CMSC414 Computer and Network Security

UI Attacks, CAPTCHAs, Security Principles

Yizheng Chen | University of Maryland
surrealyz.github.io

Feb 22, 2024

Announcement

- Project 2 released
- The deadline of Project 2 is 11:59pm ET on Friday Mar 8, with the 24 hour late deadline being 11:59pm on Mar 9.
- Don’t wait!

Agenda

- UI Attacks
- CAPTCHAs
- Security Principles

Reflected XSS vs CSRF

- Reflected XSS and CSRF both require the victim to make a request to a link
- Reflected XSS: An HTTP response contains maliciously inserted JavaScript, executed on the client side
- CSRF: A malicious HTTP request is made (containing the user's cookies), executing an effect on the server side

How to trick the users into making a HTTP request?

UI Attacks

- General theme: The attacker tricks the victim into thinking they are taking an **intended** action, when they are actually taking a **malicious** action
- Two main types of UI attacks
  - **Clickjacking**: Trick the victim into clicking on something from the attacker
  - **Phishing**: Trick the victim into sending the attacker personal information

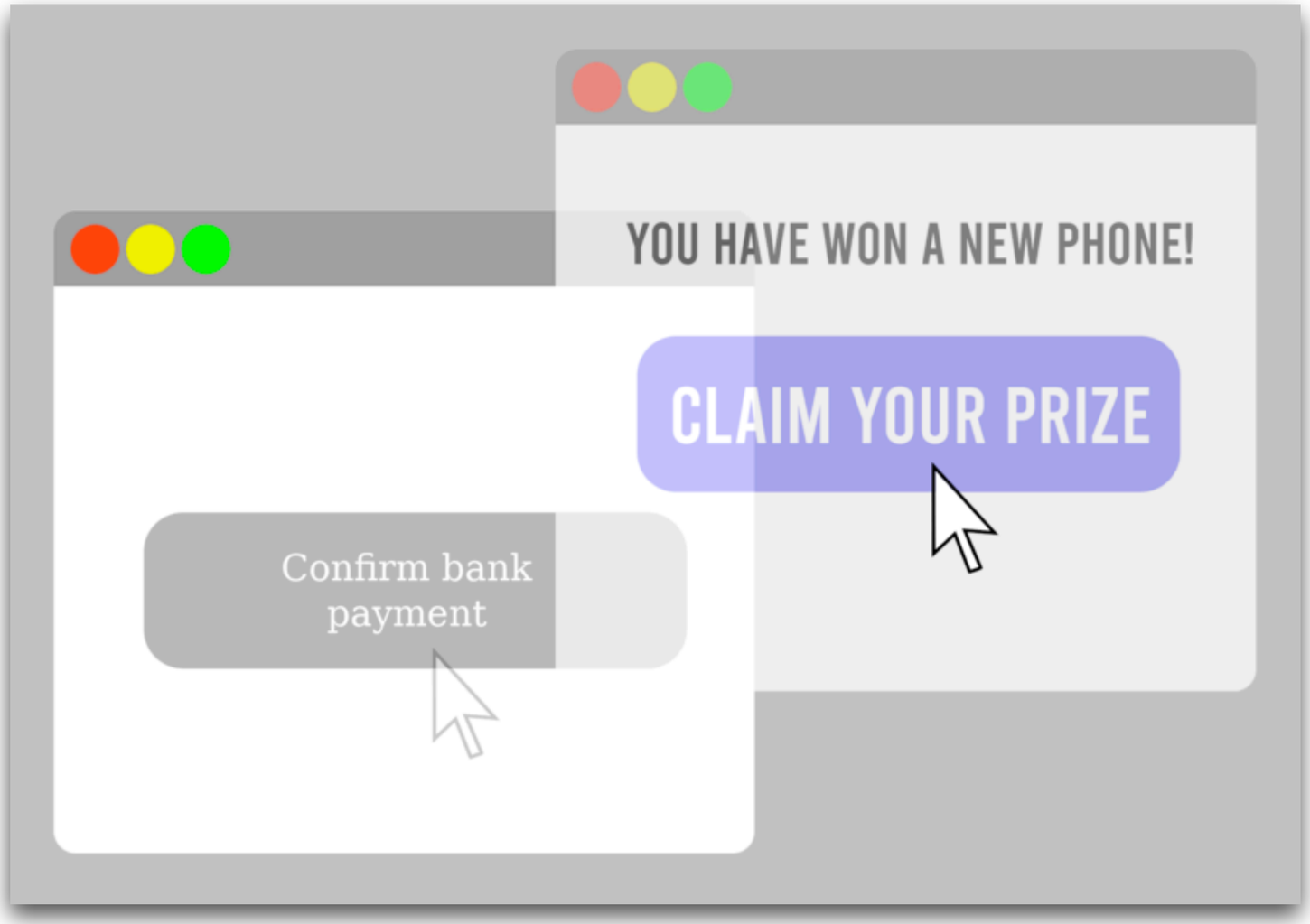
Clickjacking

- **Clickjacking**: Trick the victim into clicking on something from the attacker
- The browser trusts the user's clicks
- Why steal clicks?
  - Download a malicious program
  - Like a Facebook page/YouTube video
  - Delete an online account

Clickjacking: Download Buttons

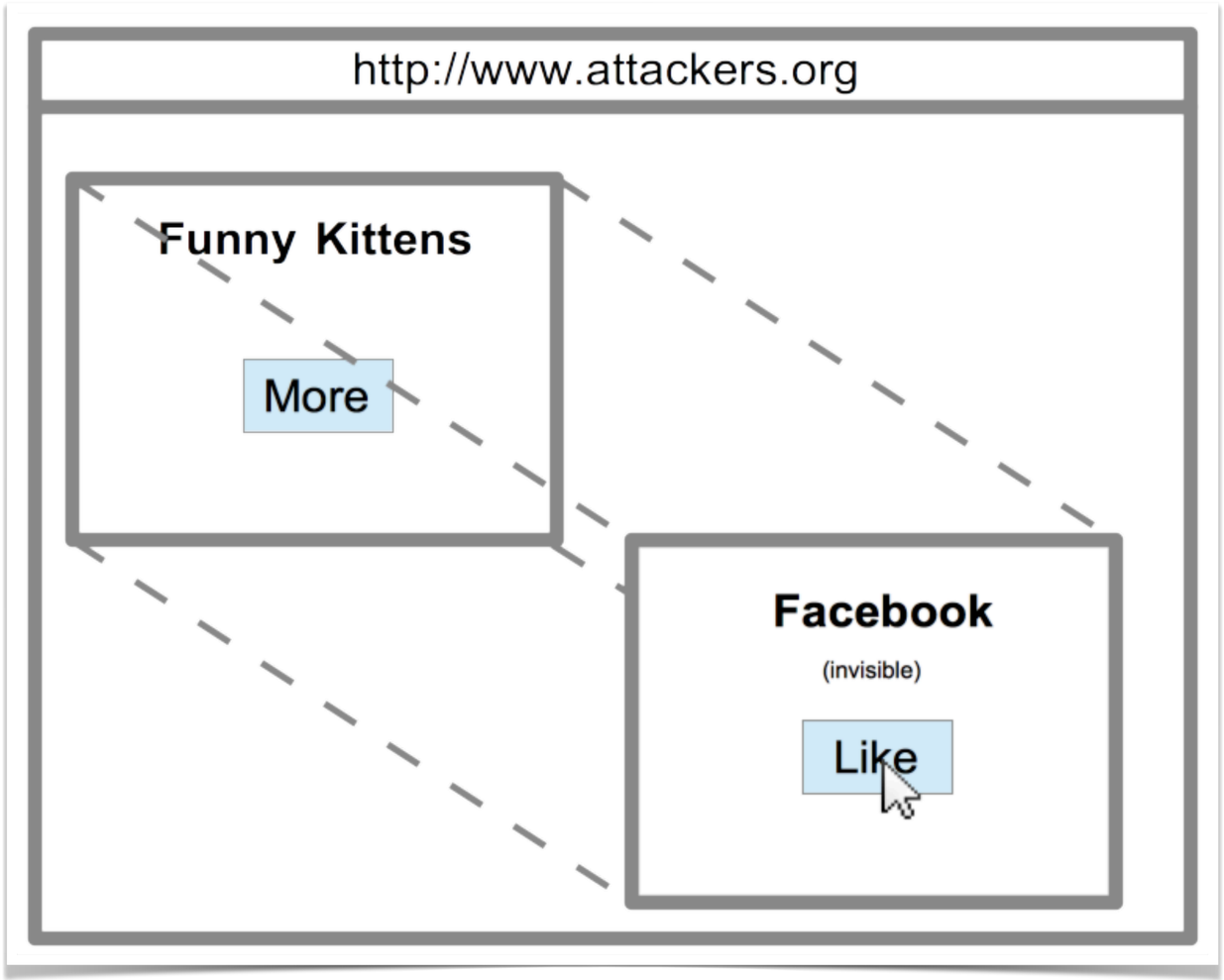
- Which is the real download button?
- What if the user clicks the wrong one?

Invisible iframe Variant #1



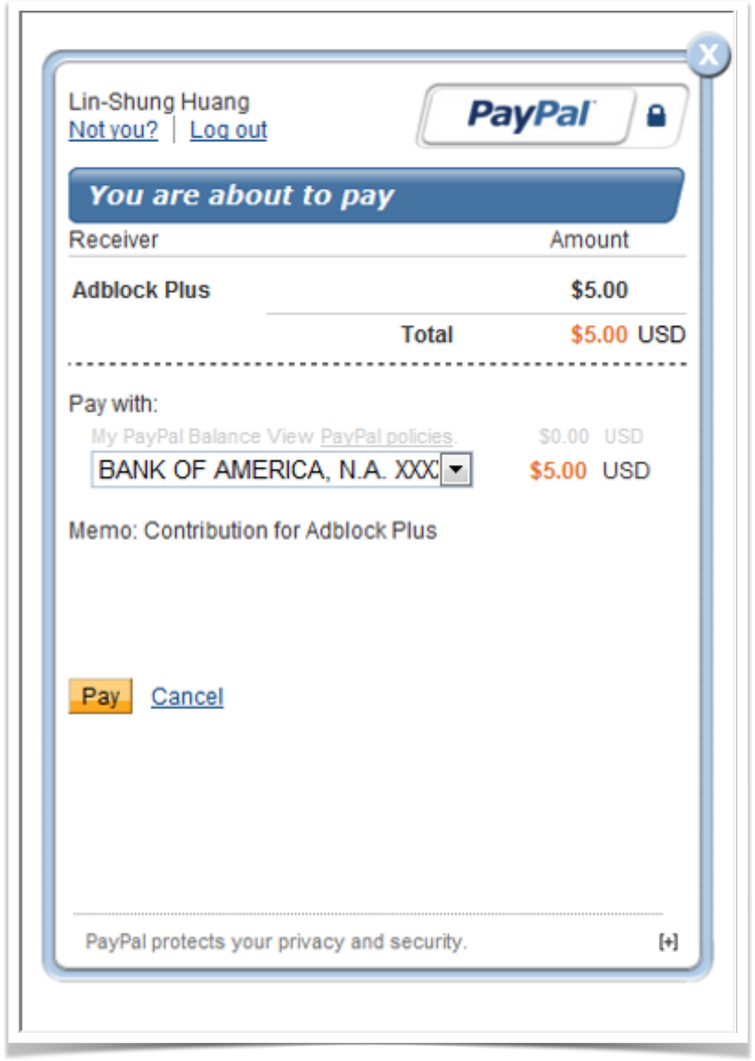
- Frame the legitimate site invisibly, over visible, enticing content
- Victims think they are clicking on the enticing site, but they click on the legitimate site, e.g., pay the attacker's account

Invisible iframe Variant #2

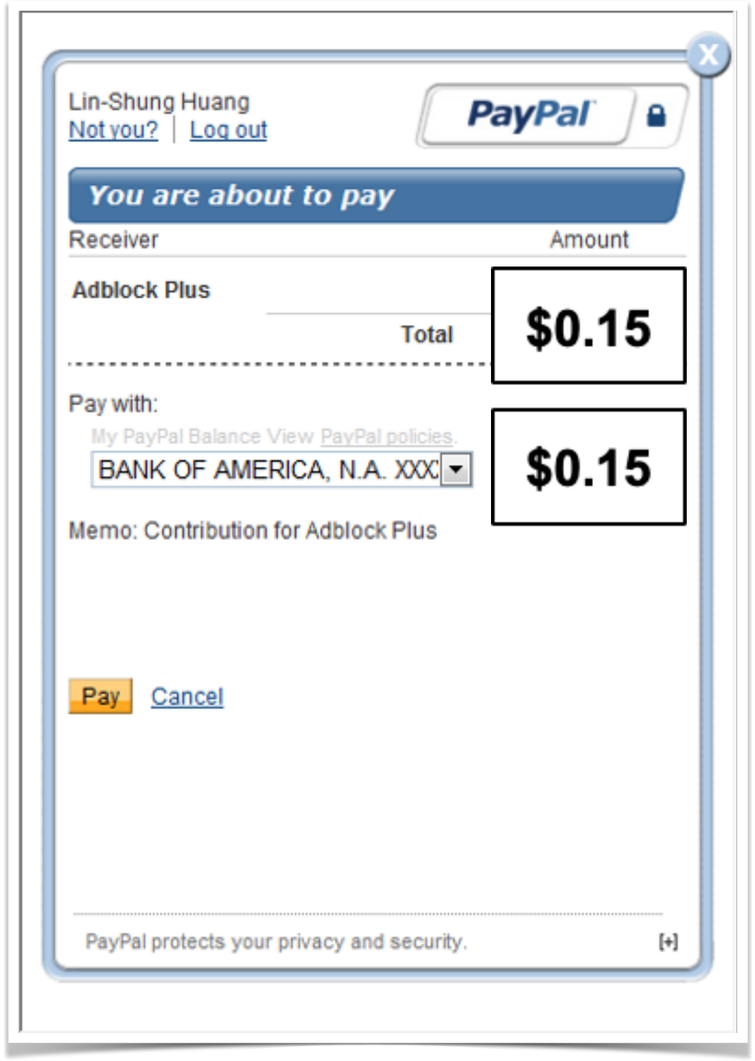


- Frame the legitimate site visibly, under invisible malicious content
- Victims think they are clicking on the visible legitimate site, but their click happens on the malicious site, e.g., fake likes, download malicious software

Invisible iframe Variant #3



Invisible iframe Variant #3



- Frame the legitimate site visibly, under malicious content partially overlaying the site
- The attacker can change the appearance of the site without breaking the Single-Origin Policy

Clickjacking: Temporal Attack

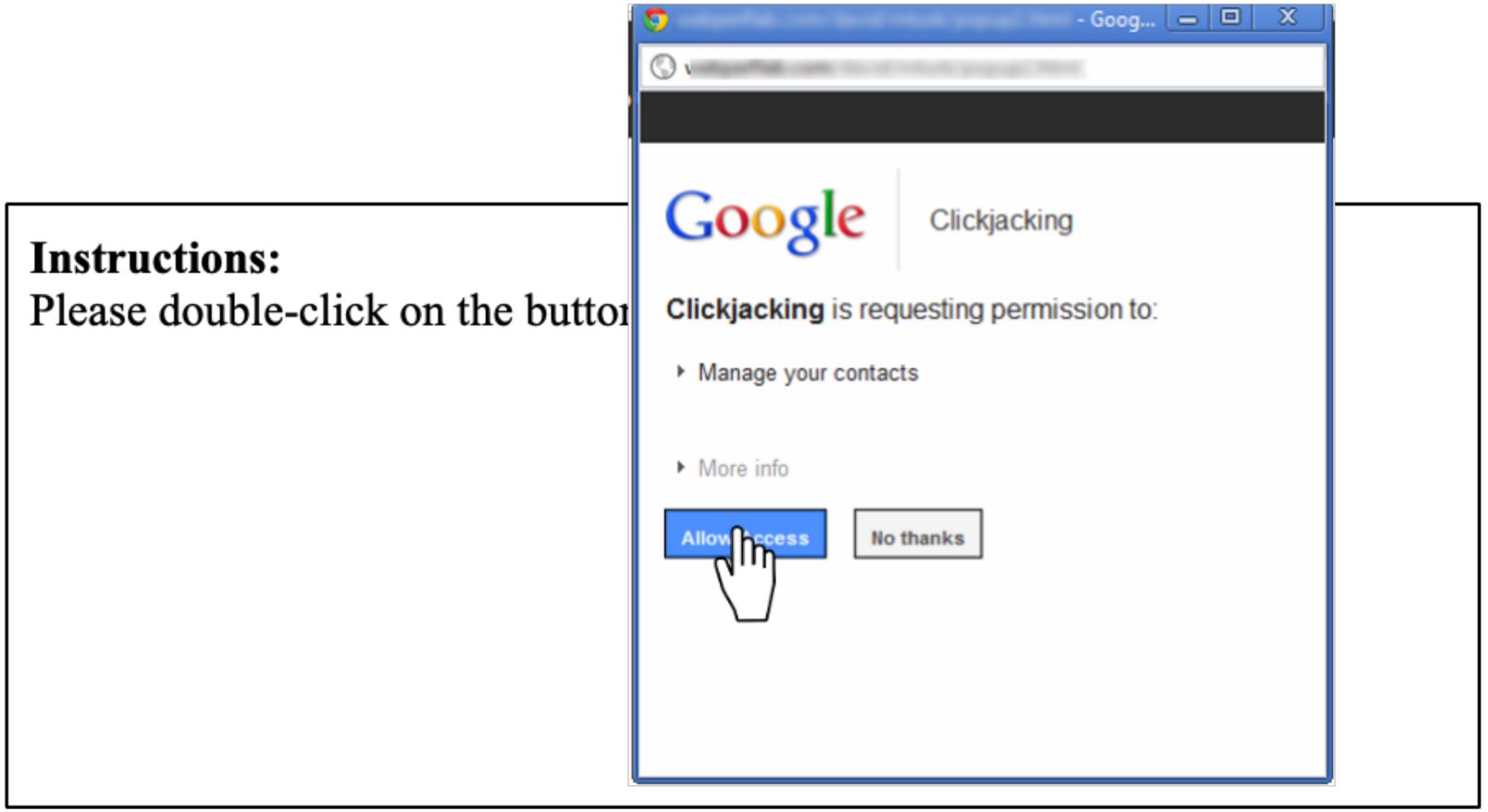
- Attacker uses JavaScript to detect the position of the cursor and change the website right before the user clicks on something
- The user clicks on the malicious input (embedded iframe, download button, etc.) before they notice that something changed

Clickjacking: Temporal Attack

Instructions:
Please double-click on the button below to continue to your content

Click here

Clickjacking: Temporal Attack



Clickjacking: Cursorjacking

Fake cursor, created with CSS and/or JavaScript

Real cursor, hidden or less visible with CSS

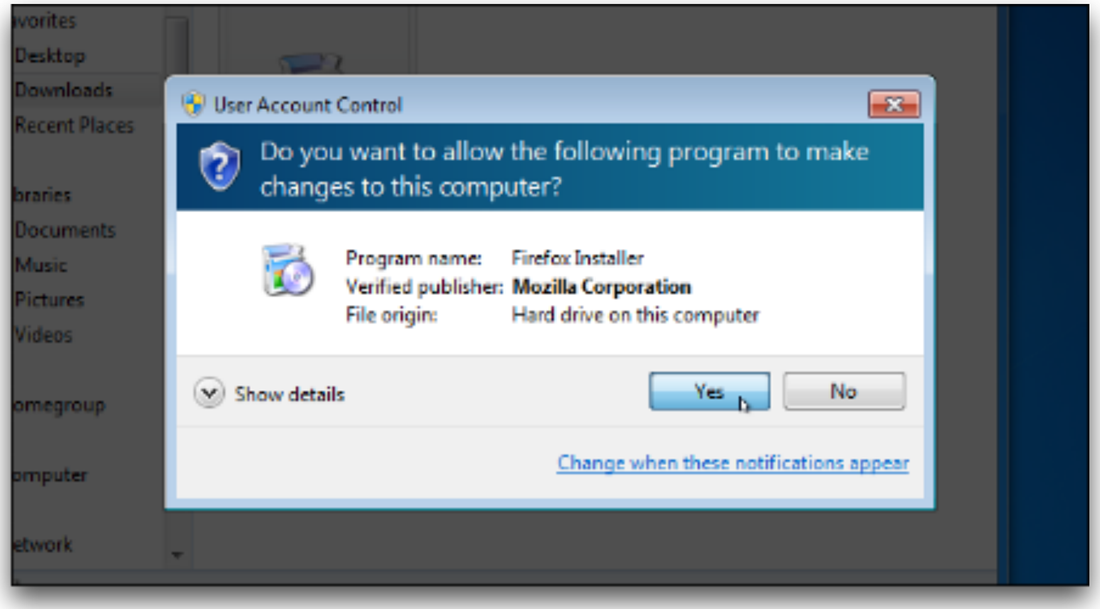
- Arrange a fixed distance between them

Clickjacking: Cursorjacking

- leads victims to misinterpret a click’s target

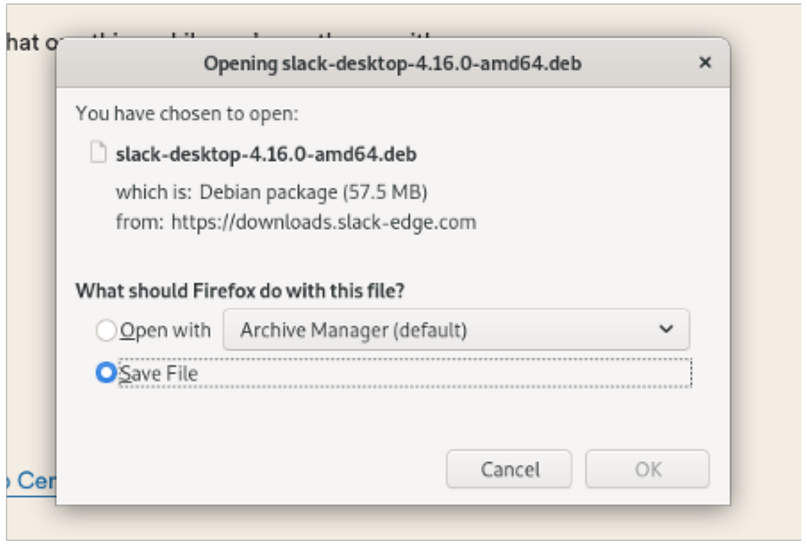
Clickjacking Defense

- **Direct the user's attention to their click:** Ensure clear visual separation between important dialogs and content, e.g., darken the background

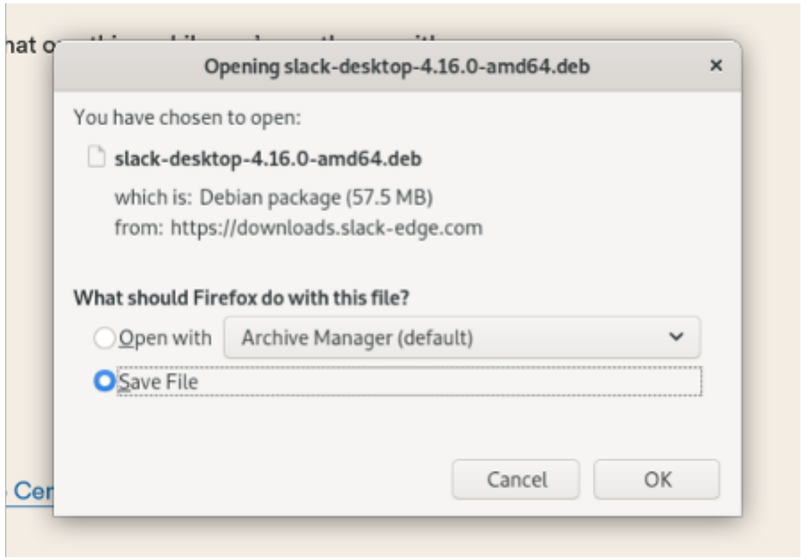


Clickjacking Defense

- **Delay the click:** Force the user to hover over the desired button for some amount of time before allowing the user to click the button.

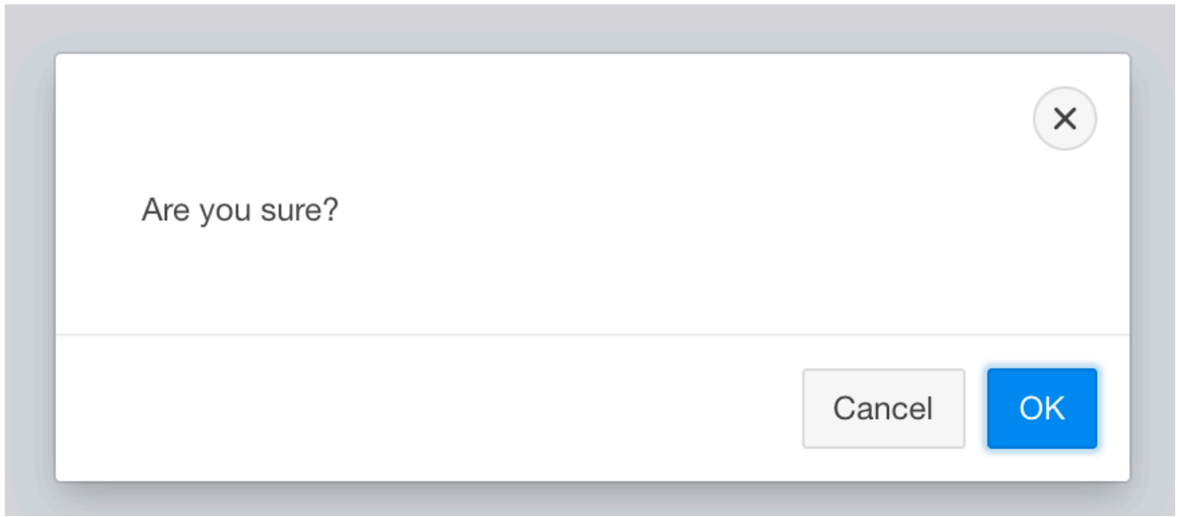


Wait 1 second before allowing click on the OK button



Clickjacking Defense

- **Confirmation pop-ups:**
  - The browser needs to confirm that the user's click was intentional
  - Drawbacks: Asking for confirmation annoys users



Clickjacking Defense

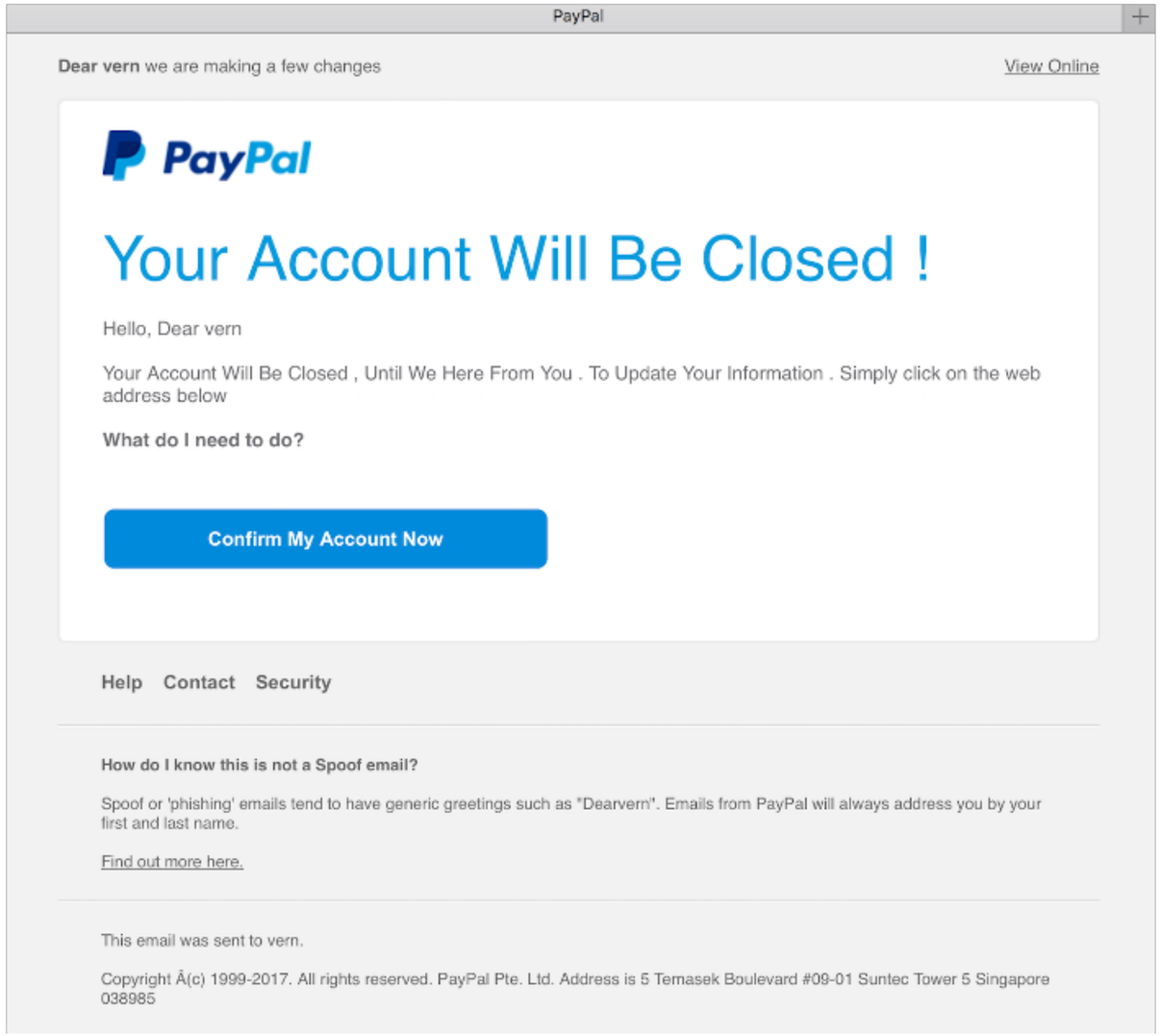
- **Frame-busting:** The legitimate website forbids other websites from embedding it in an iframe
  - Defeats the invisible iframe attacks
  - Can be enforced by Content Security Policy (CSP)
  - Can be enforced by X-Frame-Options (an HTTP header)
  - Drawbacks: relies on the end-user's browser enforcing their own security. This makes the method unreliable.

Phishing

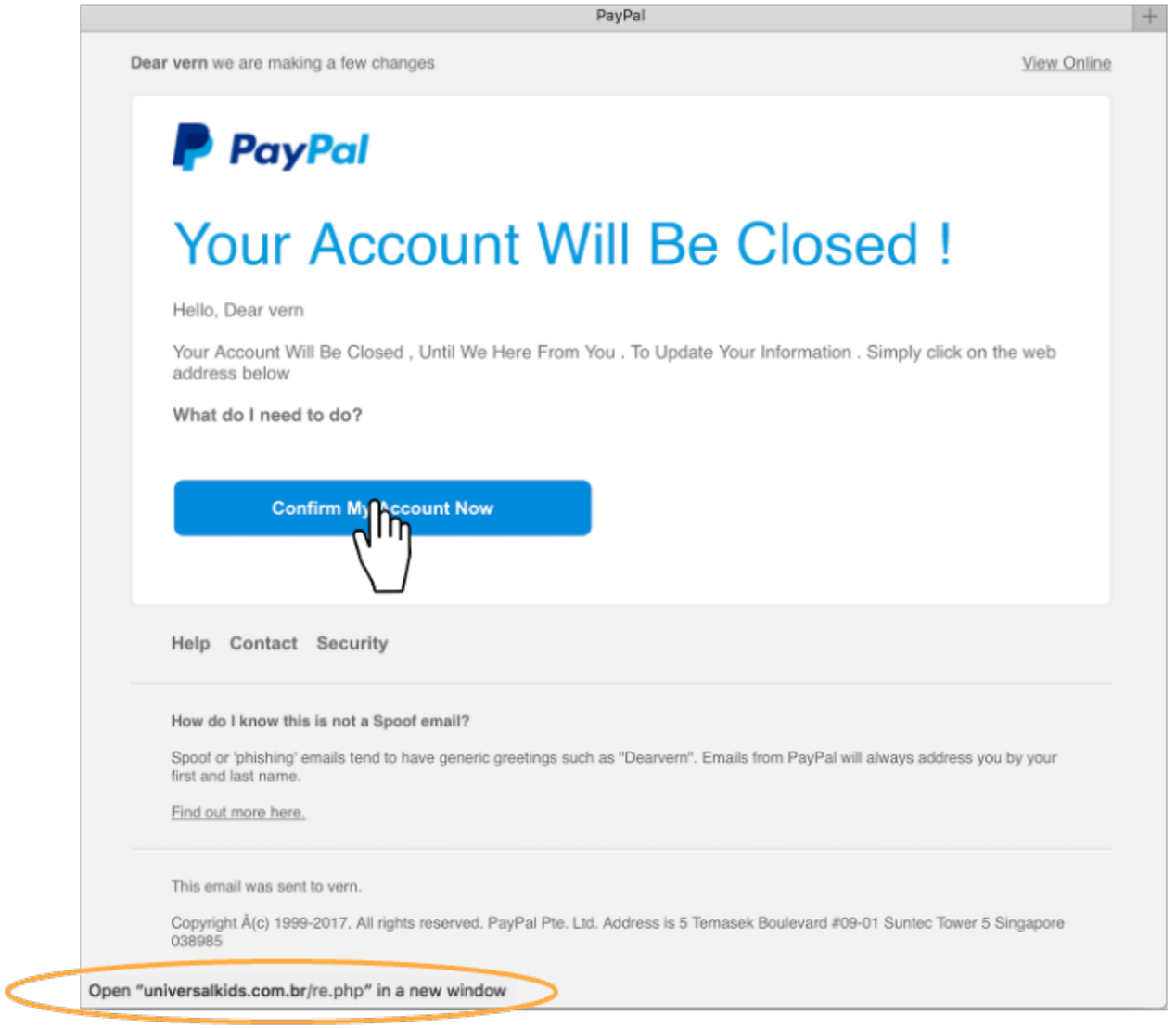
- **Phishing** is a form of social engineering and scam where attackers deceive people into revealing sensitive information or installing malware such as ransomware.

- The user can’t distinguish between a legitimate website and a website impersonating the legitimate website

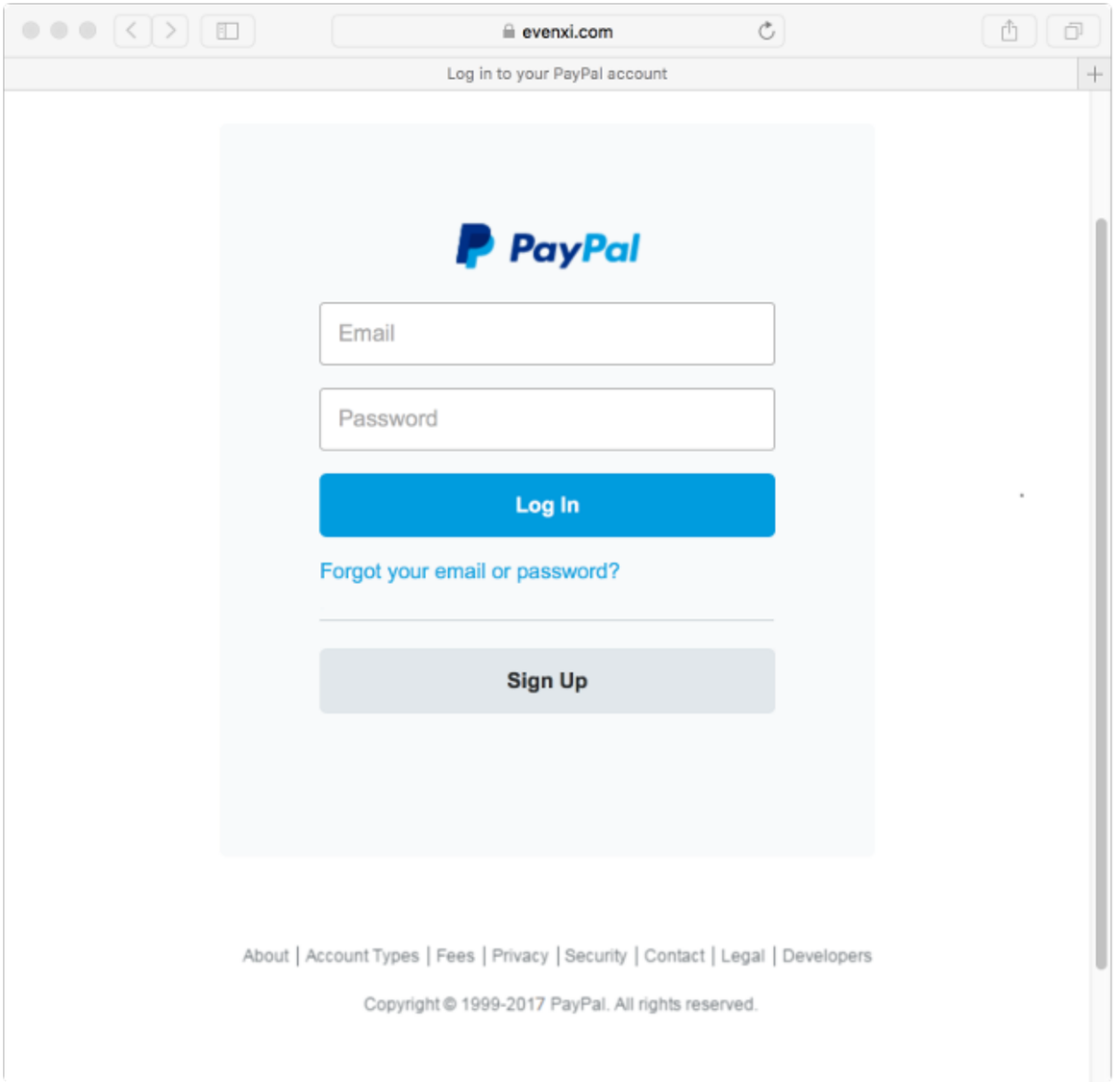
Phishing


PayPal
Dear vern we are making a few changes
View Online
PayPal
Your Account Will Be Closed !
Hello, Dear vern
Your Account Will Be Closed , Until We Here From You . To Update Your Information . Simply click on the web address below
What do I need to do?
Confirm My Account Now
Help Contact Security
How do I know this is not a Spoof email?
Spoof or 'phishing' emails tend to have generic greetings such as "Dearvern". Emails from PayPal will always address you by your first and last name.
Find out more here.
This email was sent to vern.
Copyright Â(c) 1999-2017. All rights reserved. PayPal Pte. Ltd. Address is 5 Temasek Boulevard #09-01 Suntec Tower 5 Singapore 038985

Phishing

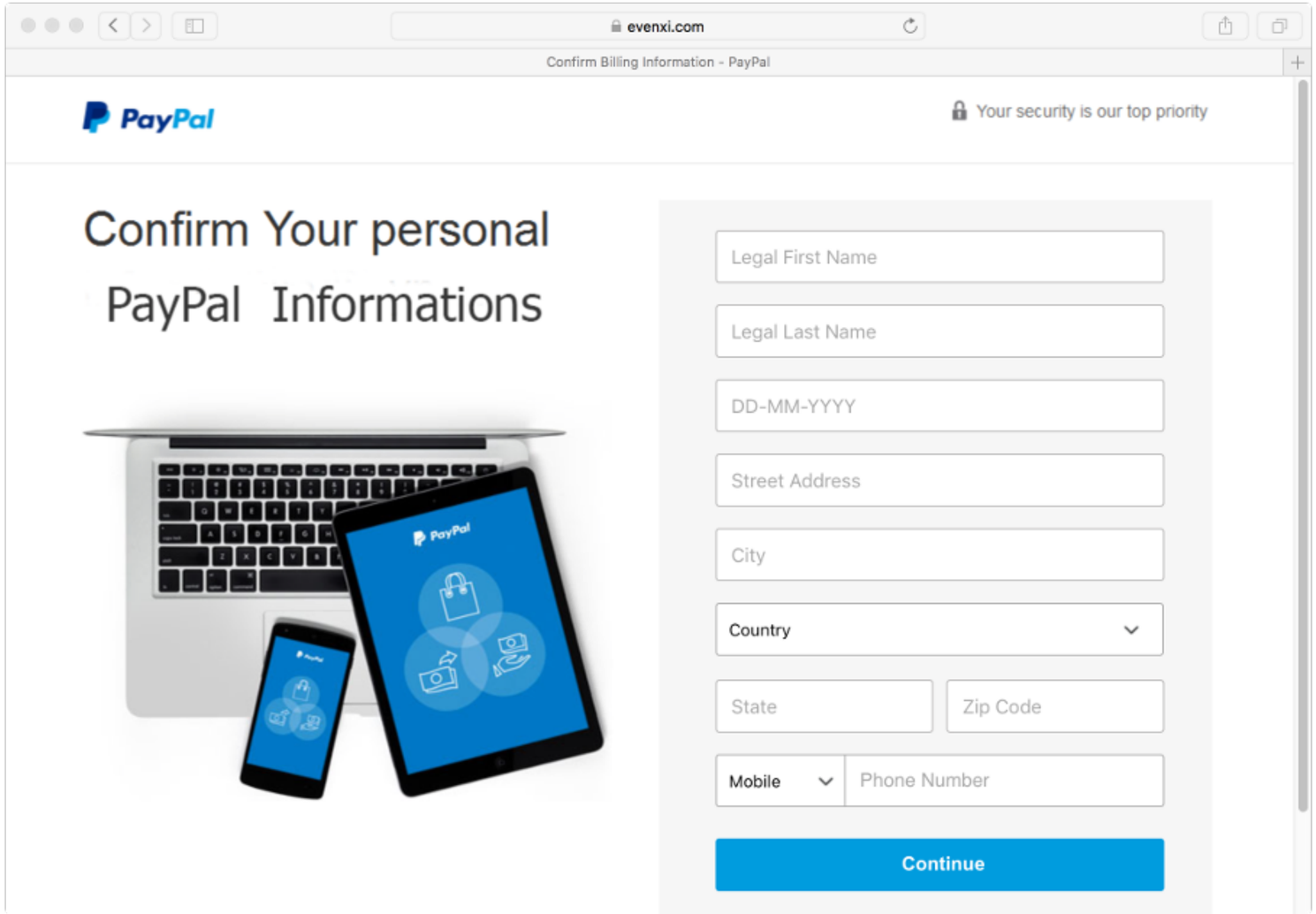

PayPal
Dear vern we are making a few changes
View Online
PayPal
Your Account Will Be Closed !
Hello, Dear vern
Your Account Will Be Closed , Until We Here From You . To Update Your Information . Simply click on the web address below
What do I need to do?
Confirm My Account Now
Help Contact Security
How do I know this is not a Spoof email?
Spoof or 'phishing' emails tend to have generic greetings such as "Dearvern". Emails from PayPal will always address you by your first and last name.
Find out more here.
This email was sent to vern.
Copyright Ã(c) 1999-2017. All rights reserved. PayPal Pte. Ltd. Address is 5 Temasek Boulevard #09-01 Suntec Tower 5 Singapore 038985
Open "universalkids.com.br/re.php" in a new window

Phishing



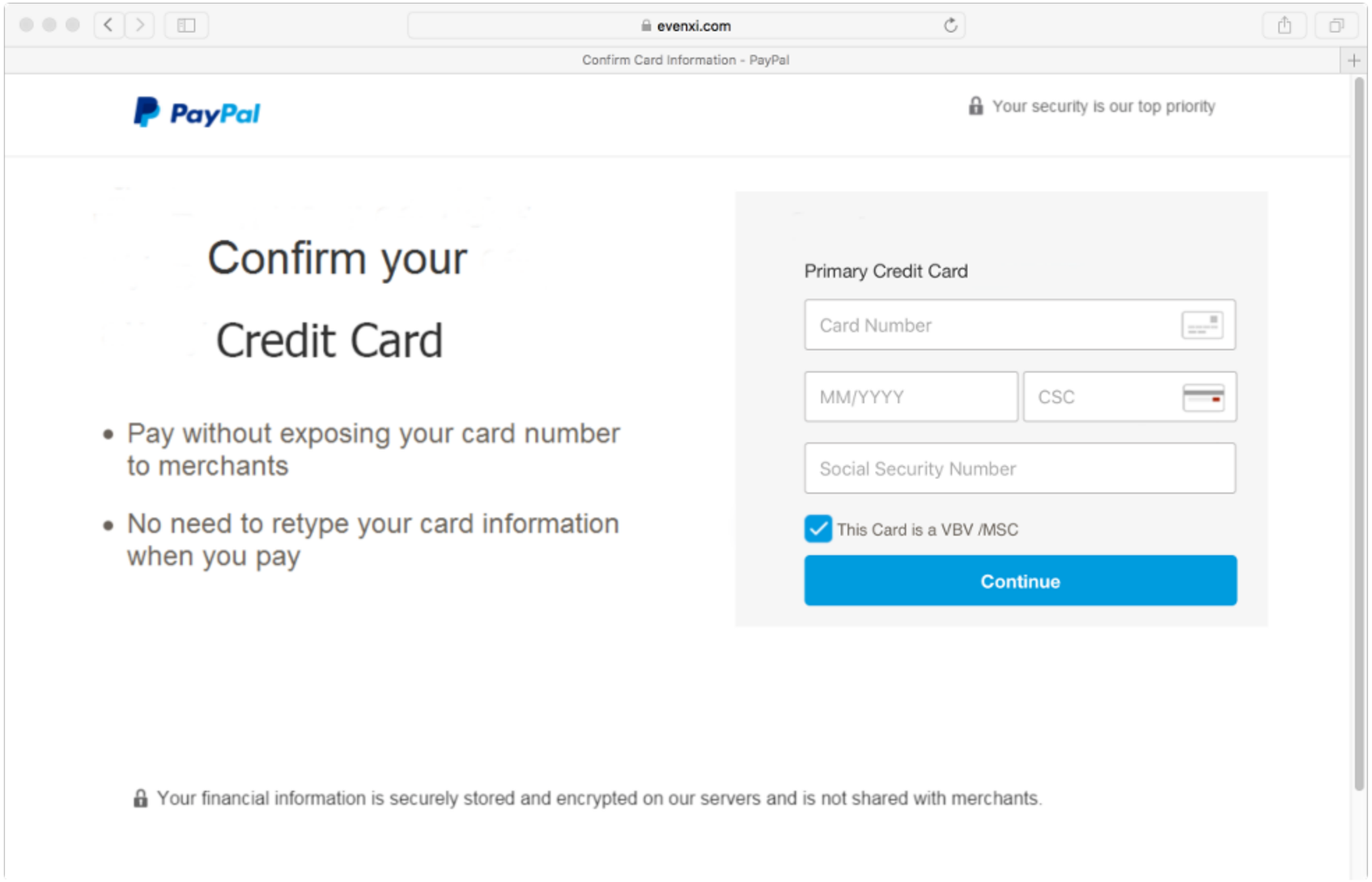
Is this PayPal?

Phishing



After filling out the previous boxes

Phishing

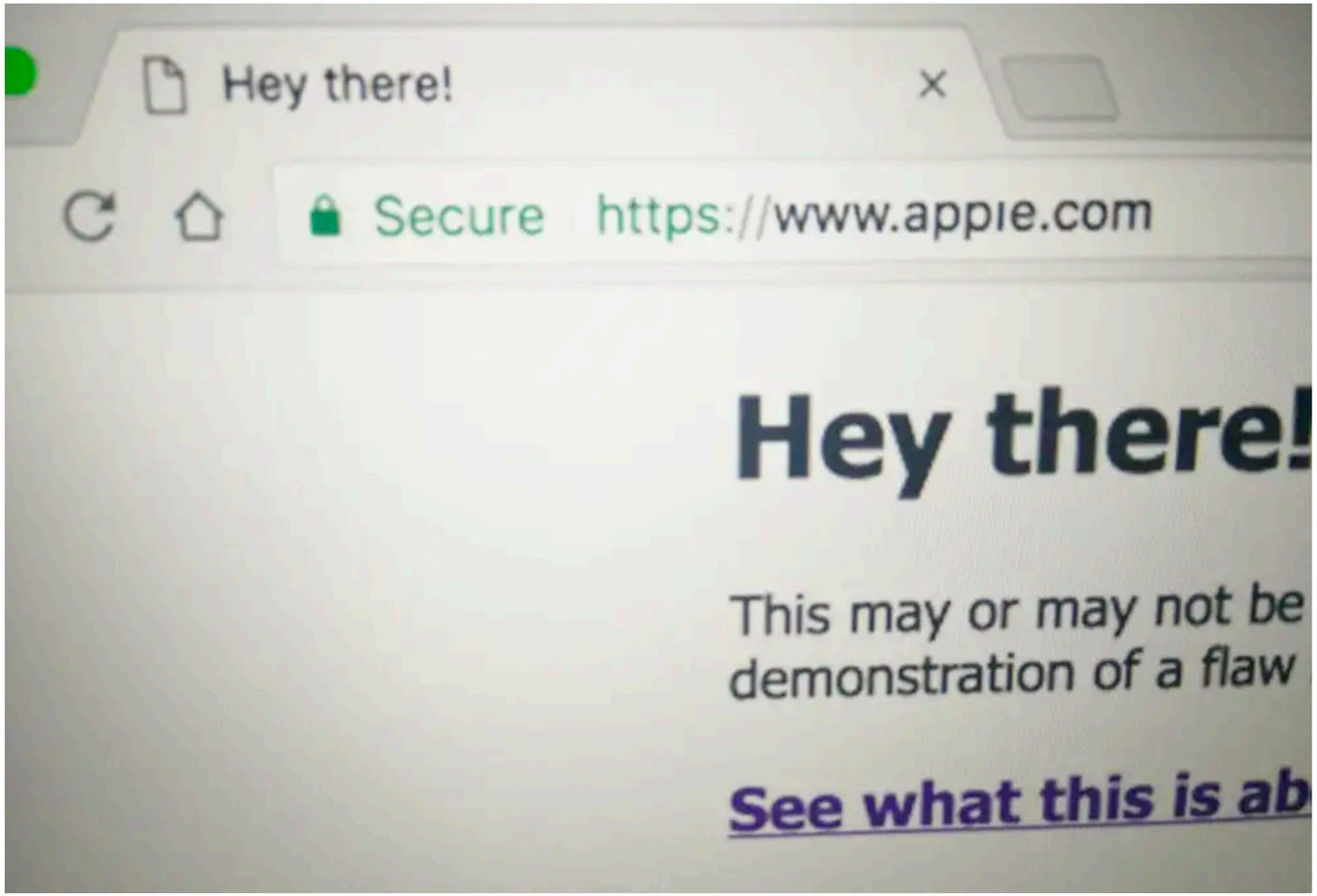


After filling out the previous boxes

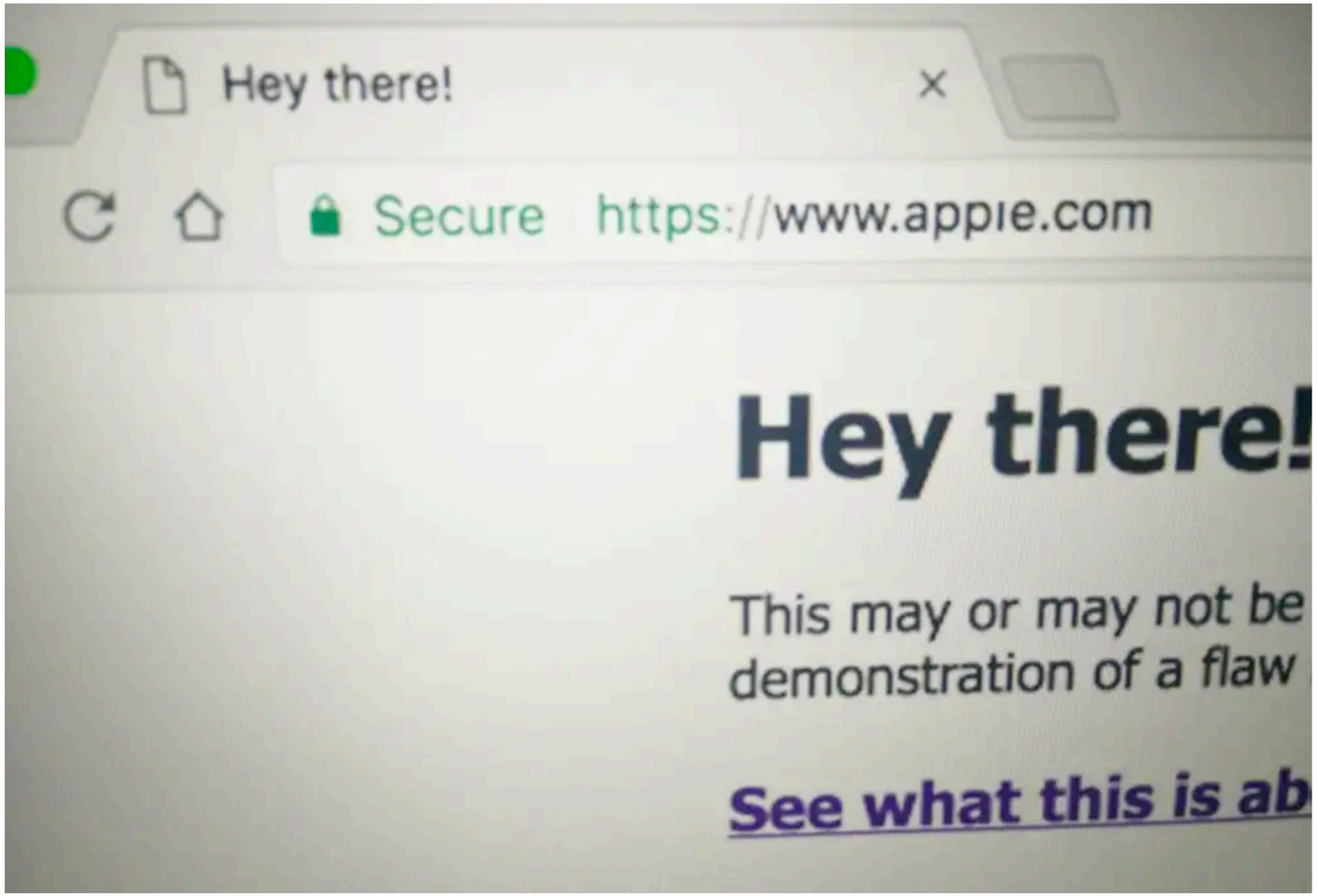
Phishing: Homograph Attacks

- Homograph: Two words that look the same, but have different meanings
- Homograph attack: Creating malicious URLs that look similar (or the same) to legitimate URLs

Phishing: Homograph Attacks



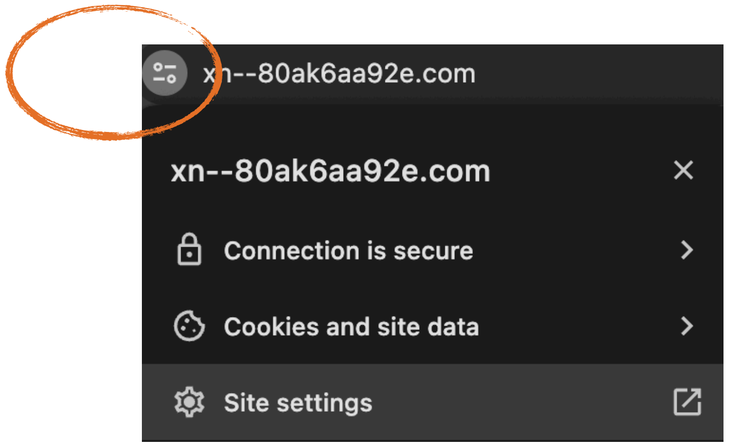
Phishing: Homograph Attacks



- Cyrllic alphabet
- Written in unicode
- Certificate under xn--80ak6aa92e.com
- Looks more real in some browsers

https://www.xudongz.com/blog/2017/idn-phishing/

Phishing: Homograph Attacks



Certificate Viewer: www.xn--80ak6aa92e.com

General | Details

Issued To

Common Name (CN)	www.xn--80ak6aa92e.com
Organization (O)	<Not Part Of Certificate>
Organizational Unit (OU)	<Not Part Of Certificate>

Issued By

Common Name (CN)	R3
Organization (O)	Let's Encrypt
Organizational Unit (OU)	<Not Part Of Certificate>

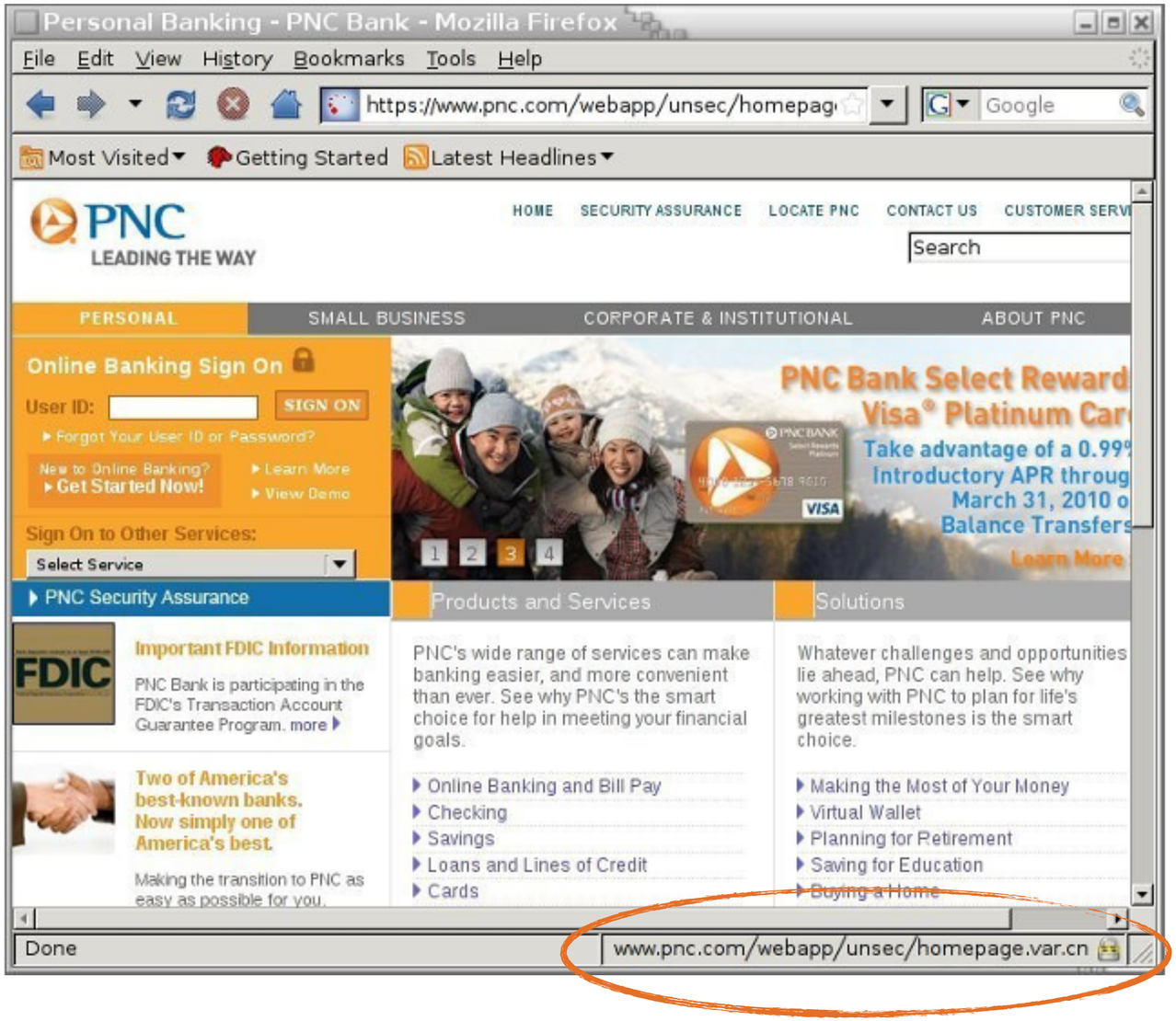
Validity Period

Issued On	Tuesday, February 13, 2024 at 6:40:10 AM
Expires On	Monday, May 13, 2024 at 7:40:09 AM

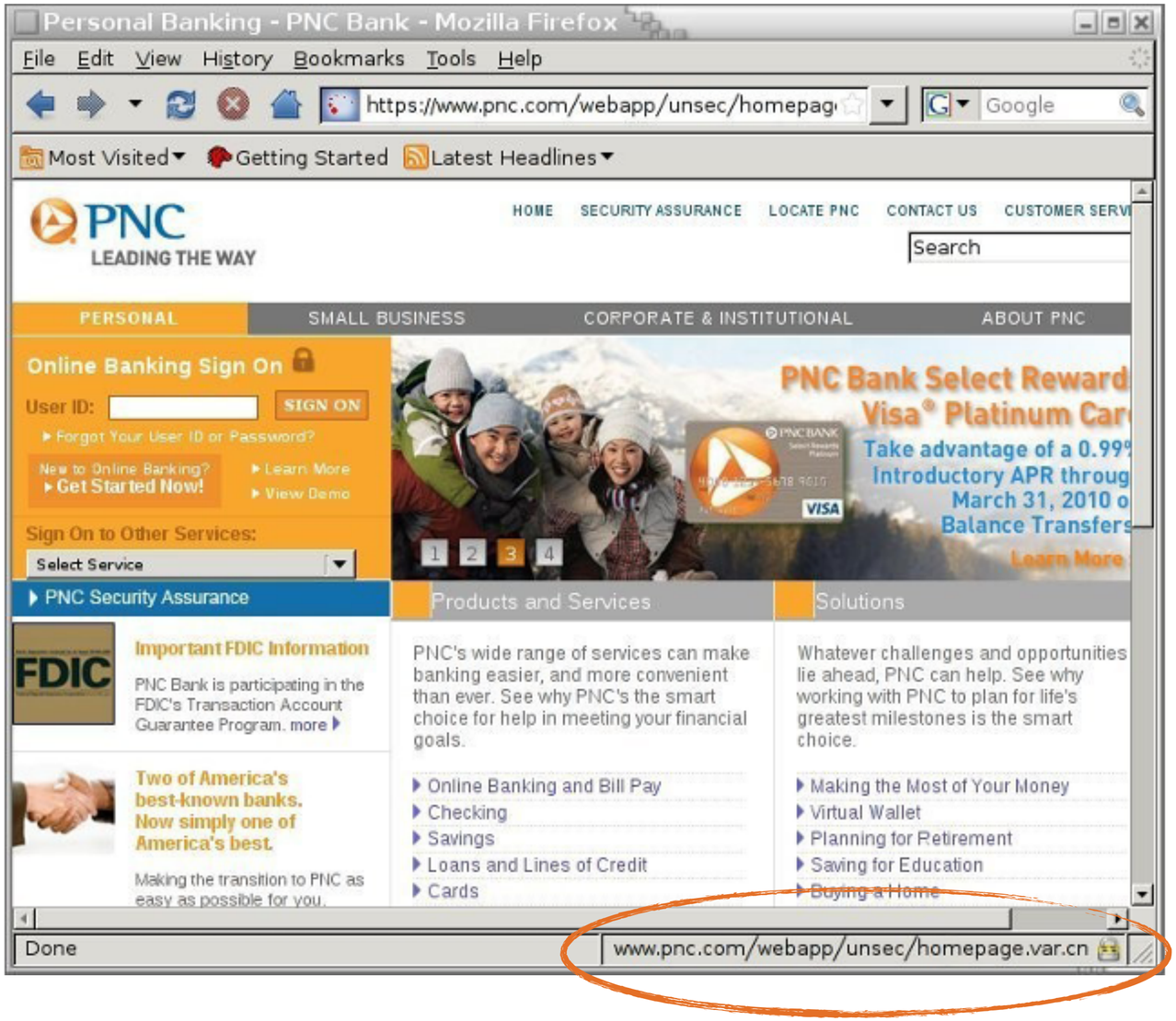
SHA-256 Fingerprints

Certificate	35ba295b1ef463f81382aae94f78046f4597bee69b5ed608a6114277702fef86
Public Key	eec04ffa1e48615fa9cf2b0b728c7543415a6a7a2691dabb08f8a7991b71e50c

Phishing: Homograph Attacks

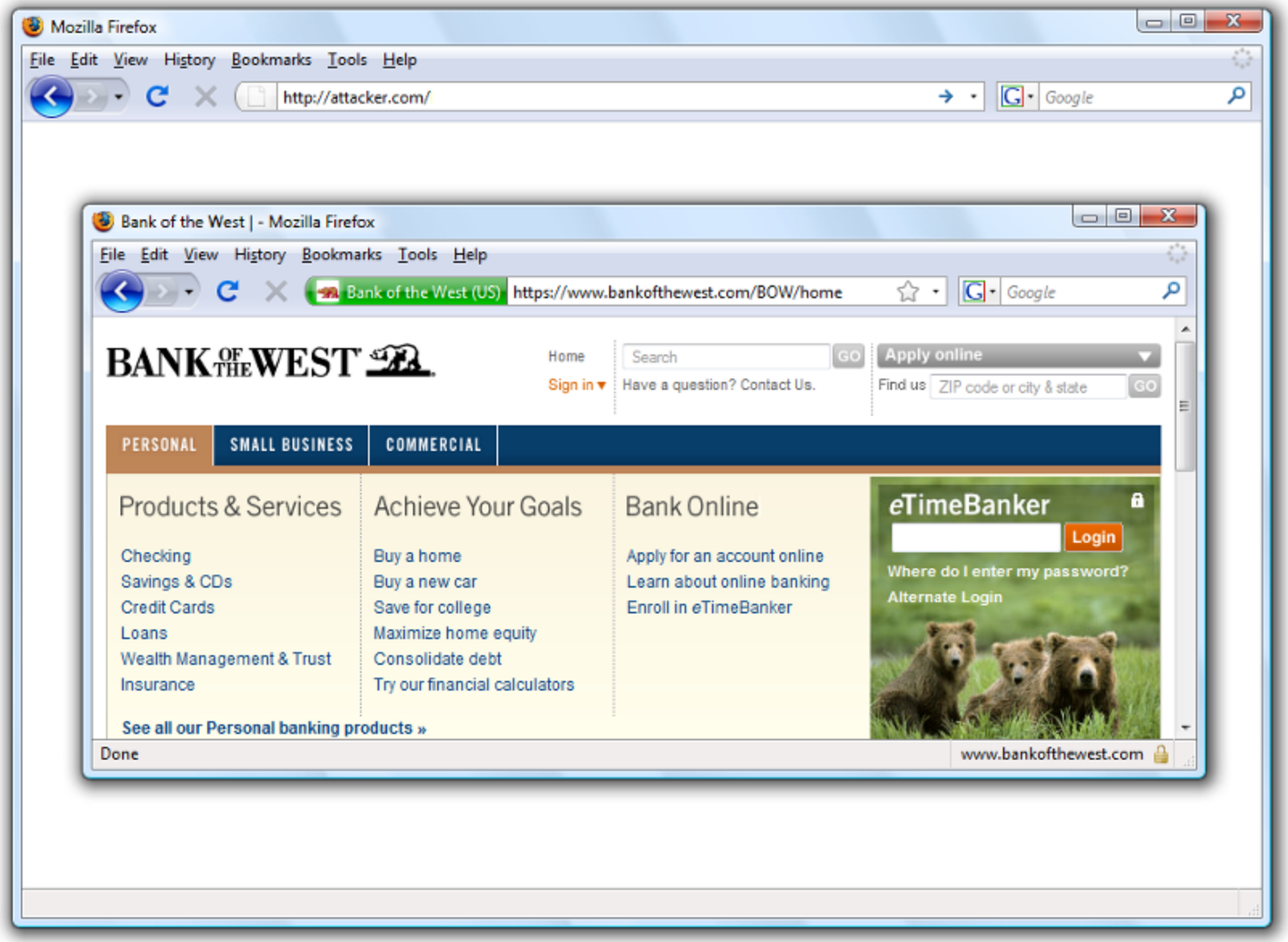


Phishing: Homograph Attacks



- Unicode characters 2044 (⁄) and 2215 (∕) are allowed in hostnames.
- Confusing chars

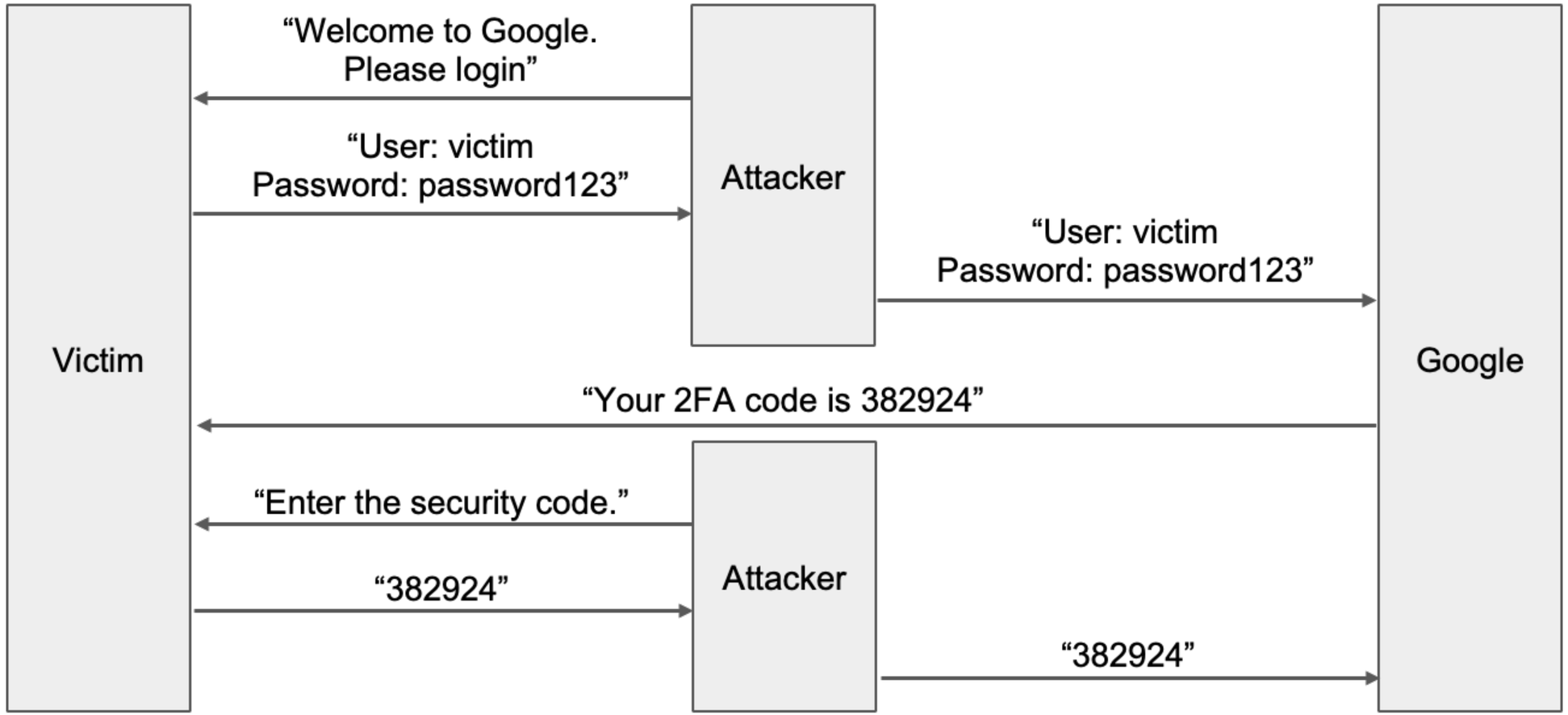
Phishing: Browser in Browser Attack



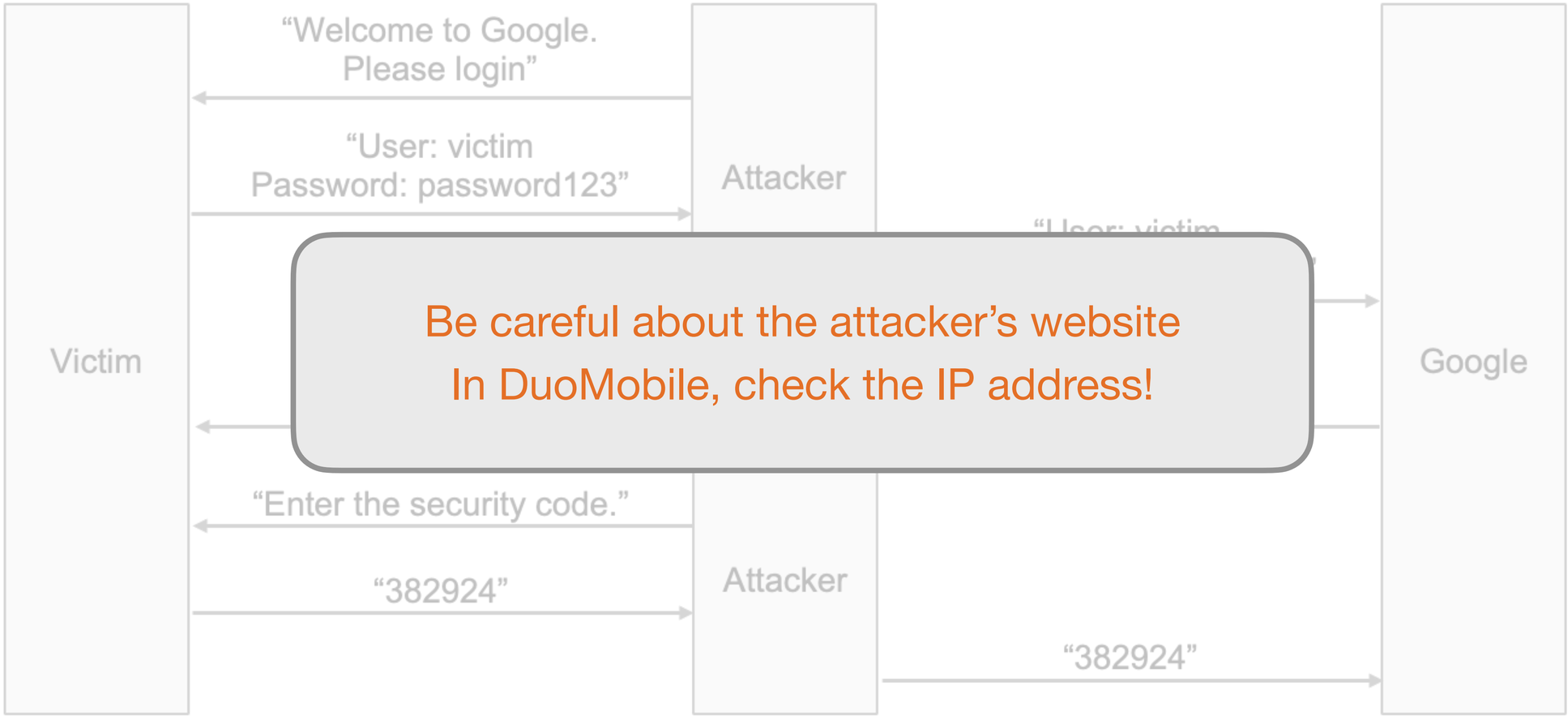
Two-Factor Authentication

- Problem: Phishing attacks allow attackers to learn passwords
- Idea: Require more than passwords to log in
- **Two-factor authentication (2FA):** The user must prove their identity in two different ways before successfully authenticating
- Three main ways for a user to prove their identity
  - **Something the user knows:** Password, security question (e.g. name of your first pet)
  - **Something the user has:** Their phone, their security key
  - **Something the user is:** Fingerprint, face ID
- Even if the attacker steals the user's password with phishing, they don't have the second factor!

Subverting 2FA: Relay Attacks / MiTM



Subverting 2FA: Relay Attacks / MiTM



2FA Example: Authentication Tokens

- Authentication token: A device that generates secure second-factor codes (Something the user owns)
- Examples: RSA SecurID, Google Authenticator, DuoMobile

2FA Example: Authentication Tokens

- Authentication token: A device that generates secure second-factor codes (Something the user owns)
- Examples: RSA SecurID, Google Authenticator, DuoMobile
  - The token and the server share a common secret key k
  - When the user wants to log in, the token generates a code HMAC(k, time)
  - The time is often truncated to the nearest 30 seconds for usability
  - The code is often truncated to 6 digits for usability
  - The user submits the code to the website
  - The website uses its secret key to verify the HMAC

2FA Example: Authentication Tokens

- Authentication token: A device that generates secure second-factor codes (Something the user owns)
- Examples: RSA SecurID, Google Authenticator, DuoMobile
  - The token and the server share a common secret key k
  - When the user wants to log in, the token generates a code HMAC(k, time)
  - The time is often truncated to the nearest 30 seconds for usability
  - The code is often truncated to 6 digits for usability
  - The user submits the code to the website
  - The website uses its secret key to verify the HMAC
- Drawback: Vulnerable to online brute-force attacks; Possible fix: Add a timeout
- Drawback: Vulnerable to relay attacks; Fix: User needs to be more careful, read the IP address

Subverting 2FA: Social Engineering

- Some 2FA schemes text a one-time code to a phone number
  - Attackers can call your phone provider (e.g. Verizon) and tell them to activate the attacker's SIM card, so they receive your texts!
  - 2FA via SMS is not great but better than nothing

Subverting 2FA: Social Engineering

- Some 2FA schemes text a one-time code to a phone number
  - Attackers can call your phone provider (e.g. Verizon) and tell them to activate the attacker's SIM card, so they receive your texts!
  - 2FA via SMS is not great but better than nothing
- Some 2FA schemes can be bypassed with customer support
  - Attackers can call customer support and ask them to deactivate 2FA!
  - Companies should validate identity if you ask to do this (but not all do)

Agenda

- UI Attacks
- CAPTCHAs
- Security Principles

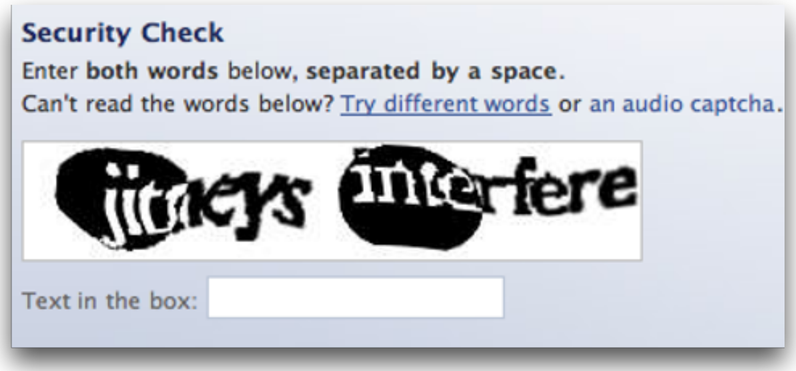
Websites are for Humans

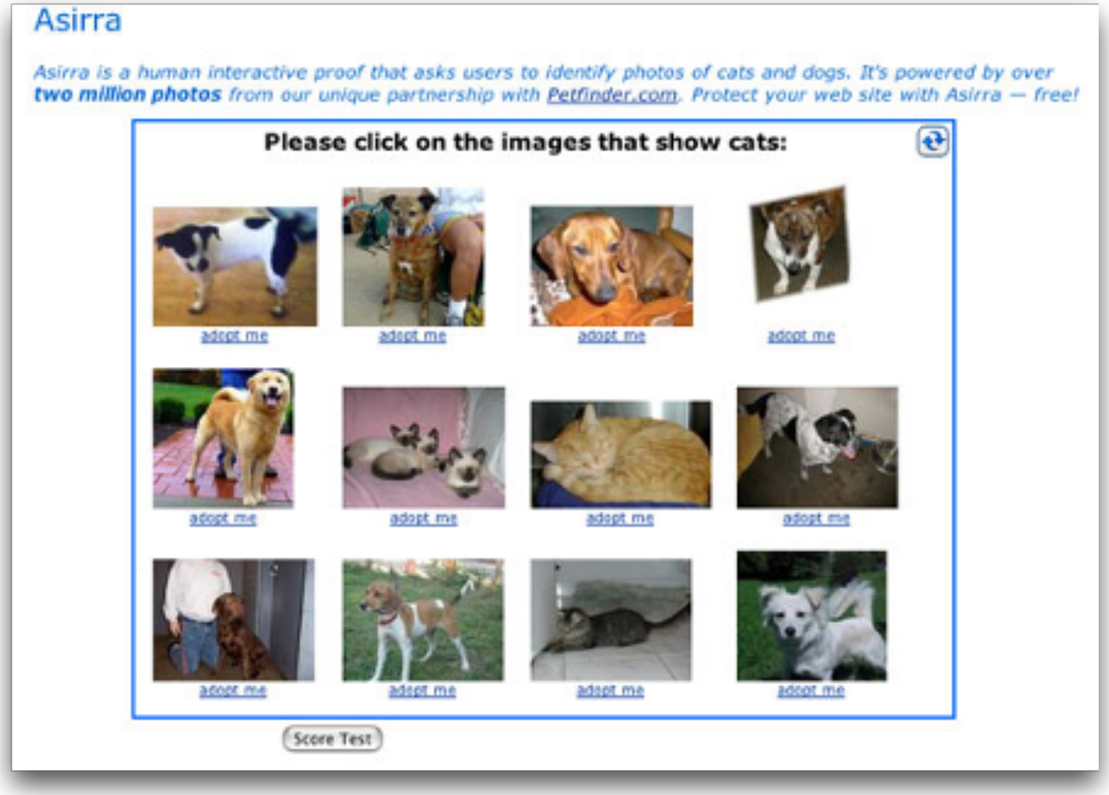
- Most websites are designed for human usage, not robot usage
  - Example: A login page is for users to submit their password, not for an attacker to automate a brute-force attack
- Robot access of websites can lead to attacks
  - Example: Denial of service: Overwhelming a web server by flooding it with requests

CAPTCHAs: Definition

- **CAPTCHA:** A challenge that is easy for a human to solve, but hard for a computer to solve
  - "**C**ompletely **A**utomated **P**ublic **T**uring test to tell **C**omputers and **H**umans **A**part"
  - Sometimes called a "reverse Turing test"
  - Used to distinguish web requests made by humans and web requests made by robots
- Usage: Administer a CAPTCHA, and if it passes, assume that the user is human and allow access

CAPTCHAs: Examples

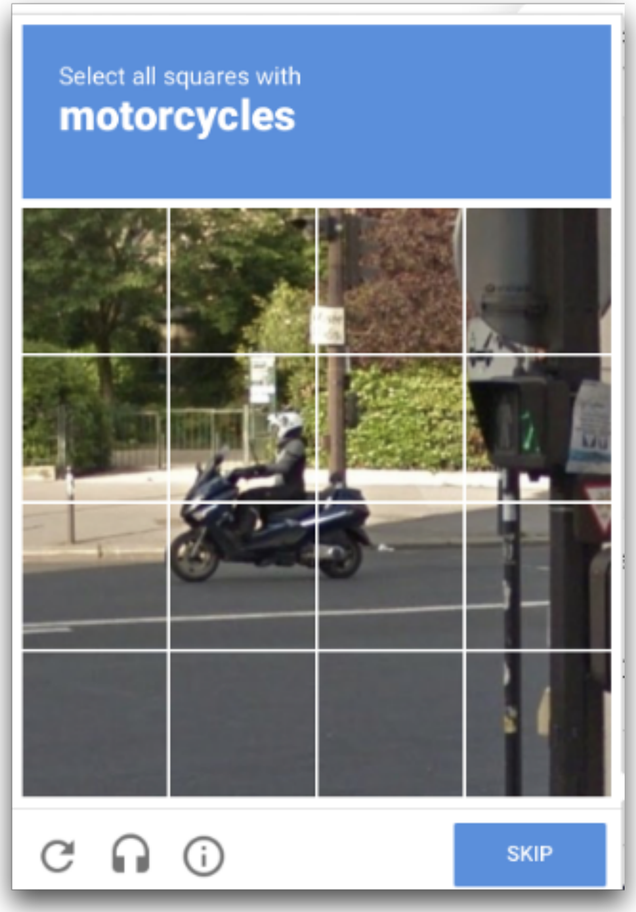




- Reading distorted text
- Identifying images
- Listening to an audio clip and typing out the words spoken

CAPTCHAs and Machine Learning

- Modern CAPTCHAs have another purpose: Training machine learning algorithms
  - Machine learning often requires manually-labeled datasets
  - CAPTCHAs crowdsource human power to help manually label these big datasets
  - Example: Machine vision problems require manually-labeled examples: “This is a stop sign”



Security Principles

- Confidentiality, Integrity, Availability, Authentication
- Detect if you can’t prevent
- Defense in depth
- Least privilege
- Separation of responsibility / privileges
- Ensure complete mediation
- Don’t rely on security through obscurity
- Use fail-safe defaults
- Design in security from the start
- Consider human factors

Detect if you can’t prevent

- **Prevention**: Stop the attack from taking place
- **Detection**: Learn that there was an attack
  - If you can’t stop the attack from happening, you should at least be able to know that the attack has happened.
- **Response**: Do something about the attack (after it happened)
  - Once you know the attack happened, you should respond
  - Detection without response is pointless!

Response: Mitigation and Recovery

- Assume that bad things will happen! You should plan security in way that lets you to get back to a working state.
- Example: Mitigate the Consequences from Potential Ransomware
  - Keep offsite backups!
- Example: Recover your homework if the computer stops working
  - Use Git version control, push frequently

Response: Mitigation and Recovery

- Assume that bad things will happen! You should plan security in way that lets you to get back to a working state.
- Example: Mitigate the Consequences from Potential Ransomware
  - Keep offsite backups!
- Example: Recover your homework if the computer stops working
  - Use Git version control, push frequently
- Bad Example: Bitcoin transactions are irreversible. If you are hacked, you can never recover your Bitcoins.
  - $68M stolen from NiceHash exchange in December 2017
  - Four multi-million-dollar attacks on Ethereum in July 2018
  - Coinbase: One detected theft per day

Defense in Depth

- Multiple types of defenses should be layered together
  - An attacker should have to breach all defenses to successfully attack a system
  - e.g., multiple defenses for buffer overflow, sql injection, XSS, CSRF
- However, consider security is economics
  - Defenses are not free.
  - Defenses are often less than the sum of their parts

Principle of Least Privilege

- Consider what permissions an entity or program needs to be able to do its job correctly
  - If you grant unnecessary permissions, a malicious or hacked program could use those permissions against you
  - e.g., non-executable pages, same-origin policy

Separation of Responsibility / Privileges

- If you need to have a privilege, consider requiring multiple parties to work together (collude) to exercise it
  - It's much more likely for a single party to be malicious than for all multiple parties to be malicious and collude with one another
  - e.g., requires multiple keys from different people to access an important system

Ensure Complete Mediation

- Ensure that every access point is monitored and protected
  - **Reference monitor**: Single point through which all access must occur
  - Example: A network firewall, airport security, the doors to the dorms
- Desired properties of reference monitors:
  - Correctness
  - Completeness (can’t be bypassed)
  - Security (can’t be tampered with)

TOCTTOU Vulnerabilities

- A common failure of ensuring complete mediation involving race conditions

```
procedure withdrawal(w)
  // contact central server to get balance
  1. let b := balance

  2. if b < w, abort

  // contact server to set balance
  3. set balance := b - w

  4. give w dollars to user
```

TOCTTOU Vulnerabilities

```
procedure withdrawal($100)
  1. let b := balance

  2. if b < $100, abort

  // contact server to set balance
  3. set balance := b - $100

  4. give w dollars to user
```

Time

```
procedure withdrawal($100)
  1. let b := balance

  2. if b < $100, abort

  // contact server to set balance
  3. set balance := b - $100

  4. give w dollars to user
```

- If I only have $100
- Withdraw $200

Use Fail-Safe Defaults

- Choose default settings that “fail safe,” balancing security with usability when a system goes down
  - e.g., Content Security Policy: By default, reject JavaScript from all websites, use an allowlist to accept some JavaScript from trustworthy website

Design in Security from the Start

- When building a new system, include security as part of the design considerations rather than patching it after the fact
  - A lot of systems today were not designed with security from the start, resulting in patches that don’t fully fix the problem!
- Keep these security principles in mind whenever you write code!

Consider Human Factors

- Users like convenience; if a security system is unusable and not user-friendly, no matter how secure it is, it will go unused
- Example:
  - Pop-up box: install secure update? Users click “remind me later”
  - Automatically downloads important updates by default, easy install and restart
- Consider factors such as developers make mistakes, users are susceptible to social engineering attacks…