<input type="password"> *must die!*

Daniel R. Sandler and Dan S. Wallach

W2SP 2008: Web 2.0 Security & Privacy 2008

May 22, 2008

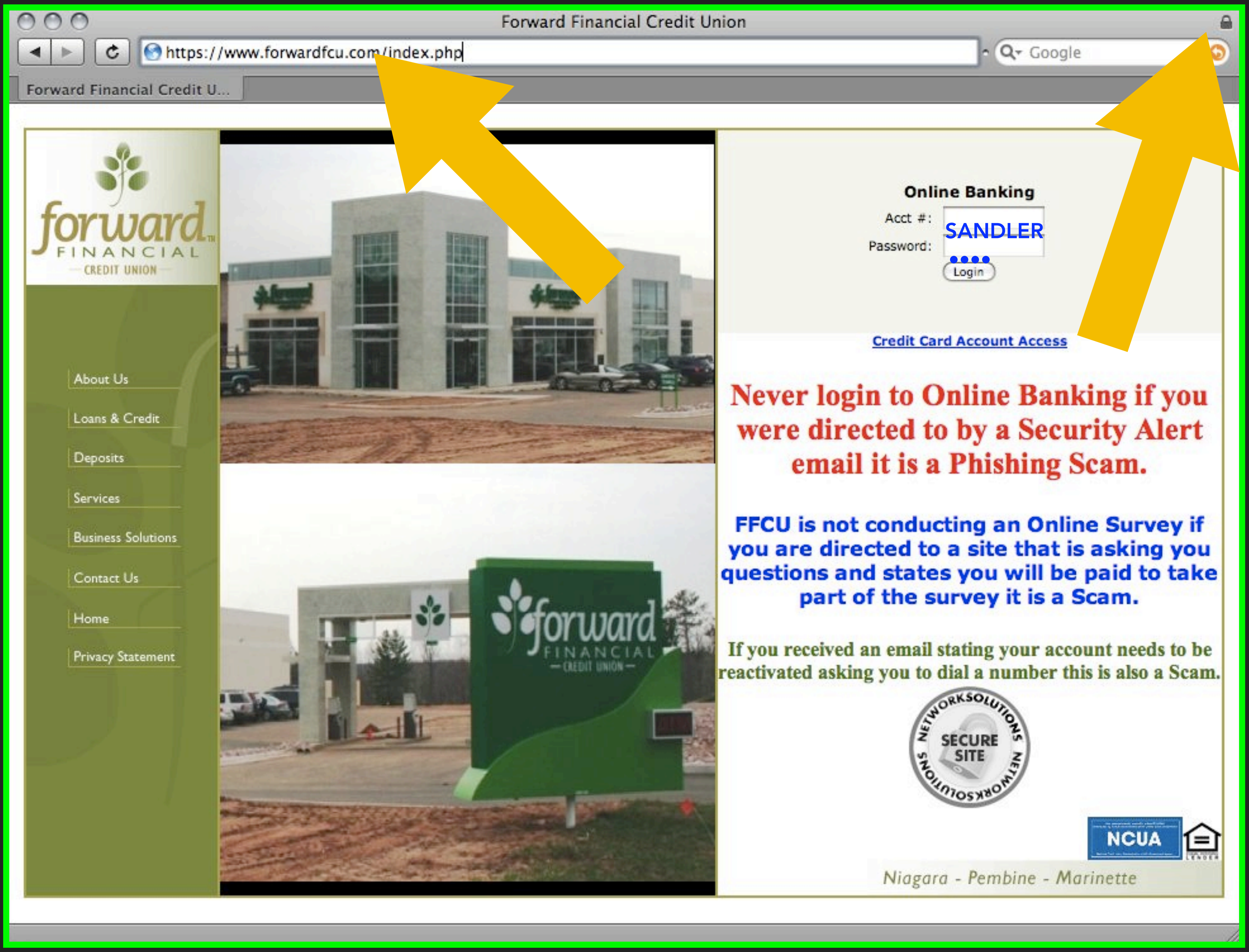
PROBLEM

(SOLUTION)

This talk is about a big problem.

And a small solution.

And the problem is this guy: the HTML password input.


Forward Financial Credit Union
https://www.forwardfcu.com/index.php
Google
Forward Financial Credit U...
forward FINANCIAL CREDIT UNION
About Us
Loans & Credit
Deposits
Services
Business Solutions
Contact Us
Home
Privacy Statement
Online Banking
Acct #:
Password:
SANDLER
Login
Credit Card Account Access
SECURE SITE
NETWORKSOLUTIONS
NCUA
Niagara - Pembine - Marinette

Never login to Online Banking if you were directed to by a Security Alert email it is a Phishing Scam.

FFCU is not conducting an Online Survey if you are directed to a site that is asking you questions and states you will be paid to take part of the survey it is a Scam.

If you received an email stating your account needs to be reactivated asking you to dial a number this is also a Scam.

Should I log in here?

(I got this URL out of my spam quarantine earlier this week.)

Now, this looks pretty bogus to us. We see that the security indicators are missing @. @

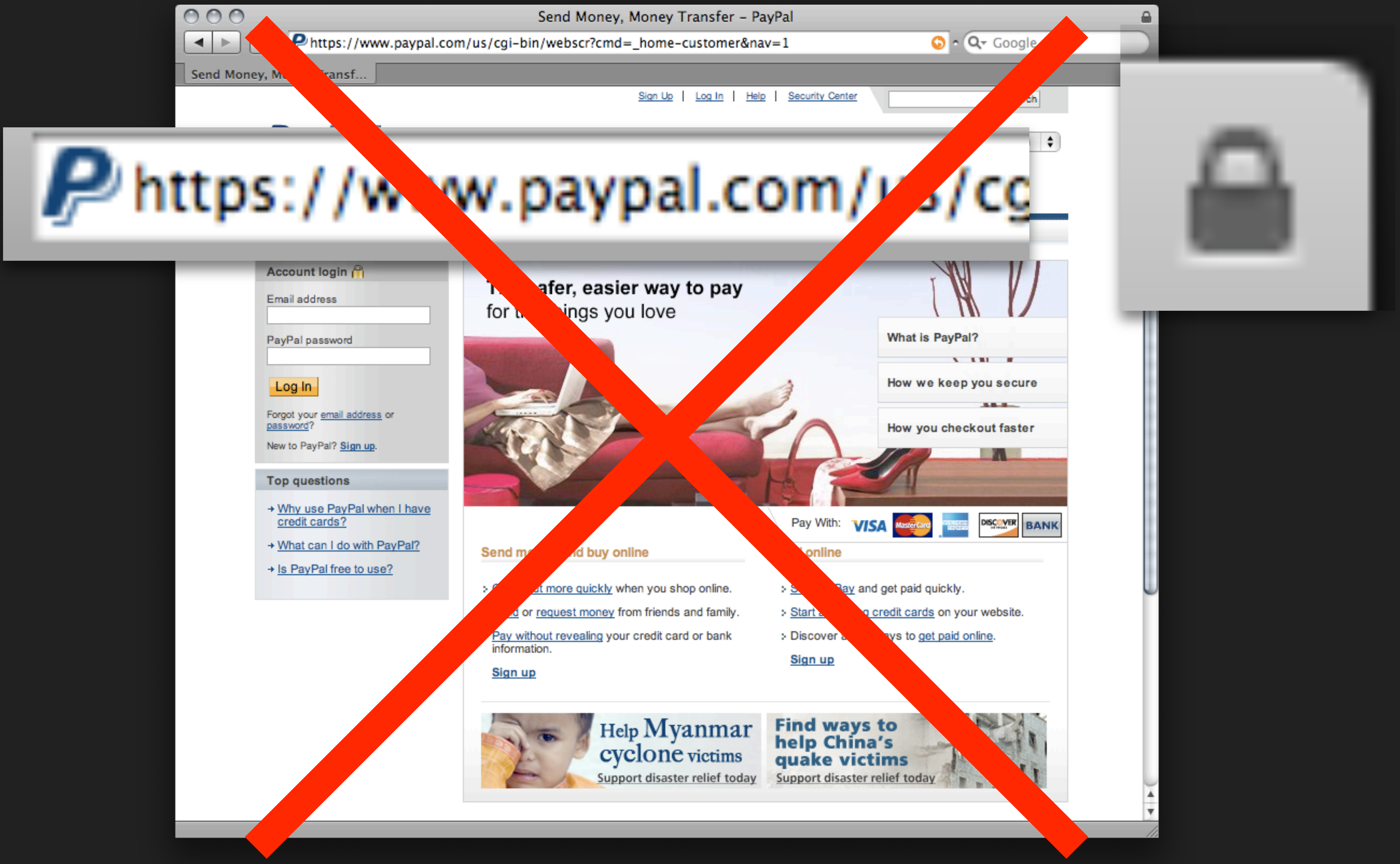
It also looks a little simplistic—like someone knocked it together in DreamWeaver—but just for reference here's the real site for this bank: @

I think the phishers actually did a slightly better job.

OK, so, here's the problem. Let's say you start typing your login credentials before really scrutinizing this page.

The user has an expectation that information you type into a form is temporary until you click Submit—giving you time to consider the page and make sure it's safe.

Unfortunately, with JavaScript it is trivially easy to steal keystrokes from your own page by adding an onkeydown handler.


Send Money, Money Transfer - PayPal
https://www.paypal.com/us/cgi-bin/webscr?cmd=_home-customer&nav=1
https://www.paypal.com/us/cg
Account login
Log In
Top questions
afer, easier way to pay
What is PayPal?
How we keep you secure
How you checkout faster
Help Myanmar cyclone victims
Find ways to help China's quake victims

How about this one?

@ Security indicators are OK.

@ Nope, sorry, this one just stole all your money.

How?

security indicators: OK

no DNS attacks in effect

submitting the login form produced no warnings

paypal.com home page (excerpts)

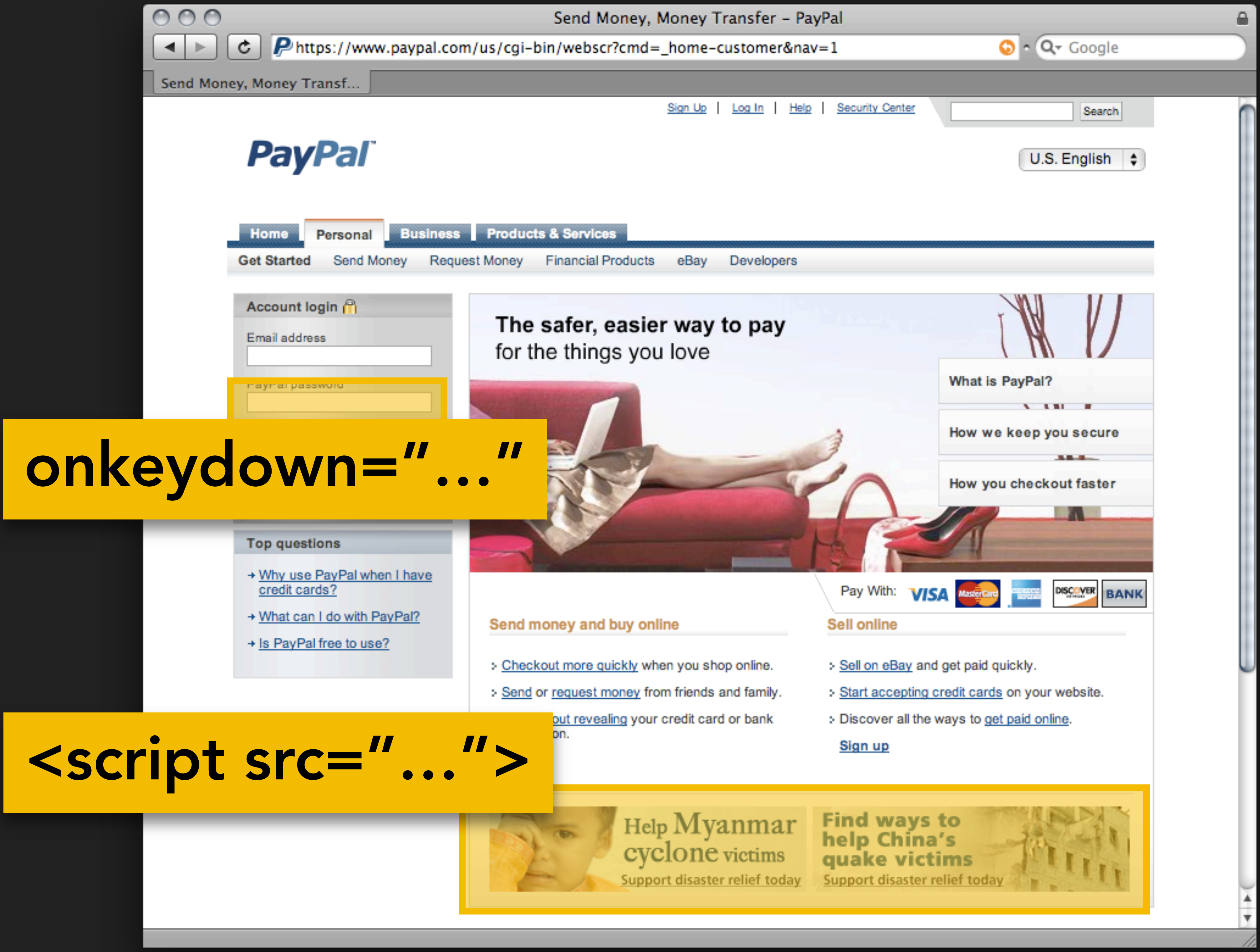
```
<input autocomplete="off" type="password" id="login_password"
name="login_password" value="">
```

```
<SCRIPT LANGUAGE="JavaScript">
ord=Math.random();
ord=ord*10000000000000000000;
document.write('<SCR' + 'IPT LANGUAGE="JavaScript"
SRC="https://ad.doubleclick.net/adj/paypal.us/IndividualHome-
outside;lang=en_us;tile=1;sz=520x70;ord='+ord+'?"><\/SCR' +
'IPT>');
</SCRIPT>
```

The all-seeing eye of JavaScript

Third-party code has full rights

…including the right to install a password keylogger


Send Money, Money Transfer - PayPal
https://www.paypal.com/us/cgi-bin/webscr?cmd=_home-customer&nav=1
Google
Sign Up | Log In | Help | Security Center
Search
PayPal
U.S. English
Home
Personal
Business
Products & Services
Get Started
Send Money
Request Money
Financial Products
eBay
Developers
Account login
Email address
PayPal password
Top questions
Why use PayPal when I have credit cards?
What can I do with PayPal?
Is PayPal free to use?
The safer, easier way to pay for the things you love
What is PayPal?
How we keep you secure
How you checkout faster
Pay With:
BANK
Send money and buy online
Sell online
Help Myanmar cyclone victims
Support disaster relief today
Find ways to help China's quake victims
Support disaster relief today

Checkout more quickly when you shop online.
Send or request money from friends and family.
Sell on eBay and get paid quickly.
Start accepting credit cards on your website.
Discover all the ways to get paid online.
Sign up

onkeydown="..."

<script src="...">

So, in this example, some unscrupulous DoubleClick employee had decided to start stealing PayPal passwords.

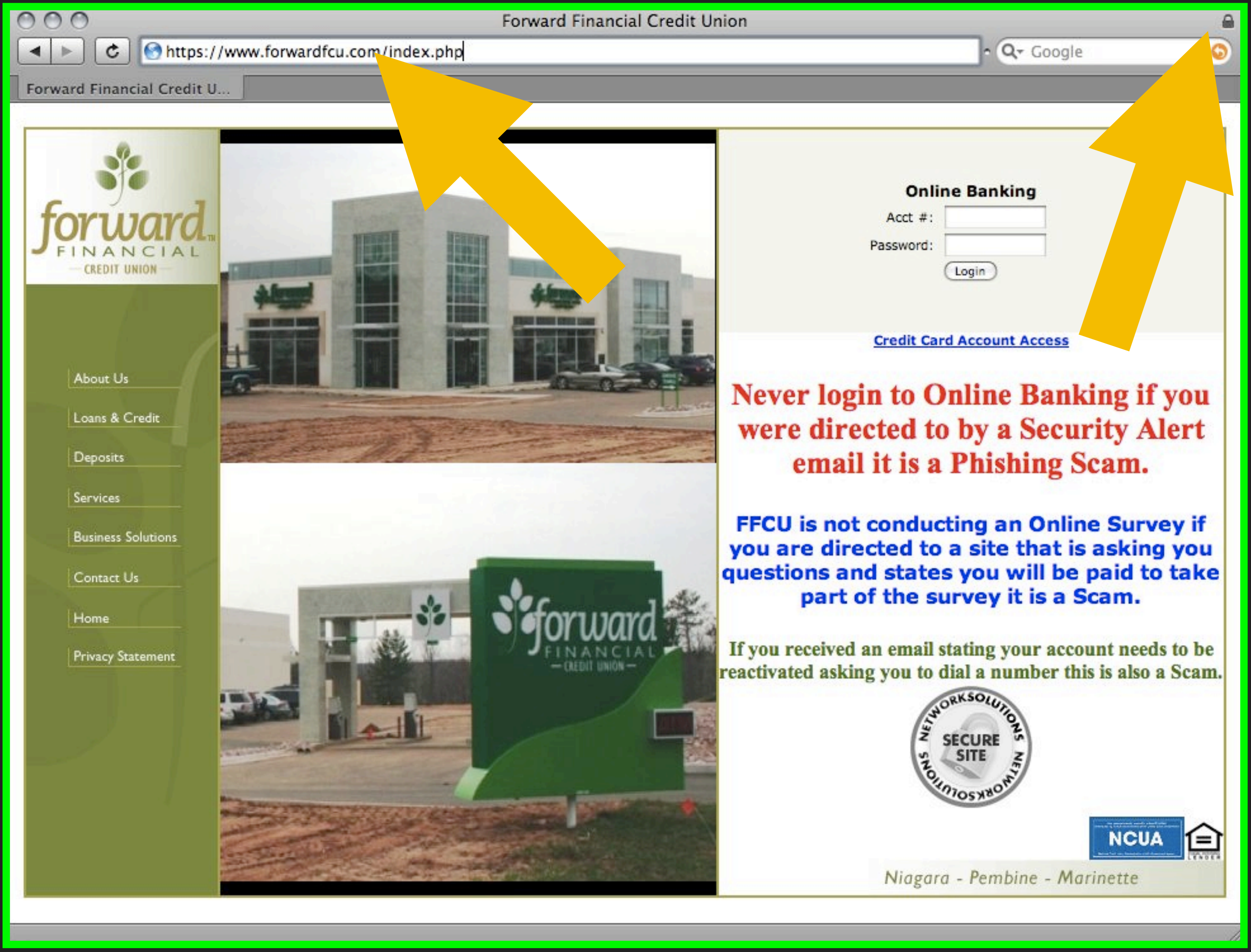
Related problem: security indicators don’t work

They didn’t work in this case due to JS tricks

They don’t even work in the general case

[“Emperor” study due to Schechter et al., Oakland ’07]

Far away (in space)



Forward Financial Credit Union

https://www.forwardfcu.com/index.php

Forward Financial Credit U...

Online Banking

Acct #:

Password:

Login

Credit Card Account Access

Never login to Online Banking if you were directed to by a Security Alert email it is a Phishing Scam.

FFCU is not conducting an Online Survey if you are directed to a site that is asking you questions and states you will be paid to take part of the survey it is a Scam.

If you received an email stating your account needs to be reactivated asking you to dial a number this is also a Scam.

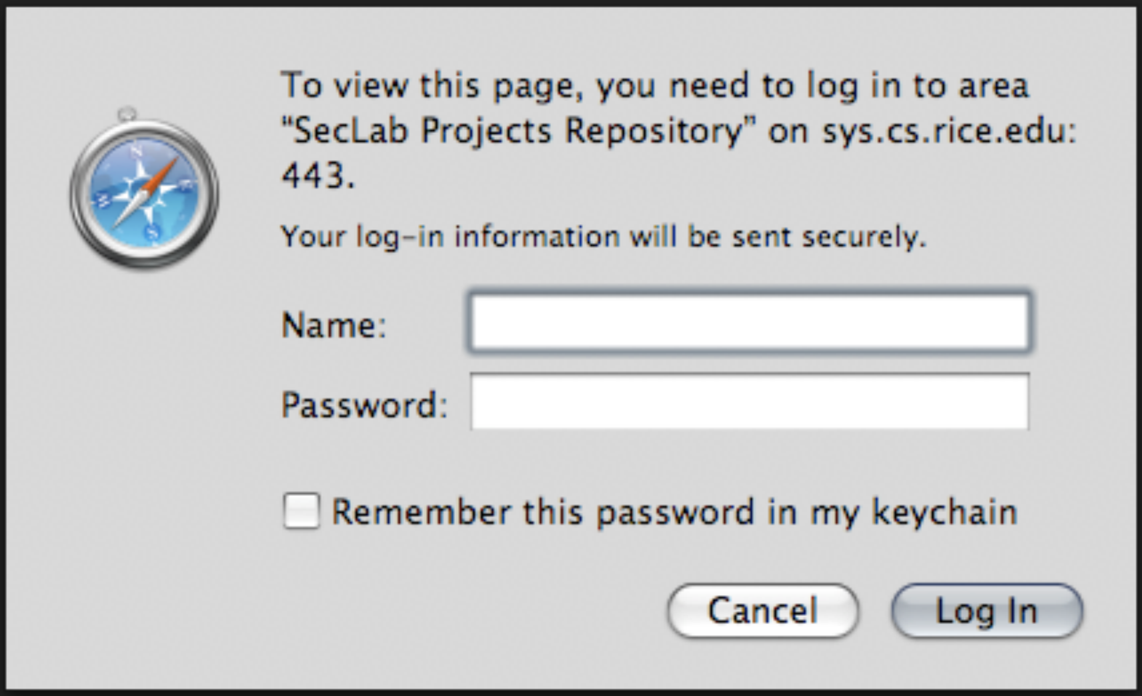
One possible cause:

We make security indicators innocuous so as not to annoy or desensitize the user.

Far away (in time)

1. “This certificate is invalid.”
2. *[time passes…]*
3. user engages in unsafe behavior (e.g. typing passwords)

A related problem: they show up at the wrong **time.**



To view this page, you need to log in to area "SecLab Projects Repository" on sys.cs.rice.edu: 443.

Your log-in information will be sent securely.

Name:

Password:

Remember this password in my keychain

HTTP Authentication?

Password entry performed in browser chrome

Immune to JS; results handed directly to the server

This looks pretty good, right?

HTTP/1.1 401 Authorization Abandoned :(

Used by very few consumer-oriented websites

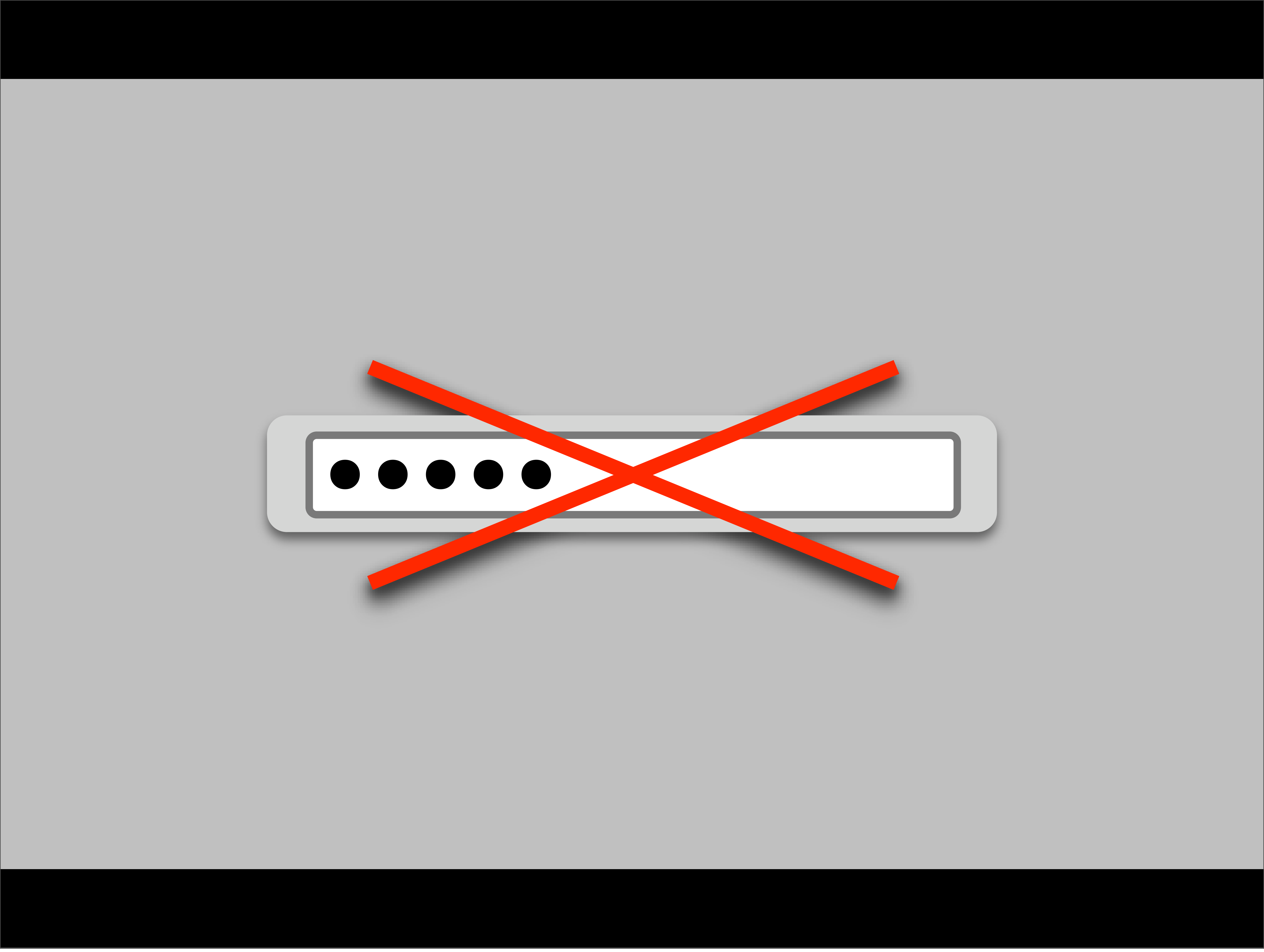
- “Log out” kind of a hack
- Aesthetics

<input type=”password”> is unsafe.

login forms are easy to spoof

it’s vulnerable to JavaScript snooping

security indicators are inadequate



So, it must die.

The user needs a *safe place* to type passwords

Outside DOM/JS (free from snooping)

Hard to spoof

Relevant & visible security indicators

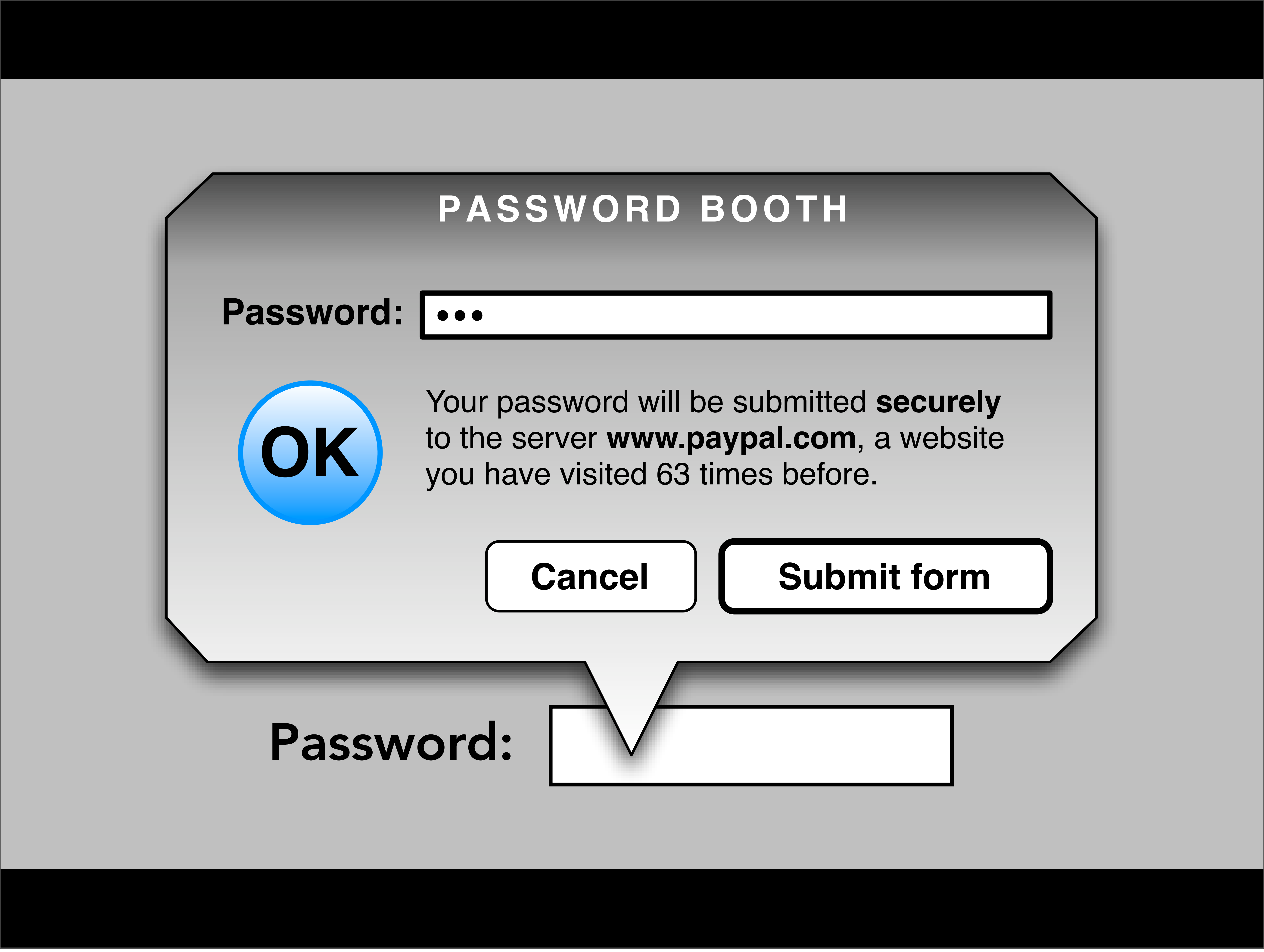
Analogies

Point-of-sale PIN pad

Voting booth

[read slide]

So now I’ll show you our idea for how we could get to a **private password entry** environment by changing only the browser. No website changes.


PASSWORD BOOTH

Password: •••

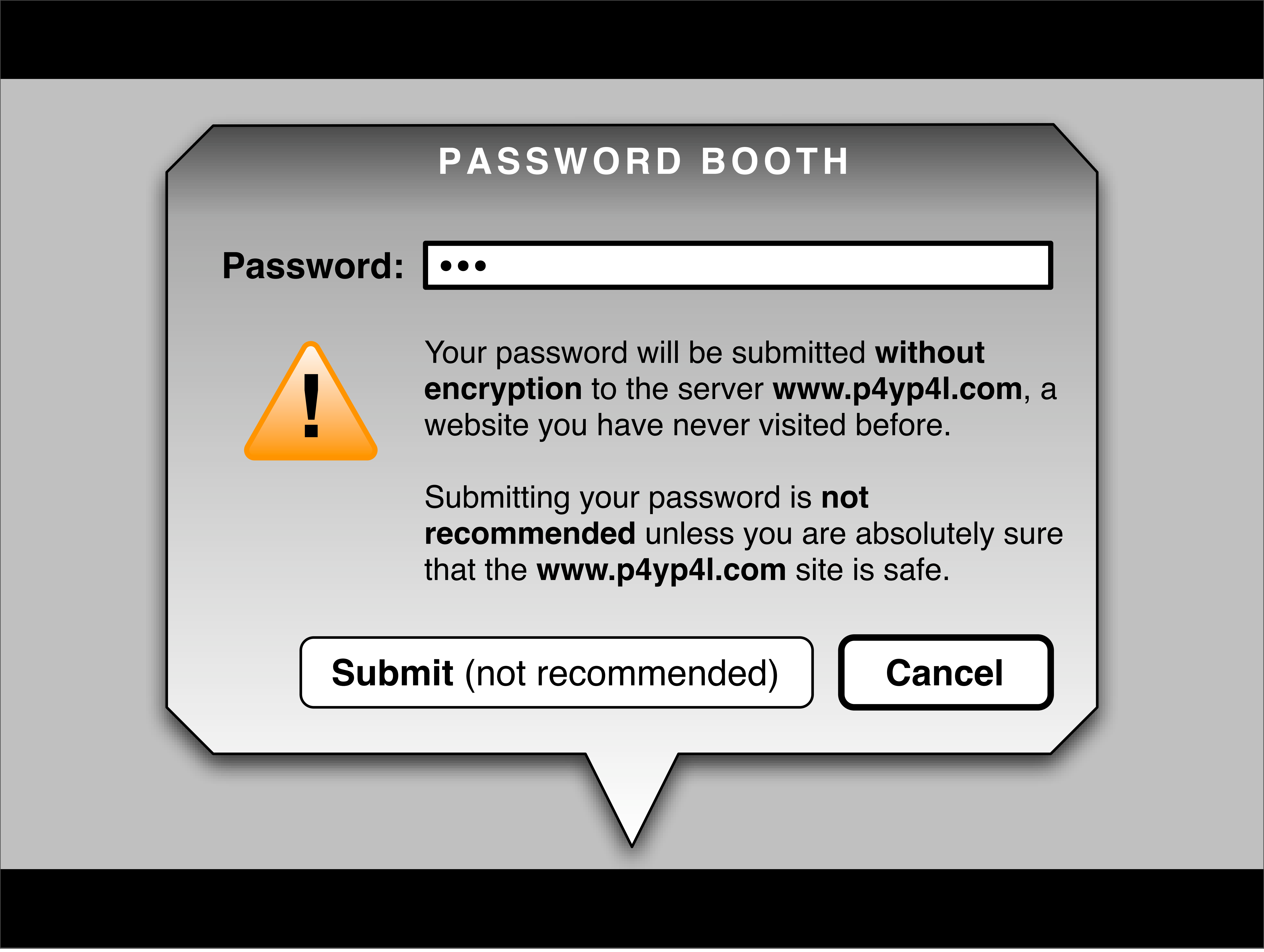
Your password will be submitted **securely** to the server **www.paypal.com**, a website you have visited 63 times before.

You have a website. @ It has a login form. You try to focus the password field to type in it.

You can't. Instead, @ you get a piece of browser chrome.

This is the Password Booth.

It lets you enter your password;
it gives you pertinent security information, like:
• it shows you where that password will be submitted;
• whether SSL is in use.
• whether you've posted to this form before.


PASSWORD BOOTH

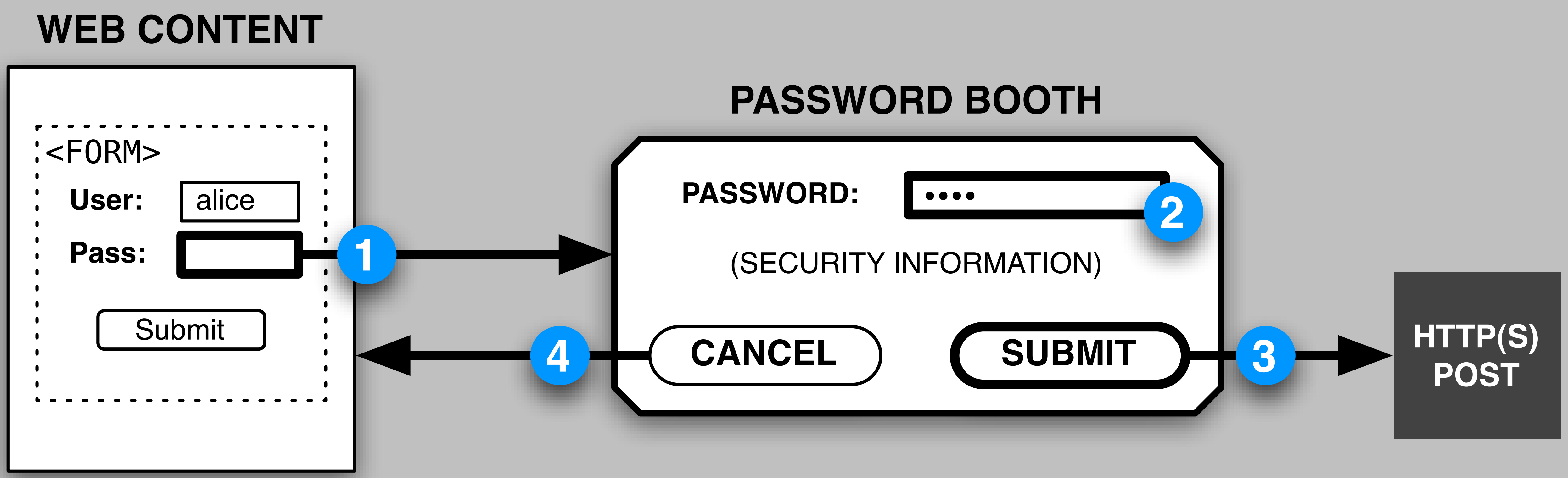
Password: •••

Your password will be submitted **without encryption** to the server **www.p4yp4l.com**, a website you have never visited before.

Submitting your password is **not recommended** unless you are absolutely sure that the **www.p4yp4l.com** site is safe.

Here’s how it might look if things were fishy.

Note that none of these safety heuristics is new, but they’re not currently explained to the user at the right time. This is a step in the right direction.



Note: exiting the booth goes straight to POST

Do not insert the sensitive data back in the <form>!

Password flow is one-way

Let’s take a step back and review the flow of this design.

You focus the password field, Arrow 1.
Enter your password into the booth, at 2.
You can either POST the data, 3,
or cancel (Arrow 4). NOTE: send no data back to the page!

Benefits of the Password Booth

1. Browser-only change
2. One-way password flow
3. Security indicators
   …right where (and when) you need them
4. External to JS
5. **Mandatory!**

Why mandatory?

"Our secure login is down right now. Please type your password."

We are trying to train the user to mistrust this.

@

DON'T BELIEVE IT!

This is entirely possible if the solution is, in any way, opt-in for the website.

We want users to begin to feel **uncomfortable** typing passwords in bare <form>s.

Gotchas

JavaScript-based login forms

Password change forms

An increasing number of **login forms** do their work via JavaScript. If they do not have a non-JavaScript fallback mode, they will be broken in the password booth.

I argue that these forms are inaccessible and therefore broken anyway. Even if you don't agree that they're broken, they still exemplify how JavaScript can do anything with your password **instead** of submitting it to a form.

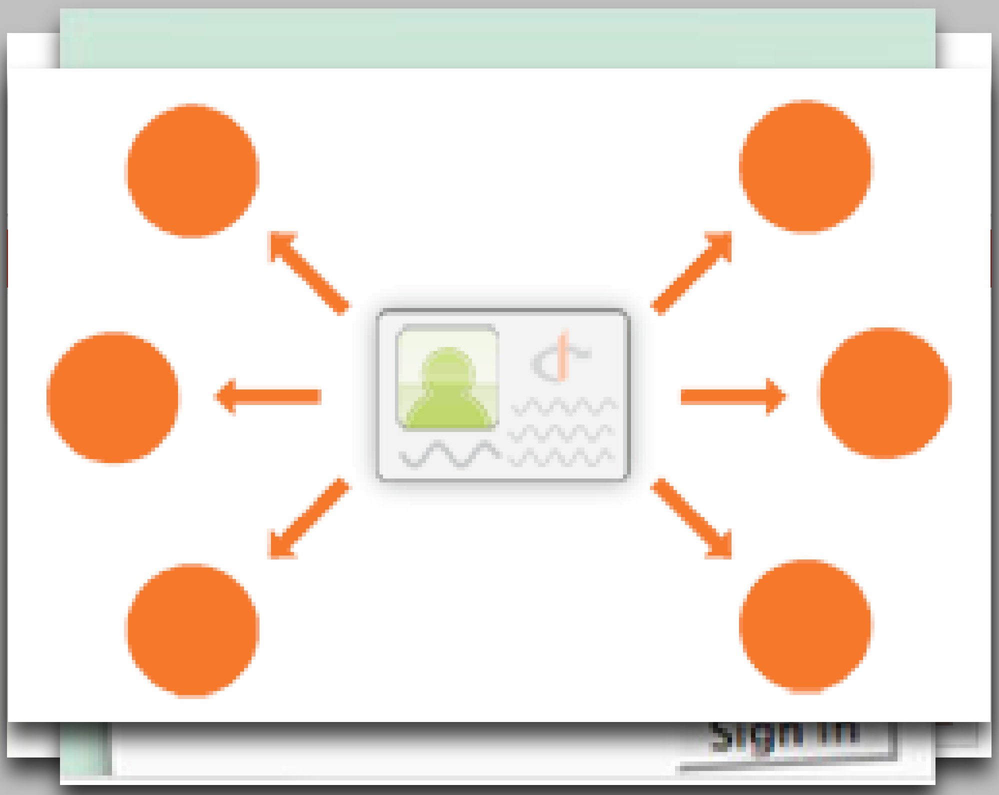
Password change forms have two password fields. Clearly this doesn't work with the booth as currently proposed. However, we can naturally extend it to allow you to enter as many passwords as the form demands.

Related approaches

SiteKey (Bank of America)

Yahoo! Sign-in Seal

OpenID

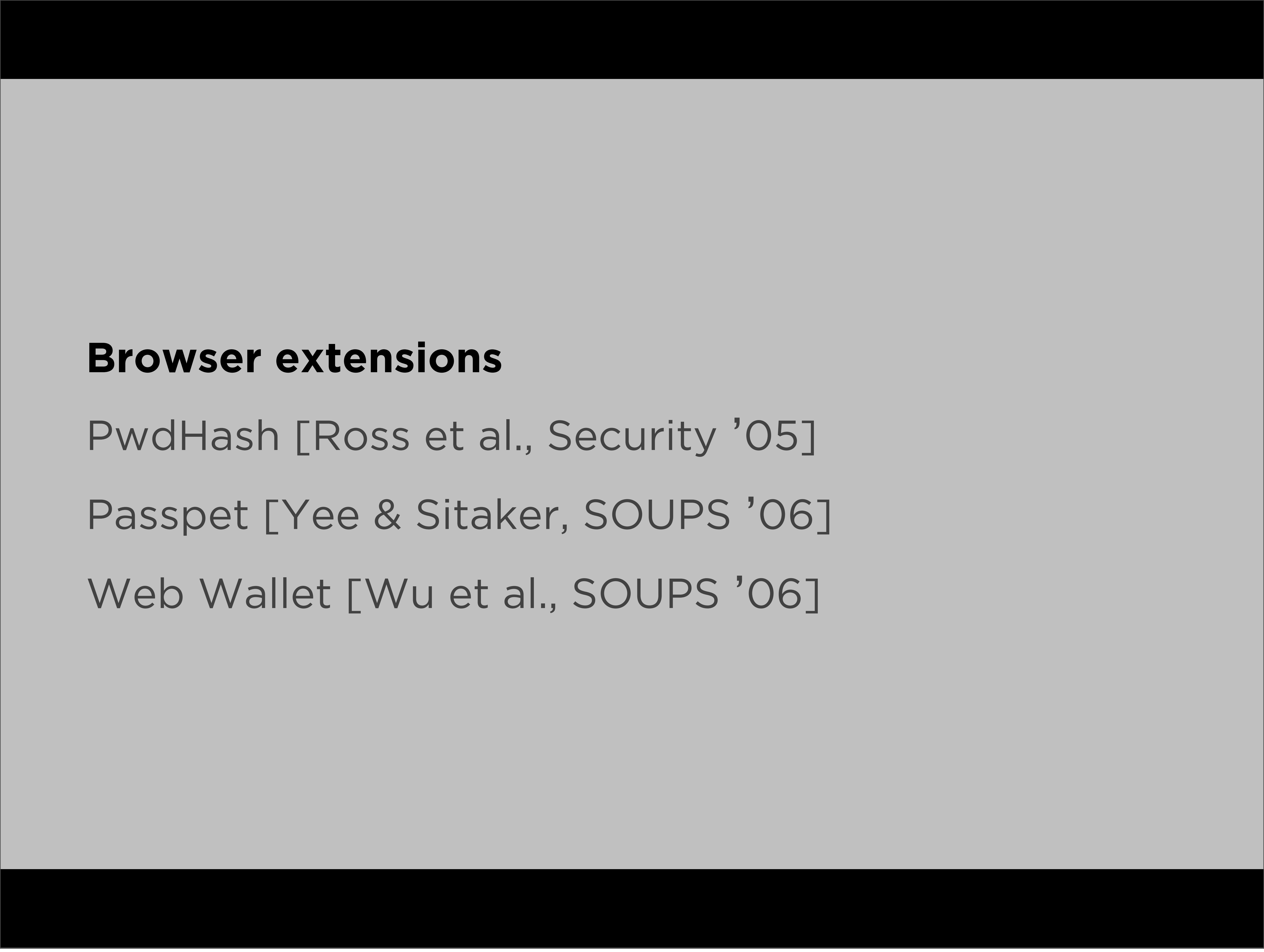


SiteKey @
First, you register a personal image with BofA.
Two-phase login: give your username, they give you back your image so you know you're dealing with the authentic site, and can safely login. **Man in the middle** is trivial with this system.

Yahoo Sign-in Seal @
…addresses this by sticking your personal info in a cookie rather than on the server. This relies on the security of the cookie jar to only show the image to you when you're visiting the site; it never goes over the wire.

Unfortunately, cookies are fragile, so there are lots of benign reasons that your seal might not show up, so the absence of the security features isn't a surefire alarm.

OpenID: largely orthogonal. Although because most OpenID providers use usernames & passwords, they're particularly vulnerable to phishing attacks and password stealing as we've seen



Browser extensions

PwdHash [Ross et al., Security ’05]

Passpet [Yee & Sitaker, SOUPS ’06]

Web Wallet [Wu et al., SOUPS ’06]

PwdHash involves a memorable master password that is hashed to make unique strong passwords for every site you visit.

Passpet does something similar, and adds automatic form filling to the equation.

The problem with these is that you no longer know (or remember) the actual password being used on the site, so you can get into trouble if you roam to other browsers.

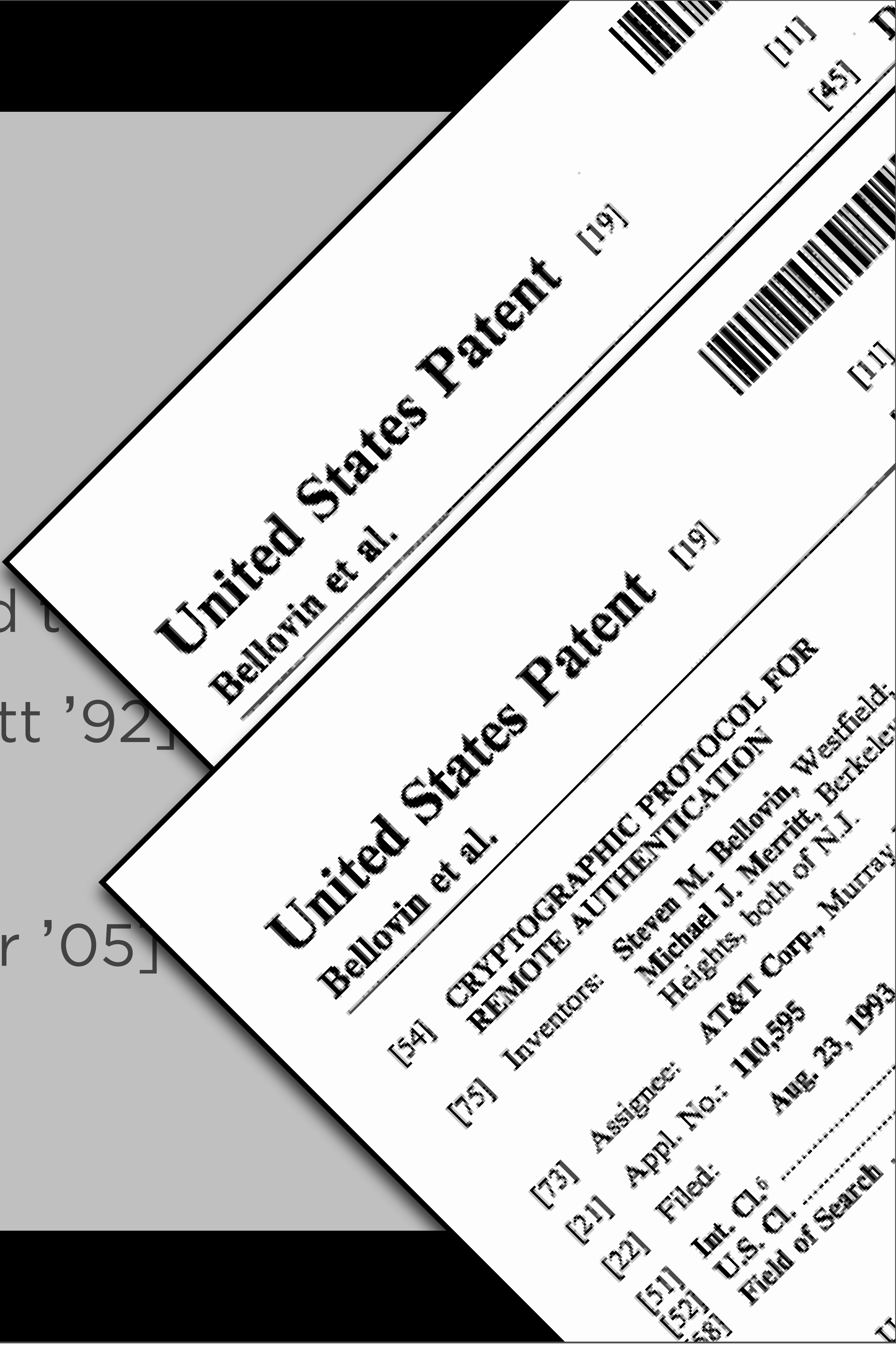
The Web Wallet bears some superficial resemblance to the Password Booth; it’s sort of a passwords auto-fill on steroids. It also has portability problems, plus it’s non-mandatory: you have to explicitly invoke it.

Long-term solutions

Microsoft CardSpace

Zero-knowledge password t

- EKE [Bellovin & Merritt ’92]
- SRP [Wu ’98]
- DSS [Dhamija & Tygar ’05]



United States Patent [19]
Bellovin et al.
[11]
[45]

United States Patent [19]
Bellovin et al.
[11]

[54] CRYPTOGRAPHIC PROTOCOL FOR REMOTE AUTHENTICATION

[75] Inventors: Steven M. Bellovin, Westfield; Michael J. Merritt, Berkele Heights, both of N.J.

[73] Assignee: AT&T Corp., Murray

[21] Appl. No.: 110,595

[22] Filed: Aug. 23, 1993

[51] Int. Cl.6

[52] U.S. Cl.

[58] Field of Search

@ **CardSpace** solves many of these problems; in a way, it’s the Super Mega Ultra password booth. But it is a much larger system: it requires changes all over the place, and is clearly a “future” technology.

@ Then there are the zero-knowledge password verification schemes. They allow you to authenticate by convincing the server that you know the password, without revealing anything about the password.

Verifier schemes also provide two-way authentication: you, the user, are assured that the server you are talking to is authentic. @ **Encrypted Key Exchange** is the grandfather of them all; @ **Secure Remote Password** (SRP) builds on this and @ **Dynamic Security Skins** applies SRP in a visual way and has other anti-spoofing features.

They will all require support from both client and server, and that support is not here yet.

Moreover, there are some non-technological reasons @ why we might expect to wait a few years before we see these verifier-based schemes.

For all of these techniques, users will need to start typing passwords in a safe place and not straight into untrusted Web forms. Something simple like the Password Booth could be used now to train users for that future.

The password field is unsafe

Well, all Web forms, really

…but passwords are an excellent place to start

The password booth

A safe place to enter passwords

A good, timely place for security indicators

We can do this in browsers—now

A step toward more sophisticated web authentication user experience

To recap.

(eof)

What if a website wants to *embrace* the booth?

Options:

1. Do nothing

2.
   ```
   if (window.booth) {
     loginform.style.display = ‘none’;
     loginbutton.style.display = ‘inherit’;
   }
   ```
3. Old-school: `<NOBOOTH>...</NOBOOTH>`
4. Other ideas?

Spoofing

The venerable "trusted path problem"

…which has not really been solved

We've done this to ourselves

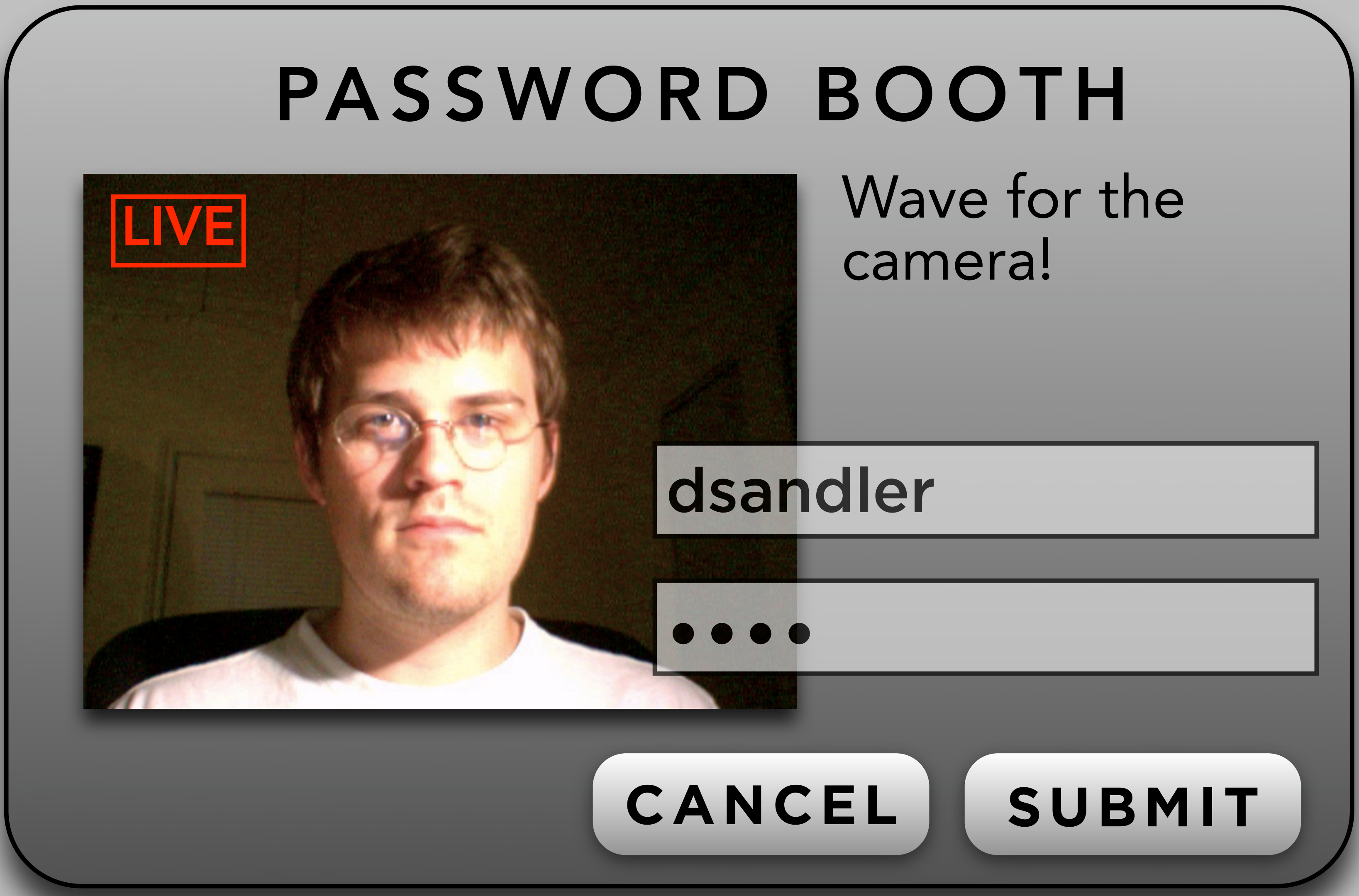
We keep making HTML & JS more powerful, which means there are fewer tricks we can pull that are un-spoofable

> **Vista:** screenwide transparent overlay
>
> Here's another idea...

(other notes)

This really ought to be solved at the system level so that you know unambiguously where each window is coming from.

You could also use a personal device that you trust, like a smartphone, to do the password entry.



Why not use the **webcam on your computer** to satisfy you that the dialog is coming from the local machine?