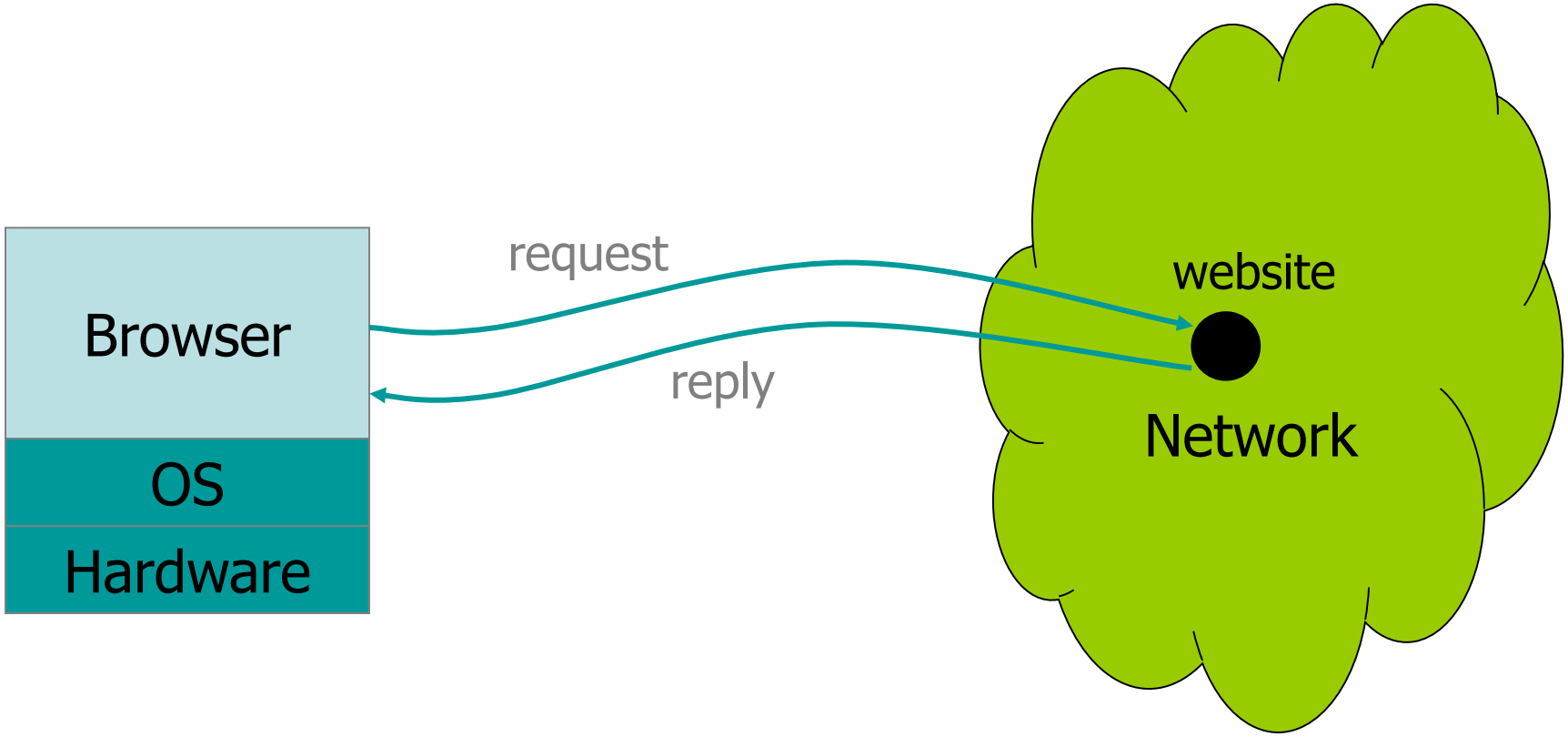
Web browser and Web application security

Browser and Network



Web security topics

- JavaScript security, Same Origin policy, Attacks: XSS, XSRF, SQL injection.
- Browser security issues.

HTTP: HyperText Transfer Protocol

- Used to request and return data
  - Methods: `GET`, `POST`, `HEAD`, …
- Stateless request/response protocol
  - Each request is independent of previous requests
  - Statelessness has a significant impact on design and implementation of applications

HTTP Request

Method File HTTP version Headers

```
GET /default.asp HTTP/1.0
Accept: image/gif, image/x-bitmap, image/jpeg, */*
Accept-Language: en
User-Agent: Mozilla/1.22 (compatible; MSIE 2.0; Windows 95)
Connection: Keep-Alive
If-Modified-Since: Sunday, 17-Apr-96 04:32:58 GMT
```

Blank line

Data – none for GET

HTTP Response

HTTP version

Status code

Reason phrase

Headers

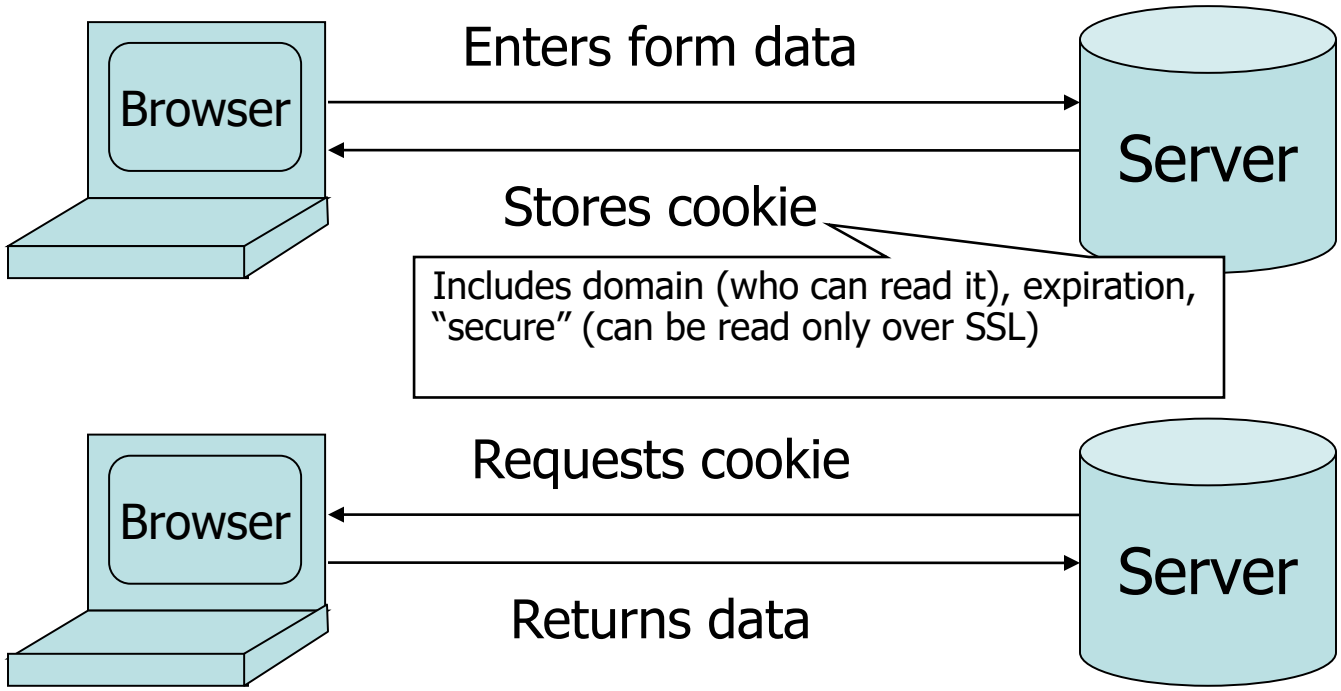
```
HTTP/1.0 200 OK
Date: Sun, 21 Apr 1996 02:20:42 GMT
Server: Microsoft-Internet-Information-Server/5.0
Connection: keep-alive
Content-Type: text/html
Last-Modified: Thu, 18 Apr 1996 17:39:05 GMT
Content-Length: 2543

<HTML> Some data... blah, blah, blah </HTML>
```

Data

Storing Info Across Sessions

- A cookie is a file created by an Internet site to store information on your computer



HTTP is a stateless protocol; cookies add state

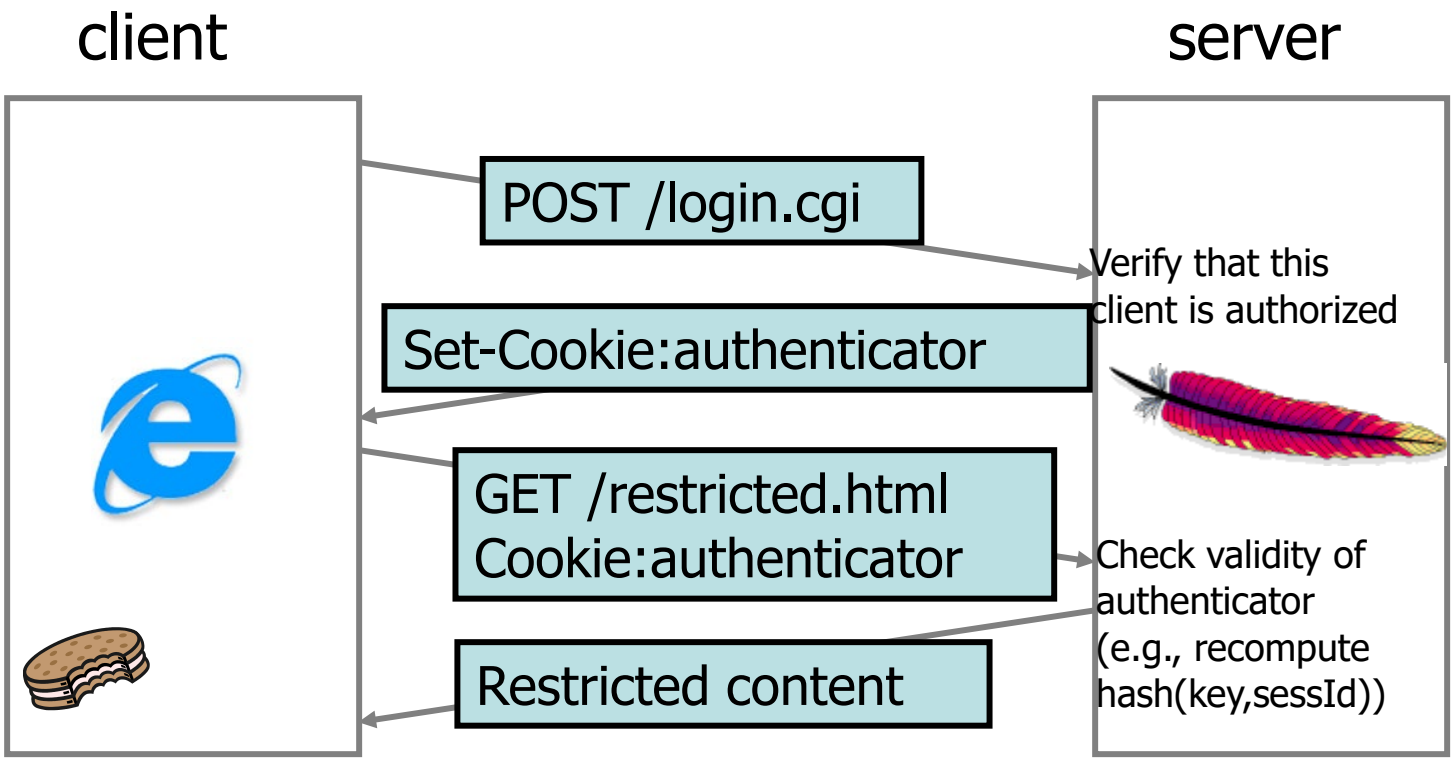
What Are Cookies Used For?

- Authentication
  - Use the fact that the user authenticated correctly in the past to make future authentication quicker
- Personalization
  - Recognize the user from a previous visit
- Tracking
  - Follow the user from site to site; learn his/her browsing behavior, preferences, and so on

Cookie Management

- Cookie ownership
  - Once a cookie is saved on your computer, only the website that created the cookie can read it (Same-origin Policy)
- Variations
  - Temporary cookies
    - Stored until you quit your browser
  - Persistent cookies
    - Remain until deleted or expire
  - Third-party cookies
    - Originates on or sent to another website

Typical Session with Cookies



Authenticators must be unforgeable and tamper-proof

(malicious client shouldn't be able to compute his own or modify an existing authenticator)

Web Applications

- Online banking, shopping, government, etc. etc.
- Website takes input from user, interacts with back-end databases and third parties, outputs results by generating an HTML page
- Often written from scratch in a mixture of PHP, Java, Perl, Python, C, ASP
- Security is now a key consideration in web app design:
  - Poorly written scripts with inadequate input validation
  - Sensitive data stored in world-readable files
  - Recent push from Visa and Mastercard to improve security of data management (PCI standard)

JavaScript

- Language executed by browser
  - Can run before HTML is loaded, before page is viewed, while it is being viewed or when leaving the page
- Often used to exploit other vulnerabilities
  - Attacker gets to execute some code on user's machine
  - Cross-scripting: attacker inserts malicious JavaScript into a Web page or HTML email; when script is executed, it steals user's cookies and hands them over to attacker's site

Scripting

```
<script type="text/javascript">
    function whichButton(event) {
    if (event.button==1) {
        alert("You clicked the left mouse button!") }
    else {
        alert("You clicked the right mouse button!")
    }}
</script>
…
<body onMouseDown="whichButton(event)">
…
</body>
```

Script defines a page-specific function

Function gets executed when some event happens (onLoad, onKeyPress, onMouseMove…)

JavaScript Security Model

- Script runs in a “sandbox”
  - Not allowed to access files or talk to the network
- Same-origin policy
  - Can only read properties of documents and windows from the same server, protocol, and port
  - If the same server hosts unrelated sites, scripts from one site can access document properties on the other
- User can grant privileges to signed scripts
  - UniversalBrowserRead/Write, UniversalFileRead, UniversalSendMail

Risks of Poorly Written Scripts

- For example, echo user's input

http://naive.com/search.php?term=`"Britney Spears"`

search.php responds with

<html> <title>Search results</title>

<body>You have searched for <?php echo $_GET[term] ?>…
</body>

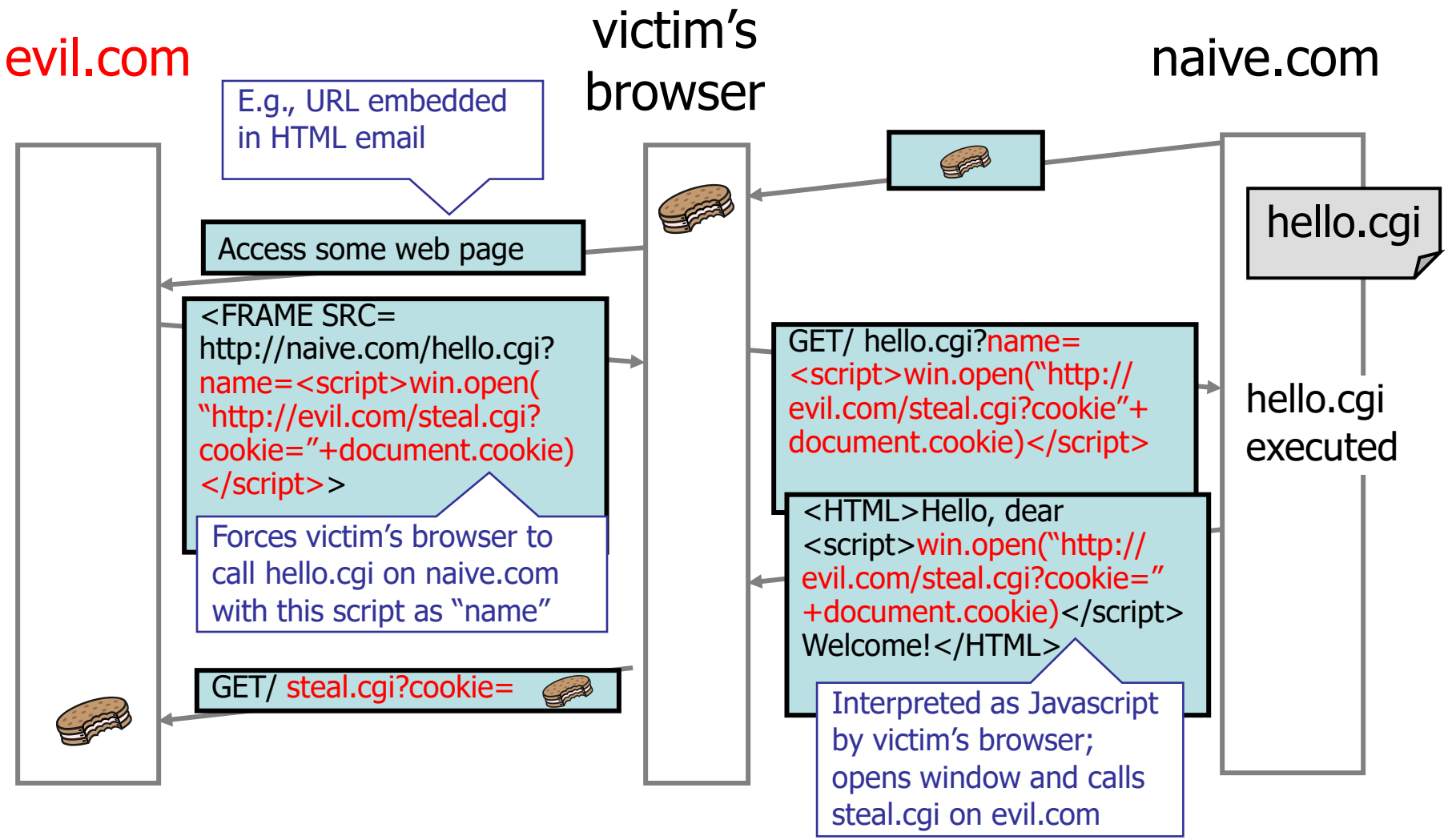
Or

GET/ hello.cgi?name=`Bob`

hello.cgi responds with

<html>Welcome, dear `Bob`</html>

XSS: Cross-Site Scripting



XSS Risks

- XSS is a form of reflection attack
  - User is tricked into visiting a badly written website
  - A bug in website code causes it to display the attack script and the user's browser to execute arbitrary operations contained in the attack script
- Can transmit user's private data to attacker
  - E.g., encode it in a URL request to attacker's site
- Can change contents of the affected website
  - Show bogus information, request sensitive data
- Can cause user's browser to attack other websites

MySpace Worm (1)

http://namb.la/popular/tech.html

- Users can post HTML on their MySpace pages
- MySpace does not allow scripts in users' HTML
  - No <script>, <body>, onclick, <a href=javascript://>
- … but does allow tags for CSS. K00L!
  -
- But MySpace will strip out "javascript"
  - Use "java<NEWLINE>script" instead
- But MySpace will strip out quotes
  - Convert from decimal instead:
    alert('double quote: ' + String.fromCharCode(34))

MySpace Worm (2)

http://namb.la/popular/tech.html

- *"There were a few other complications and things to get around. This was not by any means a straight forward process, and none of this was meant to cause any damage or piss anyone off. This was in the interest of..interest. It was interesting and fun!"*
- Started on "samy" MySpace page
- Everybody who visits an infected page, becomes infected and adds "samy" as a friend and hero
- 5 hours later "samy"

  has 1,005,831 friends

  – Was adding 1,000 friends per second at its peak

Where Malicious Scripts Live

- Hide script in user-created content
  - Social sites (e.g., MySpace), blogs, forums, wikis
- When visitor loads the page, webserver displays the content and visitor's browser executes script
  - Many sites try to filter out scripts from user content, but this is difficult (example: samy worm)
- Another reflection trick
  - Some websites parse input from URL

http://cnn.com/login?URI="><script>AttackScript</script>

  - Use phishing email to drive users to this URL
  - Similar: malicious DOM (client parses bad URL)

Attack code does not appear in HTML sent over network

Other Sources of Malicious Scripts

- Scripts embedded in webpages
  - Same-origin policy doesn’t prohibit embedding of third-party scripts
  - Ad servers, mashups, etc.

Preventing Cross-Site Scripting

- Preventing injection of scripts into HTML is hard!
  - Blocking “<” and “>” is not enough
  - Event handlers, stylesheets, encoded inputs (%3C), etc.
  - phpBB allowed simple HTML tags like <b>

    <b c=“>” onmouseover=“script” x=“<b ”>Hello<b>
- Any user input <u>must</u> be preprocessed before it is used inside HTML
  - In PHP, htmlspecialchars(string) will replace all special characters with their HTML codes
    - ‘ becomes ' “ becomes " & becomes &
  - In ASP.NET, Server.HtmlEncode(string)

User Data in SQL Queries

- set UserFound=execute(

  SELECT * FROM UserTable WHERE

  username=' ” & form(“user”) & “ ' AND

  password=' ” & form(“pwd”) & “ ' ”);

  – User supplies username and password, this SQL query checks if user/password combination is in the datab

Only true if the result of SQL query is not empty, i.e., user/pwd is in the database

- If not UserFound.EOF

  Authentication correct

else Fail

SQL Injection

- User gives username ' OR 1=1 --
- Web server executes query

```
set UserFound=execute(
    SELECT * FROM UserTable WHERE
    username='' OR 1=1 -- … );
```

Always true!

Everything after -- is ignored!

- This returns the entire database!
- UserFound.EOF is always false; authentication is always “correct”

Exploit

User appends this to the URL:

&new_pass=badPwd%27%29%2c

user_level=%27103%27%2cuser_aim=%28%27

This sets $new_pass to badPwd'), user_level='103', user_aim=('

SQL query becomes

UPDATE users SET

user_password=md5('badPwd')

user_level='103', user_aim=('???????')

WHERE user_id='userid'

with superuser privileges

User's password is set to 'badPwd'

Cookie Authentication: Issues

- Users logs into bank.com, forgets to sign off
  - Session cookie remains in browser state
- User then visits a malicious website containing

```
<form  name=BillPayForm
action=http://bank.com/BillPay.php>
<input  name=recipient  value=badguy> …
<script> document.BillPayForm.submit(); </script>
```

- What happens?

Data export

- Many ways to send information to other origins

```
<form action="http://www.b.com/">
    <input name="data" type="hidden"
value="hello">
</form>
```

```
<img src="http://www.b.com/?data=hello"/>
```

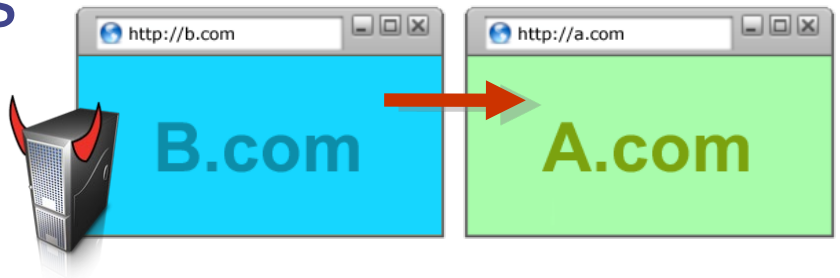
- No user involvement required
- Cannot read back response

Classic CSRF/XSRF attack

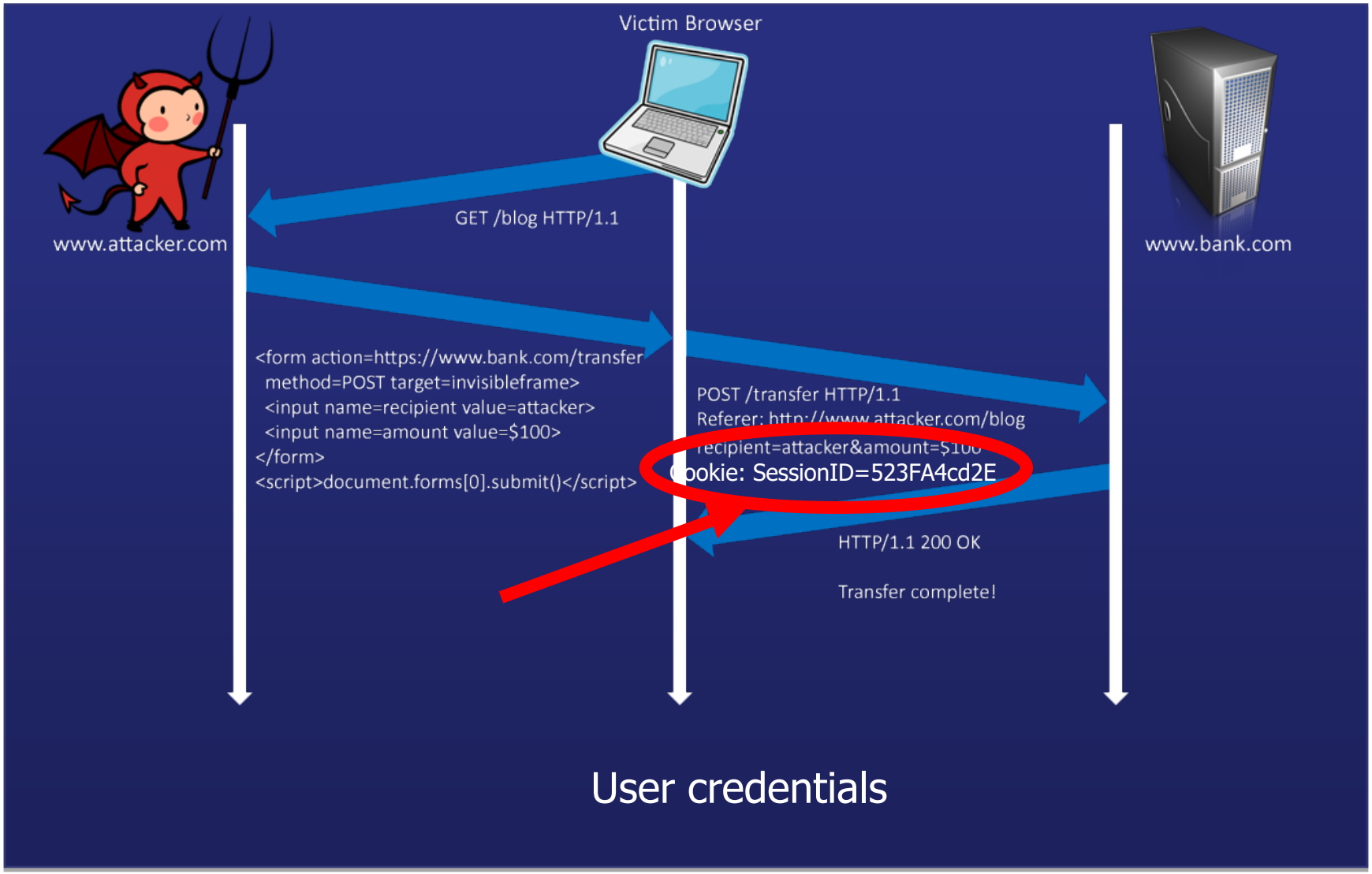
- User visits victim site site
  - Logs in

- User loads attacker's site
  - Or encounters attacker's iframe on another site

- Attacker sends HTTP requests to victim
  - Victim site assumes requests originate from itself



Classic CSRF Attack



CSRF Defenses

- Secret Validation Token


```
<input type=hidden value=23a3af01b>
```

- Referer Validation


```
Referer: http://www.facebook.com/home.php
```

- Custom HTTP Header

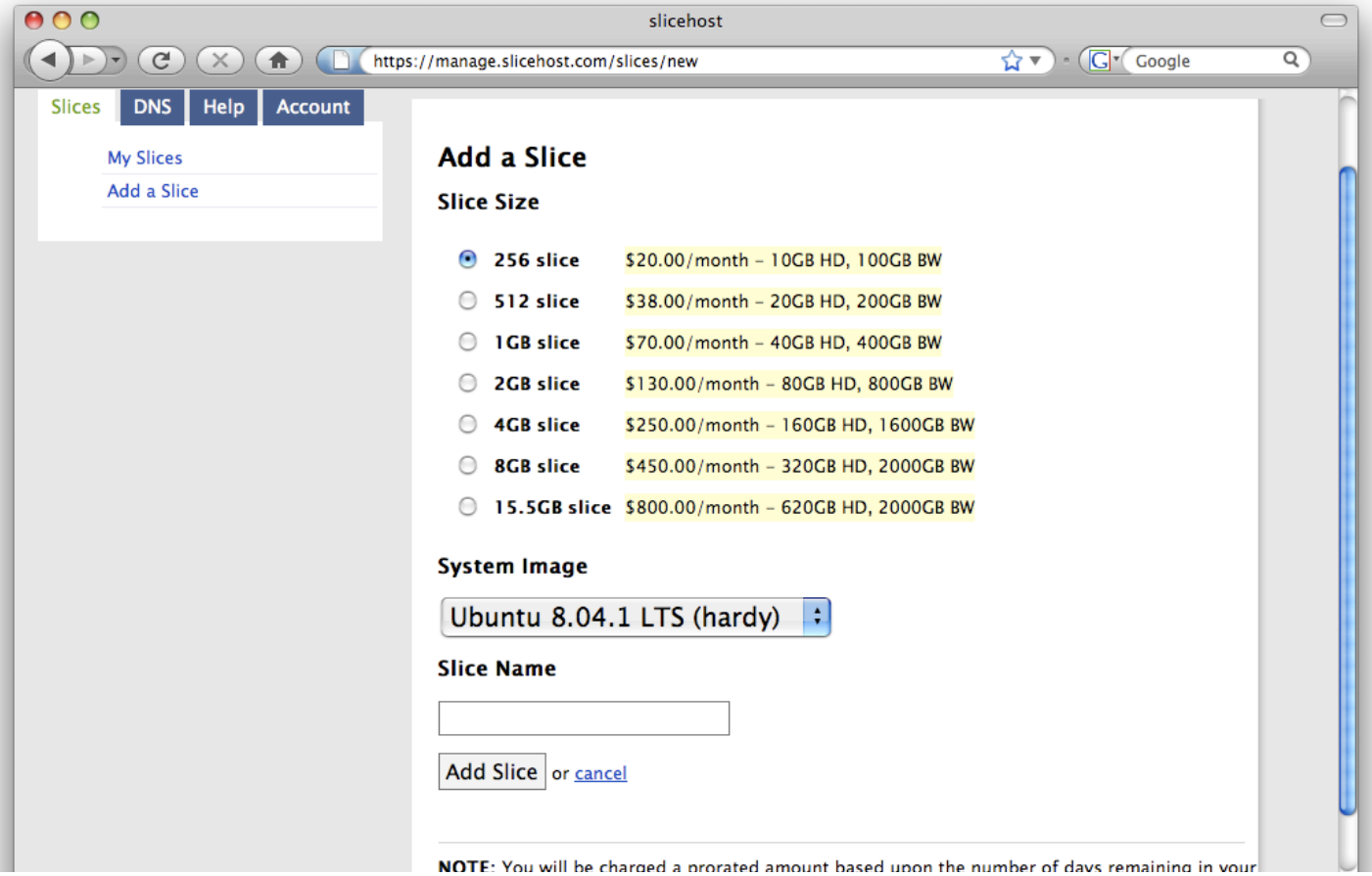
```
X-Requested-By: XMLHttpRequest
```

Secret Token Validation

- Requests include a hard-to-guess secret
  - Unguessability substitutes for unforgeability
- Variations
  - Session identifier
  - Session-independent token
  - Session-dependent token
  - HMAC of session identifier

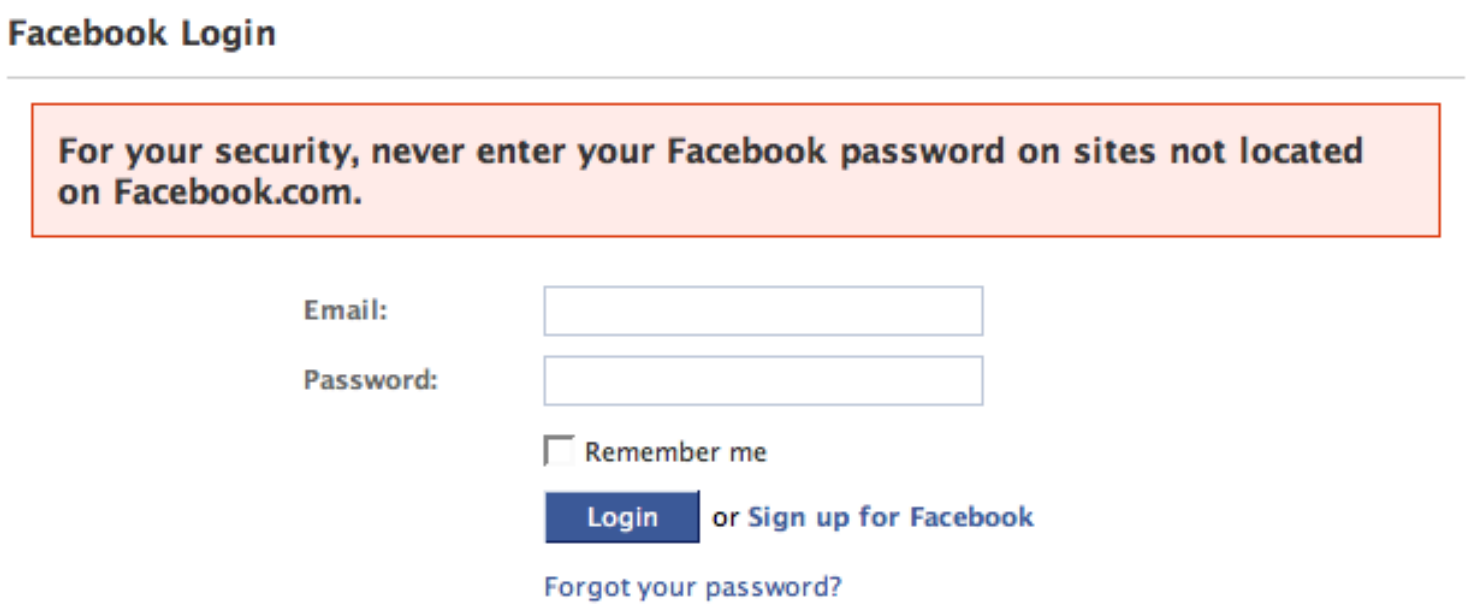
See "Robust Defenses for Cross-Site Request Forgery" for a comparison of these options.

Secret Token Validation




```
g:0"><input name="authenticity_token" type="hidden" value="0114d5b35744b522af8643921bd5a3d899e7fbd2" />
="/images/logo.jpg" width='110'></div>
```

Referer Validation



Referer Validation Defense

- HTTP Referer header
  - Referer: http://www.facebook.com/ ✓
  - Referer: http://www.attacker.com/evil.html ✗
  - Referer: ?
- Lenient Referer validation
  - Doesn't work if Referer is missing
- Strict Referer validaton
  - Secure, but Referer is sometimes absent…

Referer Privacy Problems

- Referer may leak privacy-sensitive information

  `http://intranet.corp.apple.com/projects/iphone/competitors.html`

- Common sources of blocking:
  - Network stripping by the organization
  - Network stripping by local machine
  - Stripped by browser for HTTPS -> HTTP transitions
  - User preference in browser
  - Buggy user agents
- Site cannot afford to block these users

Lenient Validation Vulnerability

- My site uses HTTPS, am I safe?
- Problem: Browsers do not append Referer if the source of the request is not an HTTP page

```
ftp://attacker.com/attack.html
data:text/html,<html>…</html>
javascript:'<html>…</html>'
```

Strict Validation Problems

- Some sites allow users to post forms
  - XSS sanitization doesn't include <form>
  - These sites need another defense
- Many sites allow users to post hyperlinks
  - Solution: Respect HTTP verb semantics
  - GET requests have no side effects
  - POST requests can change state

Custom Header Defense

- XMLHttpRequest is for same-origin requests
  - Can use setRequestHeader within origin
- Limitations on data export format
  - No setRequestHeader equivalent
  - XHR2 has a whitelist for cross-site requests
- Issue POST requests via AJAX:

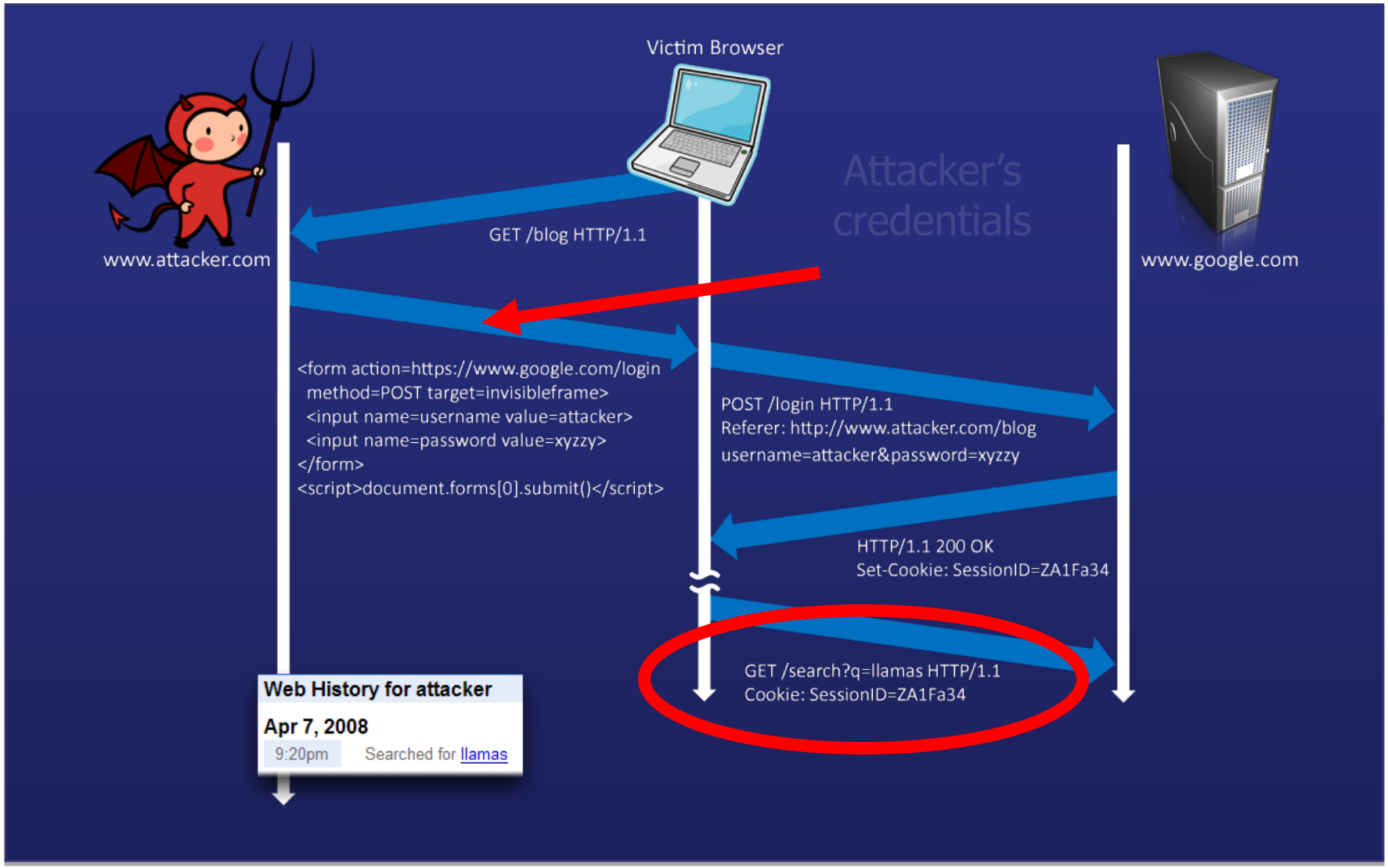
```
X-Requested-By: XMLHttpRequest
```

- Doesn't work across domains

Broader view of CSRF

- Abuse of cross-site data export feature
  - From user's browser to honest server
  - Disrupts integrity of user's session
- Why mount a CSRF attack?
  - Network connectivity
  - Read browser state
  - Write browser state
- Not just "session riding"

Login CSRF


Victim Browser
www.attacker.com
www.google.com
Attacker's credentials
GET /blog HTTP/1.1
<form action=https://www.google.com/login
method=POST target=invisibleframe>
<input name=username value=attacker>
<input name=password value=xyzzy>
</form>
<script>document.forms[0].submit()</script>
POST /login HTTP/1.1
Referer: http://www.attacker.com/blog
username=attacker&password=xyzzy
HTTP/1.1 200 OK
Set-Cookie: SessionID=ZA1Fa34
GET /search?q=llamas HTTP/1.1
Cookie: SessionID=ZA1Fa34
Web History for attacker
Apr 7, 2008
9:20pm
Searched for llamas

Payments Login CSRF

FAQ - Sura-Sura Kanji Quizzer - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://www.kanjiquizzer.com/help/faq.php

Google

Quizzer provides an interface for studying these images.

Wow! This site is so cool! How can I show my appreciation?

Sura-Sura Kanji Quizzer is supported by banner advertisements, but you can also support Sura-Sura Kanji Quizzer via PayPal donation:

PayPal Donate

How does the quizzer choose which kanji to display?

The displayed kanji is chosen at random from among the active kanji. Special effort is taken to avoid displaying the same kanji twice in a row. It might still happen, however, if only one kanji is active.

How should I use the Sura-Sura Kanji Quizzer service?

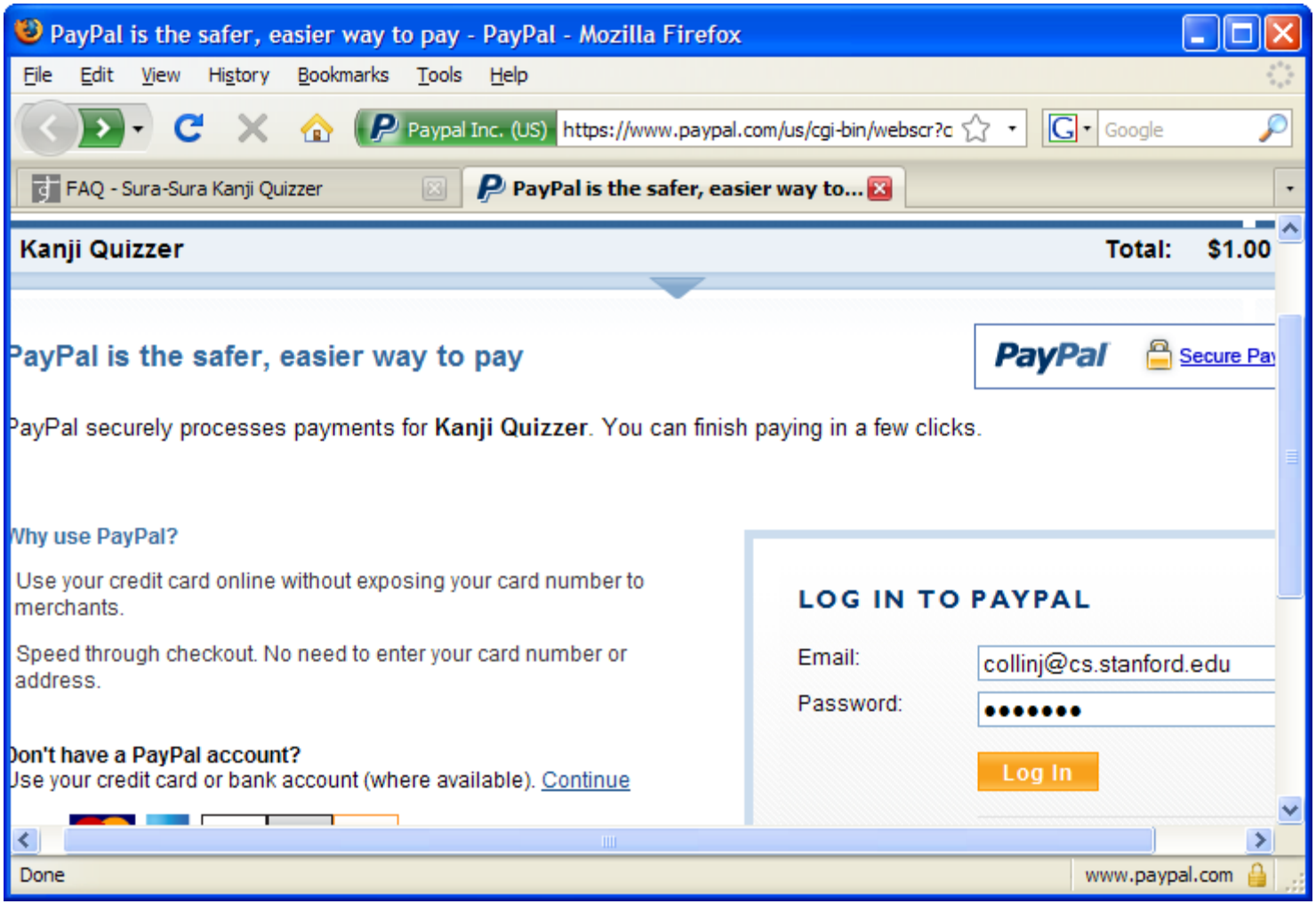
All we ask is that you use the quizzer honestly. Bad data will make the statistics less useful.

How does the quizzer calculate the "success rate" of a user?

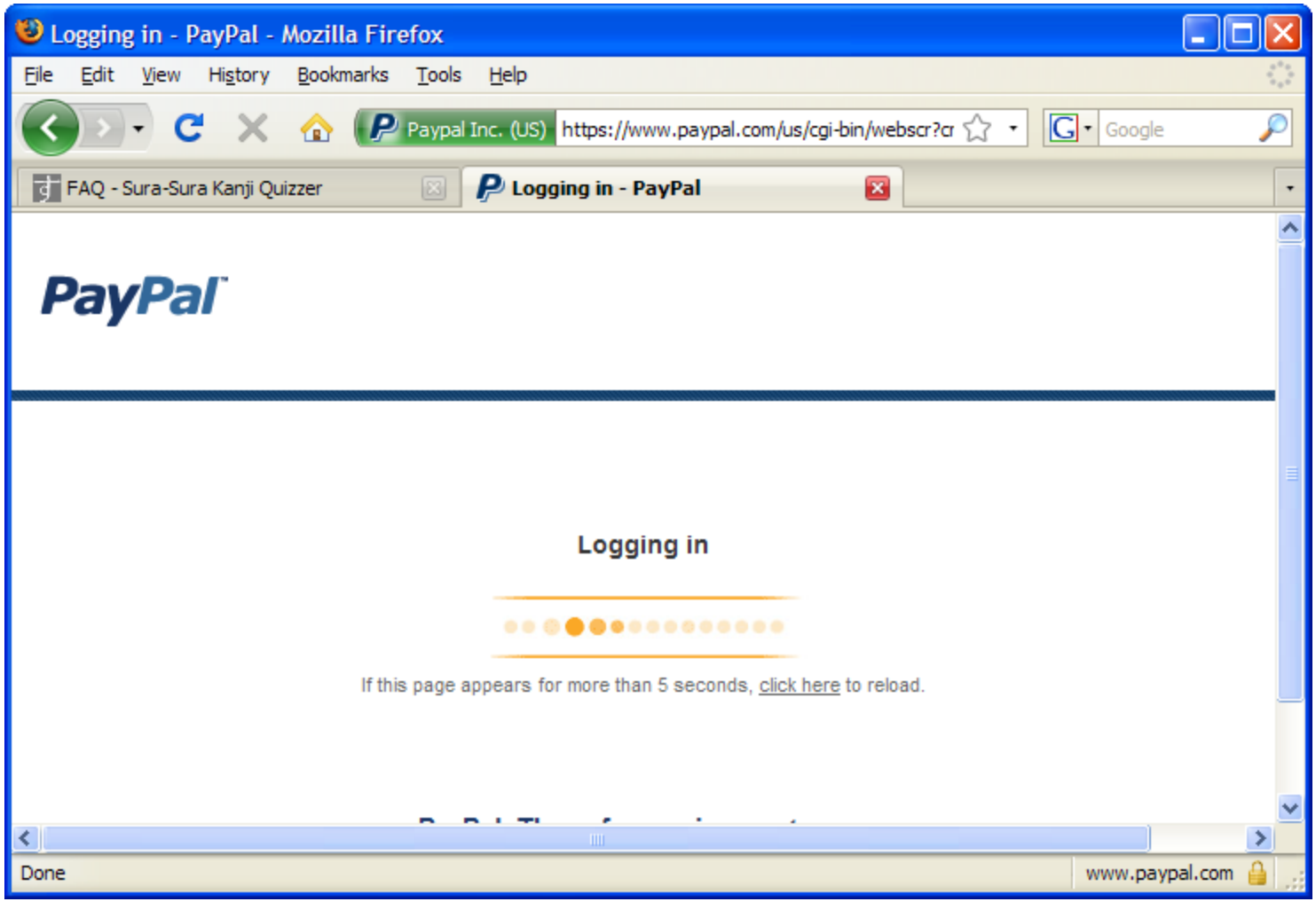
The formula is (Times Succeeded) / (Times Viewed). If you view a kanji but do not click the "Success" button (for example, if you click a link to some other part of the site), that counts against your success rate. Please do not worry too much about

Done

Payments Login CSRF



Payments Login CSRF



Payments Login CSRF

Add a Bank Account in the United States - PayPal - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Paypal Inc. (US) https://www.paypal.com/us/cgi-bin/webscr?dispatch=5885d80a13

FAQ - Sura-Sura Kanji Quizzer | Add a Bank Account in the United...

Log Out | Help | Security Center Search

PayPal

My Account | Send Money | Request Money | Merchant Services | Auction Tools | Products & Services

Add a Bank Account in the United States

Secure Transaction

PayPal protects the privacy of the your financial information regardless of your payment source. This bank account will become the default funding source for most of your PayPal payments, however you may change this funding source when you make a payment. Review our education page to learn more about PayPal policies and your payment-source rights and remedies.

The safety and security of your bank account information is protected by PayPal. We protect against unauthorized withdrawals from your bank account to your PayPal account. Plus, we will notify you by email whenever you deposit or withdraw funds from this bank account using PayPal.

Country: United States

*Bank Name:

Account Type: Checking / Savings

U.S. Check Sample

Routing Number (9 digits) | Check# | Account Number (3-17 digits)

*Routing Number: (9 digits)

Is usually located between the symbols on your check.

*Account Number: (3-17 digits)

Typically comes before the symbol. Its exact location and number of digits varies from bank to bank.

*Re-enter Account Number:

Continue Cancel

Done www.paypal.com

Can browsers help with CSRF?

- Does not break existing sites
- Easy to use
- Hard to misuse
- Allows legitimate cross-site requests
- Reveals minimum amount of information
- Can be standardized

Proposed Approaches

- HTTP Headers
  - Identify the source of requests
  - Change Referer header or add a new Origin header
  - Send more information for POST than GET
  - Experiment: Cross-domain POSTs out of firewall accounted for ~0.0001% of traffic
  - Problem: Unsafe GET requests
  - Problem: Third-party content within an origin
  - Problem: How to handle redirects
- Same-origin-only cookies
  - Doesn't help multi-domain sites: amazon.com and amazon.co.uk
  - These sites could use other defenses

Conclusion

- Server-side defenses are required
  - Secret token validation – use frameworks like Rails
  - Referer validation – works over HTTPS
  - Custom headers – for AJAX
- No easy solution
  - User does not need to have an existing session for attacks to work
  - Hard to retrofit existing applications with defenses