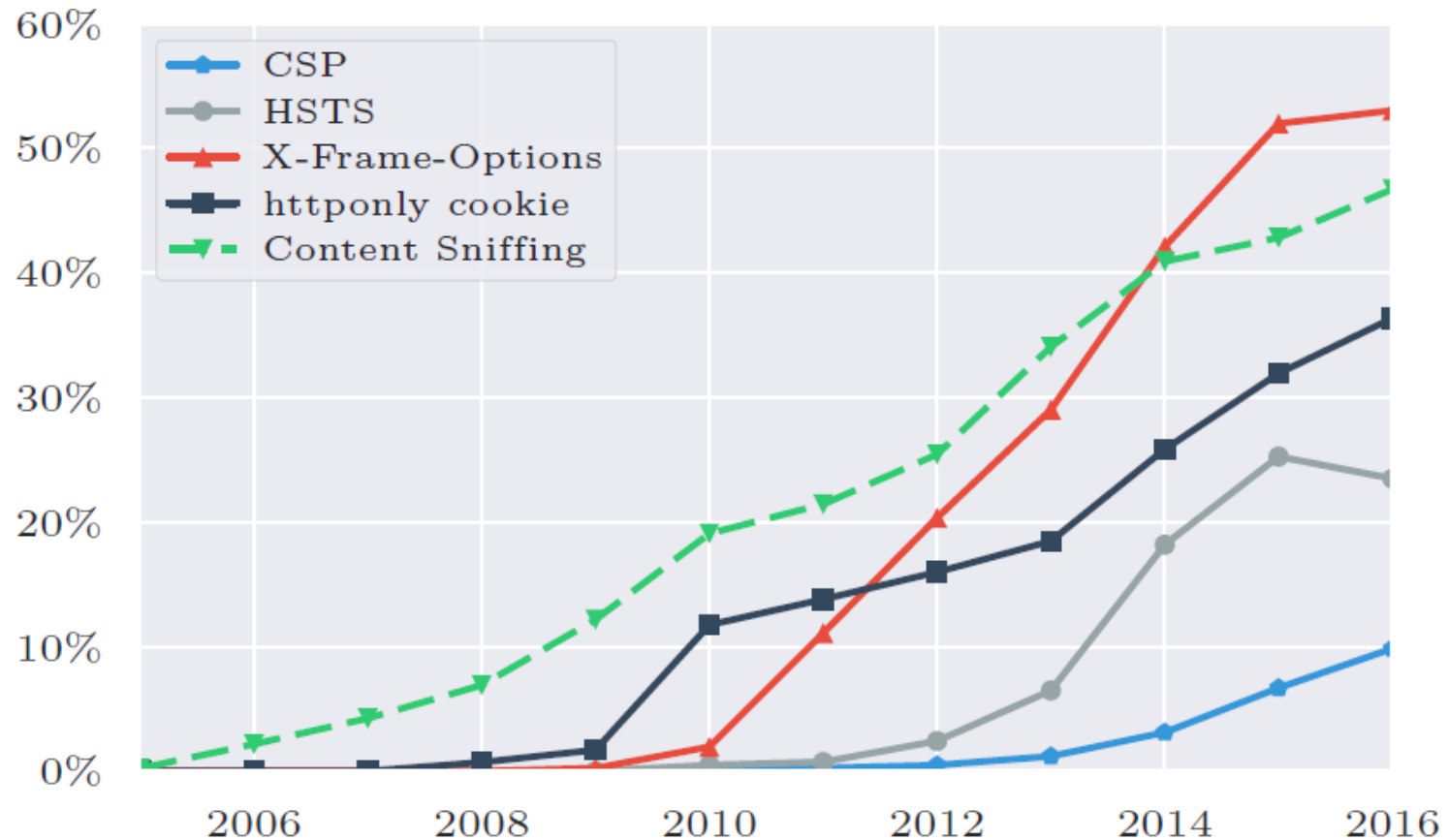
- **But...**

- Any CA can generate a valid certificate for any web site
- What happens with rogue CAs (e.g. WoSign's Github certs, Symantec test certs)?
 - Removing WoSign from browsers

- **Certificate pinning**

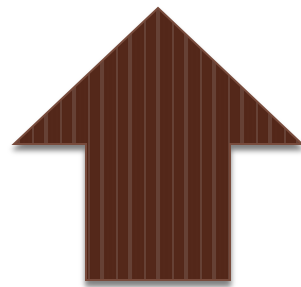
- Associate a site's certificate to a specific CA
 - Initial attempt HTTP Public-Key Pins failed
- Use TLS/SSL transparency logs to identify rogue certificates

Prevalence of usage

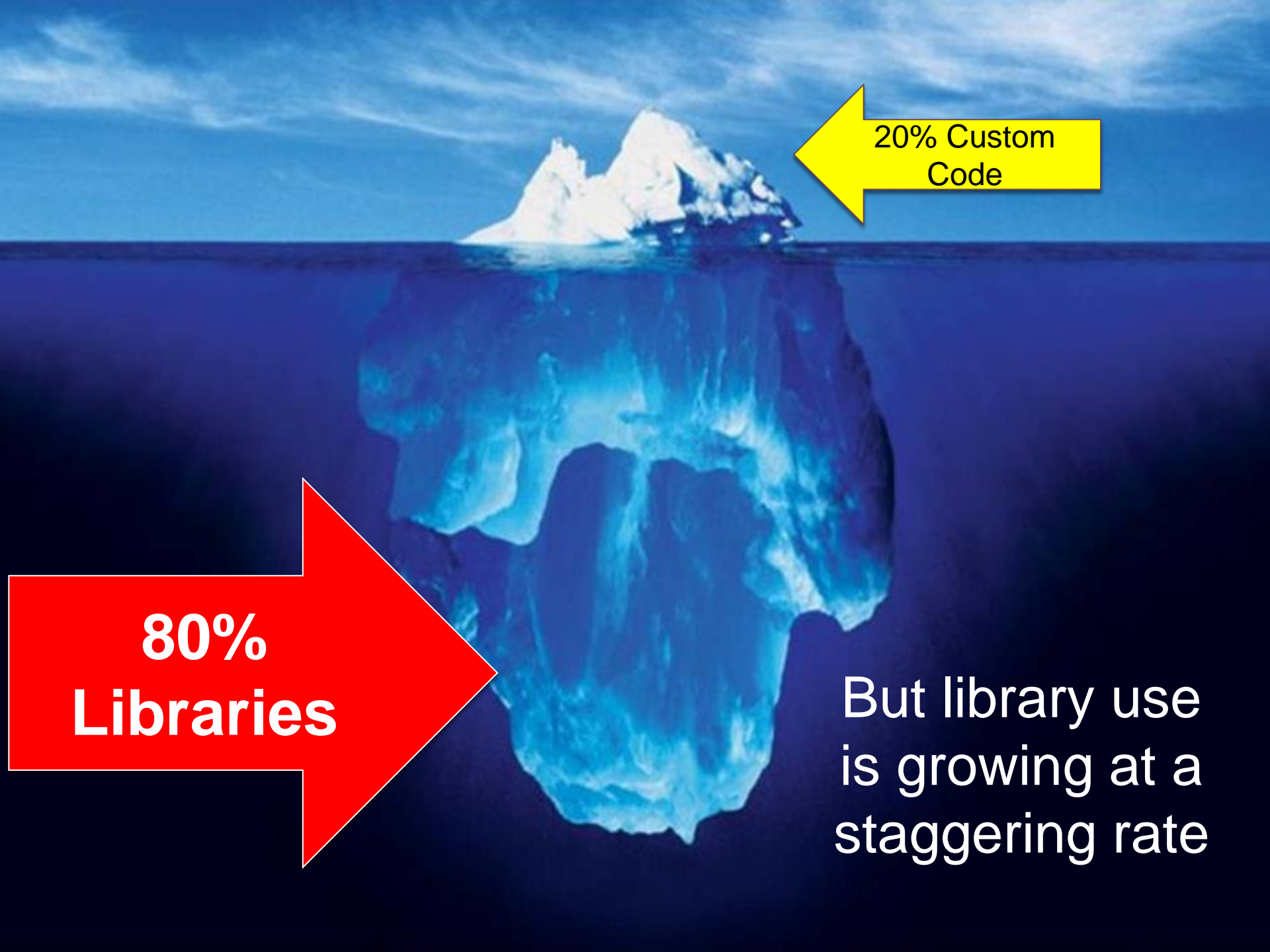


Use of Security Headers per year

A9: Using Known Vulnerable Components



The amount of custom code in an application hasn't changed very much in the past 10 years.



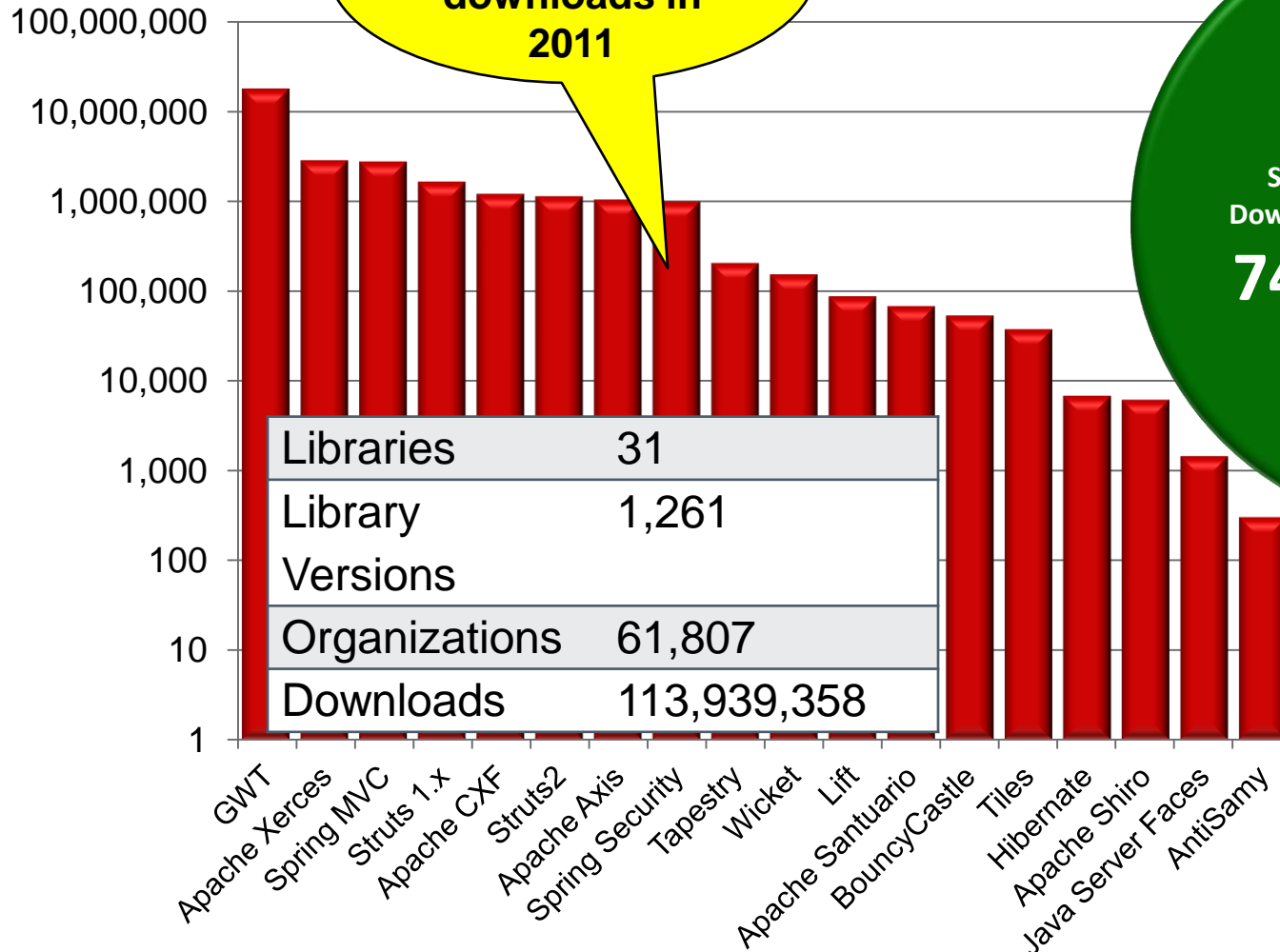
20% Custom Code

80%
Libraries

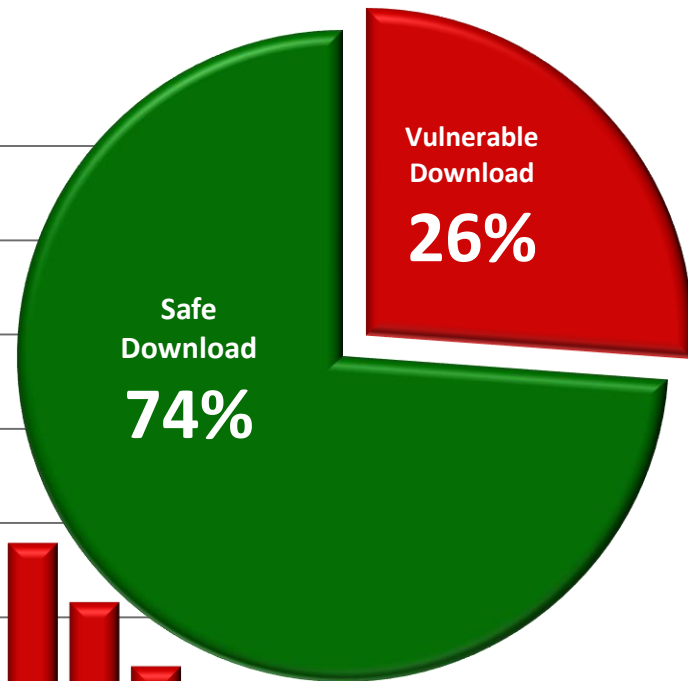
But library use is growing at a staggering rate

Everyone Uses Vulnerable Libraries

**29 MILLION
vulnerable
downloads in
2011**



Libraries	31
Library	1,261
Versions	
Organizations	61,807
Downloads	113,939,358



A9: Using Known Vulnerable Components

- **Ubiquitous problem**

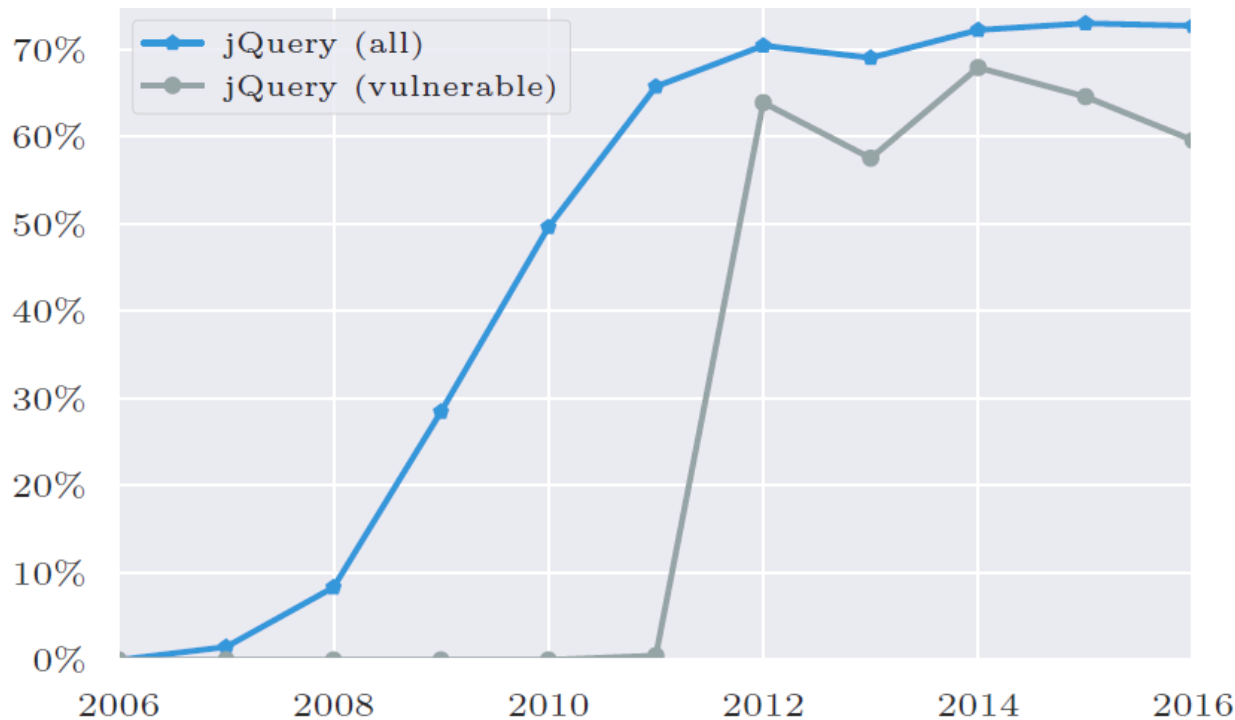
- Often identified and exploited with automated tools
- Virtually every application has them unless development teams focus on ensuring their components/libraries are up to date
 - Wherever they are located...(e.g. VMs and Containers (i.e. Docker))
- Developers often don't know all the components they are using and when they were last updated

- **Typical Impact**

- Full range of weaknesses is possible, including the rest of the OWASP Top 10

Example: jQuery

- Ubiquitous client-side Javascript library
- Often included once upon page creation, but not often updated when patches happen



jQuery usage and vulnerability statistics

Example: ImageTragick (2016)

- **Bug in ubiquitous image processing library**
 - Used in many photo and image web sites
 - Sometimes statically compiled into other code
 - Extremely difficult to update universally


Example: Tesla (2016)

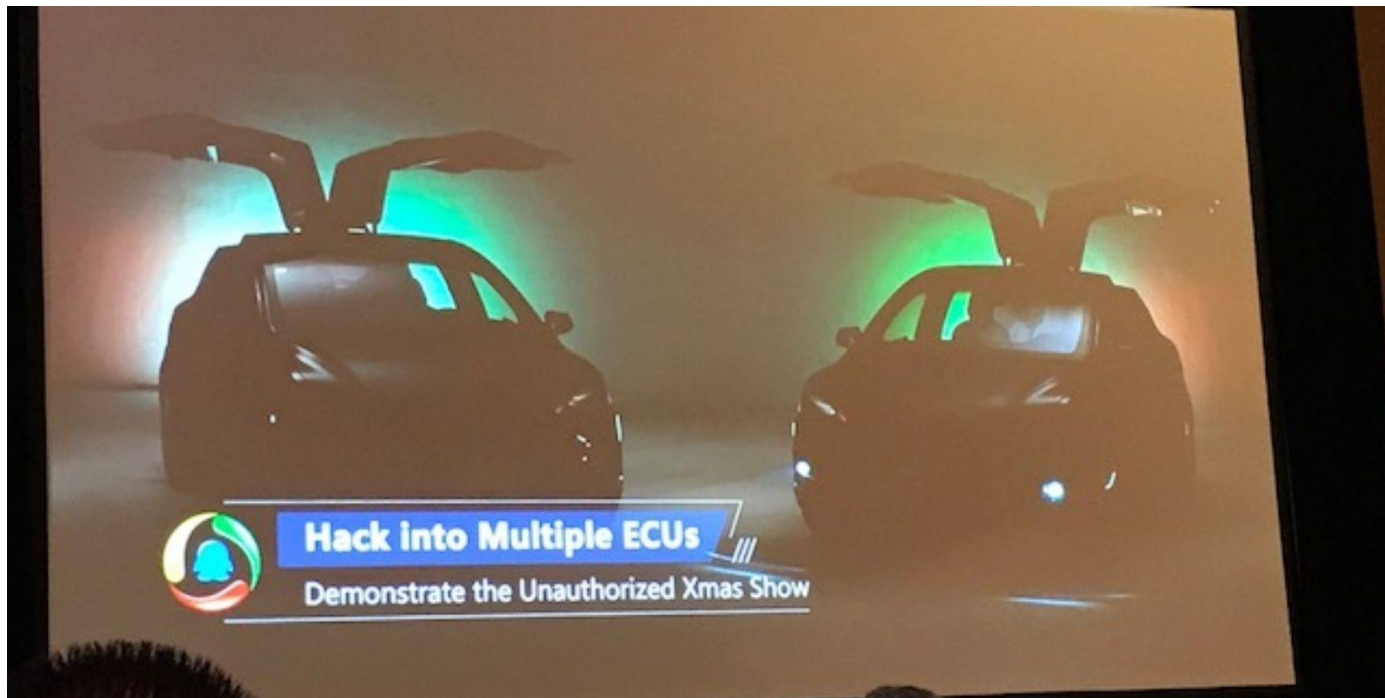
Software

Hackers hijack Tesla Model S from afar, while the cars are moving

Chinese researchers control brakes, lights and mirrors with wireless attack

By [Darren Pauli](#) 20 Sep 2016 at 04:20

62  [SHARE](#) ▼



Example: Tesla (2016)

BROWSER HACKING

Since the User Agent of Tesla web browser is "Mozilla/5.0 (X11; Linux) AppleWebKit/534.34 (KHTML, like Gecko) QtCarBrowser Safari/534.34", it can be deduced that the version of QtWebkit is around 2.2.x. **In such old version, there are many vulnerabilities in QtWebkit.** Our exploit utilizes two vulnerabilities to achieve arbitrary code execution.

The second vulnerability is CVE-2011-3928 founded by KeenTeam, which could be used for leaking memory. The POC is simple.

```
<script>if (window.addEventListener) {  
    window.addEventListener('load', func, false);  
}  
function func()  
{  
e = document.getElementById('t1');  
document.importNode(e,true);  
}  
</script>  
<table id="t1">  
    <td>  
        <xht:input>
```

Table 5 POC of CVE-2011-3928

Example: Tesla (2016)

free-fall-hacking-tesla-from-wireless-to-can-bus

4 / 16



It seems that the Linux kernel version of CID is very old, there is nearly no exploiting mitigations on Linux kernel 2.6.36.

```
Linux cid 2.6.36.3-pdk25.023-Tesla-20140430 #see_/etc/commit SMP PREEMPT 12027984  
60 armv7l GNU/Linux
```

Figure 1 CID Linux Kernel Version

Linux 2 6 36

Linux 2.6.36 released 20 October, 2010.

Summary: Linux 2.6.36 includes support for the [Tilera architecture](#), a new filesystem notification interface called fanotify, a redesign of workqueues optimized for concurrency, CIFS local caching, support for Intel Intelligent Power Sharing in i3/5 systems, integration of the kernel debugger and KMS, inclusion of the [AppArmor](#) security system, fixes for desktop unresponsiveness in some cases and several new drivers and small improvements.

Example: gSOAP (2017)



- **Bug allowing remote code execution found**
- **Library for processing XML (that many use, but don't know that they use)**
 - Used in countless IoT products *already deployed*
 - Axis surveillance cameras
 - 1 million+ downloads
 - Code and vulnerability often cloned from prior version of software
 - Code and vulnerability copied by vendor from generation to generation
 - Code often embedded in firmware that can never (or will never) be updated

Lior Div, co-founder and CEO at Cybereason, said a review of the code built into these devices shows the manufacturer does not appear to have made security a priority, and that people using these devices should simply toss them in the trash.

“There is no firmware update mechanism built into these cameras, so there’s no way to patch them,” Div said. “The version of Linux running on these devices was in some cases 14 years old, and the other code libraries on the devices are just as ancient. These devices are so hopelessly broken from a security perspective that it’s hard to really understand what’s going on in the minds of people putting them together.”

A9 - Prevention

- **Automated periodic check for out-of-date libraries**
 - Nightly build
- **Never buy a product that can't be updated**
- **Proactive upgrading**
 - Upgrade those with security issues quickly
- **Vulnerability scanning**
 - Static analysis for vulnerable source code
 - Scanning for known CVEs (vulnerabilities)
 - nessus, metasploit

OWASP Dependency Check

Run **DependencyCheck** during every build

(and do a build once a month even if nothing changed)



Page [Discussion](#)

OWASP Dependency Check

CVE	CWE	Severity (CVSS) [†]	Dependency
CVE-2011-2730	CWE-16 Configuration	High (7.5)	spring-security-core-3.0.5.RELEASE.jar
CVE-2011-2894	CWE-264 Permissions, Privileges, and Access Controls	Medium (6.8)	spring-security-core-3.0.5.RELEASE.jar
CVE-2012-5784	CWE-20 Improper Input Validation	Medium (5.8)	axis-1.2.jar [Ⓜ]
CVE-2011-2731	CWE-362 Concurrent Execution using Shared Resource with Improper Synchronization ("Race Condition")	Medium (5.1)	spring-security-core-3.0.5.RELEASE.jar
CVE-2007-6059	CWE-399 Resource Management Errors	Medium (5.0)	mail-1.4.2.jar
CVE-2007-6059	CWE-399 Resource Management Errors	Medium (5.0)	mailapi-1.4.2.jar
CVE-2012-5055	CWE-200 Information Exposure	Medium (5.0)	spring-security-core-3.0.5.RELEASE.jar
CVE-2011-2732	CWE-94 Improper Control of Generation of Code ("Code Injection")	Medium (4.3)	spring-security-core-3.0.5.RELEASE.jar
CVE-2013-0248	CWE-264 Permissions, Privileges, and Access Controls	Low (3.3)	commons-fileupload-1.2.1.jar

Java-Maven Versions Plugin

Output from the Maven Versions Plugin – Automated Analysis of Libraries' Status against Central repository

Dependencies

Status	Group Id	Artifact Id	Current Version	Scope	Classifier	Type	Next Version	Next Incremental	Next Minor	Next Major
⚠	com.fasterxml.jackson.core	jackson-annotations	2.0.4	compile		jar		2.0.5	2.1.0	
⚠	com.fasterxml.jackson.core	jackson-core	2.0.4	compile		jar		2.0.5	2.1.0	
⚠	com.fasterxml.jackson.core	jackson-databind	2.0.4	compile		jar		2.0.5	2.1.0	
⚠	com.google.guava	guava	11.0	compile		jar		11.0.1	12.0-rc1	12.0
⚠	com.ibm.icu	icu4j	49.1	compile		jar				50.1
⚠	com.theoryinpractise	halbuilder	1.0.4	compile		jar		1.0.5		
⚠	commons-codec	commons-codec	1.3	compile		jar			1.4	
✅	commons-logging	commons-logging	1.1.1	compile		jar				
⚠	joda-time	joda-time	2.0	compile		jar			2.1	
⚠	net.sf.ehcache	ehcache-core	2.5.1	compile		jar		2.5.2	2.6.0	
⚠	org.apache.httpcomponents	httpclient	4.1.2	compile		jar		4.1.3	4.2	
⚠	org.apache.httpcomponents	httpclient-cache	4.1.2	compile		jar		4.1.3	4.2	
⚠	org.apache.httpcomponents	httpcore	4.1.2	compile		jar		4.1.3	4.2	
⚠	org.jdom	jdom	1.1	compile		jar		1.1.2		2.0.0
✅	org.slf4j	slf4j-api	1.7.2	provided		jar				

Most out of Date!

Details Developer Needs

This can automatically be run EVERY TIME software is built!!

Homework

- **Security Misconfiguration (see last class's handout)**

Final project

- **From web site**

- <https://www.pentesterlab.com/exercises?only=free>
- General description and difficulty labeled
 - Range from easy levels that include walkthroughs to hard levels without guidance
- Sign-up your group today
 - No more than 2 groups per level
- MediaSpace submission
 - Most of you are now added to channel as contributors
 - Use recordmydesktop or other software to create walkthroughs

Questions

- <https://sayat.me/wu4f>

Extra

HTTP's Public-Key-Pins:

Public-Key-Pins-Report-Only:

- **NOW DEPRECATED!**
- **HTTP response header to prevent certificate hijacking**
 - For implementing HTTP Public Key Pinning (HPKP)
 - Allow website to resist impersonation by attackers using fraudulent certificates
 - `Public-Key-Pins`: enforce pin and disable request
 - `Public-Key-Pins-Report-Only`: allow request, but report it
- **Issue**
 - What if someone spoofs your DNS record, forces a victim to their bogus site, and sets a public key pin on your domain?
 - Your site is no longer reachable to victim
 - What if someone hijacks your DNS server and forces everyone to set a public key pin on your domain?
 - Your site is no longer reachable to anyone who got the pin while site was hijacked
 - Now, sites want option to disable header!
 - <https://scotthelme.co.uk/im-giving-up-on-hpkp/>

HTTP's Public-Key-Pins: Public-Key-Pins-Report-Only:

- **Now, sites want option to disable header!**
 - <https://scotthelme.co.uk/im-giving-up-on-hpkp/>

HPKP Suicide

Sadly there is a term for this and all it involves is a site making a potentially simple error. You enable HPKP, tell the browser which keys you will always use and then you lose those keys. They could be accidentally deleted, stolen in a hack or whatever, it doesn't matter. If you pin yourself to a set of keys and then no longer have the ability to use them, you're in big trouble! The most notable site I've come across that's done this is Smashing Magazine and they wrote about it in their article [Be Afraid Of HTTP Public Key Pinning \(HPKP\)](#).

RansomPKP

This is another side effect of HPKP and it's a way to use it to cause harm. In a breach scenario an attacker would gain control of your site via a server compromise or a domain hijack and then enable these headers on your behalf. When your visitors go to your site they pick up the malicious HPKP header set by the bad guys. At some point you then fix the problem and take back control of your site except now, none of the browsers will connect because of the HPKP policy they picked up from the bad guys. I have more details on this in my blog on [Using security features to do bad things](#).