Web Security & Phishing

1

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 rarely the main concern
  - 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)

2

Web-browser security

- User interface
- Buggy code
- Authentication
- Active content
  - JavaScript, Java, Flash, ActiveX, ...
- Attackers goals
  - Steal personal information
  - Gain bots

3

JavaScript

- Language executed by browser
  - Before HTML is loaded
  - Before page is viewed
  - While it is being viewed
  - When leaving the page
- Used by attackers to exploit other vulnerabilities
  - Execute some code on user's machine
  - Cross-scripting:
    - Inserts malicious JavaScript into Web page or HTML email
    - E.g.: to steal user's cookies

4

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

5

Risks of poorly written scripts

- For example, echo user's input

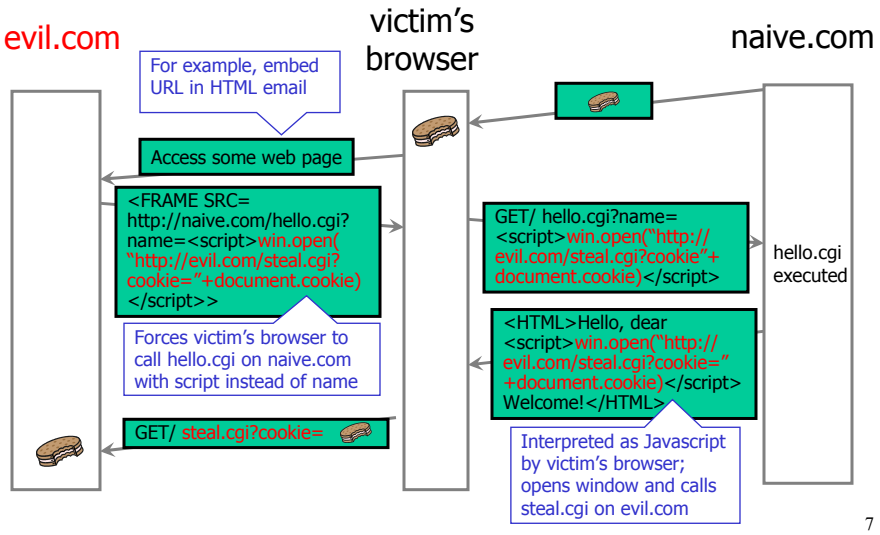
http://naive.com/search.php?term="Britney Spears"
search.php responds with
<html> <title>Search results</title>
<body>You have searched for <?php echo $_GET[term] ?>… </body>

Or

GET/ hello.cgi?name=Bob
hello.cgi responds with
<html>Welcome, dear Bob</html>

6

Stealing cookies by cross scripting



7

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
  - Use:

    alert('double quote: ' + String.fromCharCode(34))

8

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
- Infection via visit to infected page
  Adds "samy" as a friend and hero
- 5 hours later "samy"
  has 1,005,831 friends
  - Peak: 1,000 new friends per second

9

ActiveX

- ActiveX controls are downloaded and installed
  - Compiled binaries for client's OS
- ActiveX controls reside on client's machine
  - Activated by HTML object tag on page
  - > 1000 controls on new out-of-the box machine!
- Security model
  - Digital signatures to verify source of binary
  - Browser policy can reject controls from network zones
  - Controls can be marked by author as "safe for initialization" or "safe for scripting"

Once accepted, installed, started,
no control over execution!

10

Installing Controls

If you install and run, no further control over the code

In principle, browser/OS could apply sandboxing, etc. for containing risks in native code

11

ActiveX risks

- From MSDN:
  - "An ActiveX control can be an extremely insecure way to provide a feature. Because it is a Component Object Model (COM) object, it can do anything the user can do from that computer. It can read from and write to the registry, and it has access to the local file system. From the moment a user downloads an ActiveX control, the control may be vulnerable to attack because any Web application on the Internet can repurpose it, that is, use the control for its own ends whether sincere or malicious."
- How can a control be "repurposed?"
  - Once installed, control can be accessed by any page that knows its class identifier (CLSID)

12

IE browser "Helper Objects"

- COM components loaded upon IE start up
- Same memory context as browser
- Perform any action on IE windows and modules
  - Detect browser events
    - GoBack, GoForward, and DocumentComplete
  - Access browser menu, toolbar and make changes
  - Create windows to display information (or ads!!)
  - Install hooks to monitor messages and actions
- There is no protection from extensions
  - Spyware writers' favorite!
  - Try running HijackThis on your computer

13

Attacks on browser privacy

- "Same-origin" principle
- Not fully enforced in today's browsers
  - Firefox checks third-party cookie policy only when cookie is read, not when cookie is set
    - Any site can set a third-party cookie
- Cache tracking and timing attacks
  - Measure time it takes to load a page
    - If fast, user must have visited it recently (still in the cache)
  - Measure time it takes to do a DNS lookup

14

Web-server security

- Servers are tempting targets
- Defacements
- Steal data
- Distribute malware

- Defense
  - Check all inputs
  - Trust nothing
  - Scrub your site

15

Preventing cross-site scripting

- Difficult to prevent injection of scripts into HTML
- Preprocess any input
  - E.g. use PHP, htmlspecialchars(string) to replace special characters with their HTML codes
    - ` becomes '
    - " becomes "
    - & becomes &

16

Inadequate input validation

- http://victim.com/copy.php?name=username
- copy.php includes

  Supplied by the user!

  system("cp temp.dat $name.dat")
- User calls

  http://victim.com/copy.php?name="a; rm *; "
- copy.php executes

  system("cp temp.dat a; rm *;.dat");

17

URL redirection

- http://victim.com/cgi-bin/loadpage.cgi?page=url
  - Redirects to url
  - E.g.: used for tracking user clicks; referrals
- Phishing website puts

  http://victim.com/

  cgi-bin/loadpage.cgi?page=phish.com
- Looks Ok (link is pointing to victim.com), but user redirected to phishing site!

18

Dangerous Web sites

- Recent "Web patrol" study at Microsoft:
  752 unique URLs for exploiting
  unpatched Windows XP machines
- "But I never visit risky websites"
  - 11 exploit pages among top 10,000 most visited
  - Create page with popular content
  - Get into search engines
  - Page redirects to the exploit site
  - E.g.: one malicious sites provided exploits to 75 "innocuous" sites focusing on:
    (1) celebrities, (2) song lyrics, (3) wallpapers,
    (4) video game cheats, and (5) wrestling

19

User data in SQL queries

- set UserFound=execute(
  SELECT * FROM UserTable WHERE
  username='" & form("user") & "' AND
  password='" & form("pwd") & "' ");
  - User supplies username and password,
    SQL query checks if user/password is in database
- if not UserFound.EOF
  Authentication correct
  else Fail

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

20

SQL injection

Always true!

- Username ’ OR 1=1 --
- Web server executes

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

Everything after -- is ignored!

- Returns entire database!
- UserFound.EOF always false;
  Authentication always “correct”

21

It gets better

- Username

  ’ exec cmdshell ’net user badguy badpwd’ / ADD --
- Web server executes

```
set UserFound=execute(
    SELECT * FROM UserTable WHERE
    username=’ ’ exec … -- … );
```

- Creates account for badguy on DB server
- Fix: escape user-supplied arguments
  - Convert ’ into \’

22

Uninitialized inputs

```
/* php-files/lostpassword.php */
for ($i=0; $i<=7; $i++)
    $new_pass .= chr(rand(97,122))
…
$result = dbquery("UPDATE ".$db_prefix."users
   SET user_password=md5('$new_pass')
   WHERE user_id='".$data['user_id']." ' ");
```

Creates a password with 7 random characters, assuming $new_pass is set to NULL

In normal execution, this becomes

```
UPDATE users SET user_password=md5('???????')
WHERE user_id='userid'
```

SQL query setting password in the DB

23

Exploit

User appends this to 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'

24

SQL injection in the real world

- "A programming error in the University of Southern California's online system for accepting applications from prospective students left the personal information of as many as 280,000 users publicly accessible… The vulnerability in USC's online Web application system is a relatively common and well-known software bug, known as database injection or SQL injection"
  - SecurityFocus, July 6, 2005

25

Server-side scripts

- Popular languanges: CGI, ASP, PHP, server-side includes, ...
- Each script is a separate network service!
- All scripts have to be secure
- Context to run scripts in? The Web server's? How to protect its sensitive files against bad clients?
- What about server plug-ins, e.g., PHP
- Partial defense: suexec

26

Phishing

Spoofed emails

27

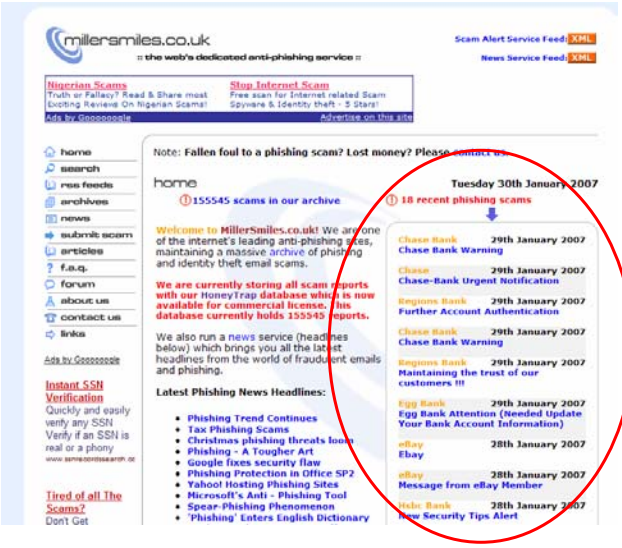
A Few Headlines

- "11.9 million Americans clicked on a phishing e-mail in 2005"
- "Gartner estimates that the total financial losses attributable to phishing will total $2.8 bln in 2006"
- "Phishing and key-logging Trojans cost UK banks £12m"
- "Swedish bank hit by 'biggest ever' online heist"

  "Swedish Bank loses $1 Million through Russian hacker"

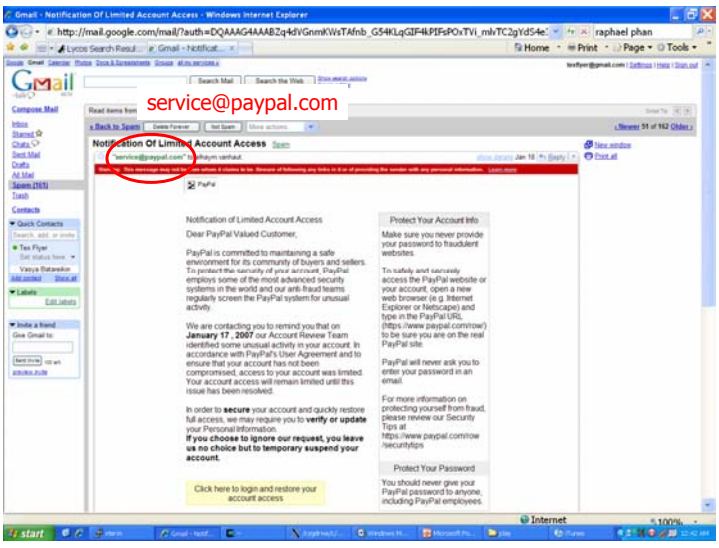
28

MillerSmiles.co.uk



29

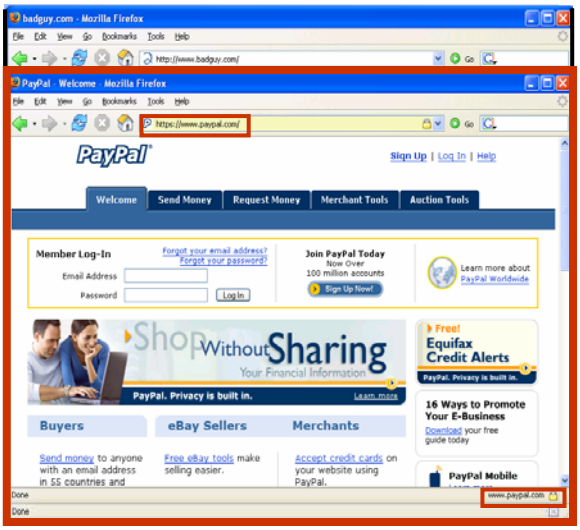
A Snapshot of My Mailbox



30

Typical Phishing Page

Or Even Like This



A Closer Look

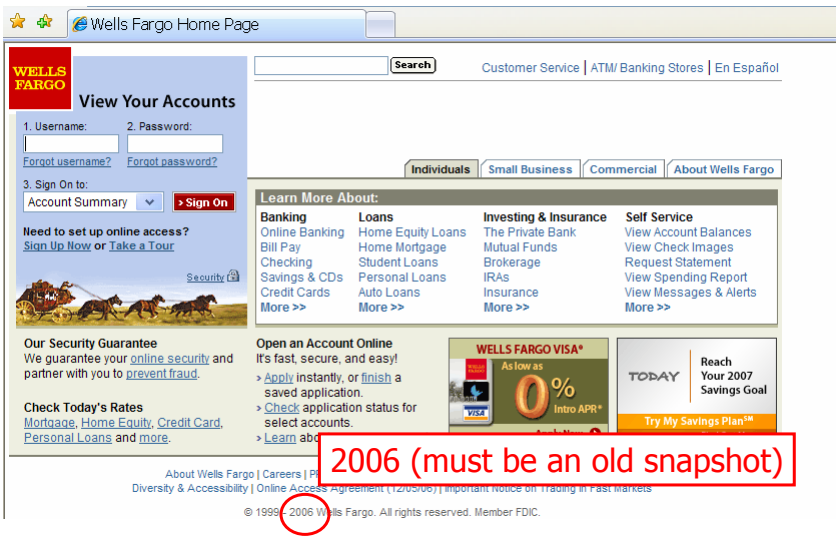
From: "Wells Fargo" <aw-updateWells.Fargo.com@abm-tech.com>

What you'll see on the page

Where the link actually goes

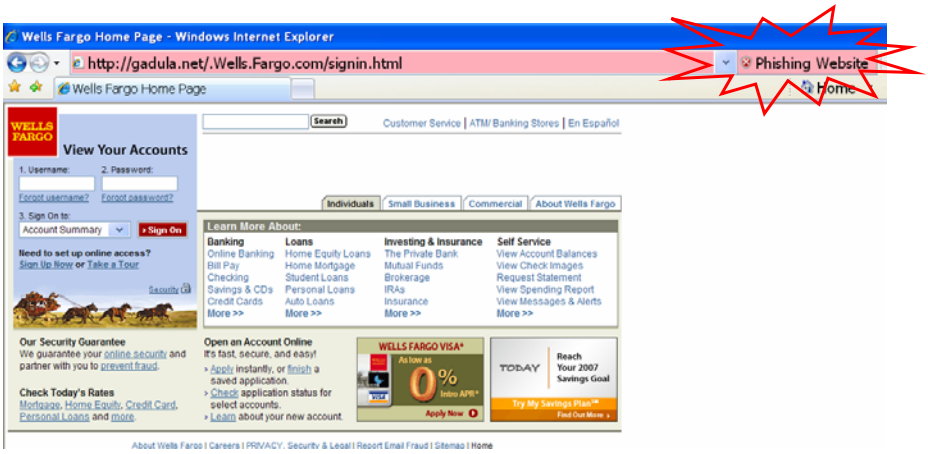
<a target="_blank"
href="**http://www.members.axion.net/~rod/.Wells.Fargo.com**" >
https://online.wellsfargo.com/signon?LOB=CONS

And You End Up Here



2006 (must be an old snapshot)

Thank Goodness for IE 7.0 ☺



Phishing Techniques

- Use confusing URLs
  - http://gadula.net/.Wells.Fargo.com/signin.html
- Use URL with multiple redirection
  - http://www.chase.com/url.php?url="http://phish.com"
- Host phishing sites on botnet zombies
  - Move from bot to bot using dynamic DNS
- Pharming
  - Poison DNS tables so that victim's address (e.g., www.paypal.com) points to the phishing site
  - URL checking doesn't help!

Bad Idea: Echoing User Input

- User input echoed in HTTP header
- For example, language redirect:

```
<% response.redirect("/by_lang.jsp?lang=" +
                request.getParameter("lang") )   %>
```

- Browser sends

  http://.../by_lang.jsp ? lang=french

- Server responds

  `HTTP/1.1 302` redirect

  `Date: …` to here

  `Location: /by_lang.jsp ? lang=french`

37

HTTP Response Splitting

- Malicious user requests

  http://.../by_lang.jsp ? lang=
  "french \n
  Content-length: 0 \r\n\r\n
  HTTP/1.1 200 OK
  <Encoded URL of phishing page>"

- Server responds:

  HTTP/1.1 302
  Date: ...
  Location: /by_lang.jsp ? lang= french
  Content-length: 0
  HTTP/1.1 200 OK
  Content-length: 217
  Phishing page

  Looks like a separate page

38

Why?

- Attacker submitted a URL to victim.com
- Response from victim.com contains phishing page
- All cache servers along the path will store the phishing page as the cache of victim.com
- If an unsuspecting user of the same cache server requests victim.com, server will give him the cached phishing page instead

39

Trusted Input Path Problem

- Users are easily tricked into entering passwords into insecure non-password fields

```
<input  type="text"  name="spoof"
    onKeyPress="(new Image()).src=
        'keylogger.php?key=' +
        String.fromCharCode( event.keyCode );
    event.keyCode = 183;" >
```

Sends keystroke to phisher

Changes character to *

40

Social Engineering Tricks

- Create a bank page advertising an interest rate slightly higher than any real bank; ask users for their credentials to initiate money transfer
  - Some victims provided their bank account numbers to "Flintstone National Bank" of "Bedrock, Colorado"
- Exploit social network
  - Spoof an email from a Facebook or MySpace friend
    - Read Jan 29 WSJ article about MySpace hack
  - In a West Point experiment, 80% of cadets were deceived into following an embedded link regarding their grade report from a fictitious colonel

41

Experiments at Indiana University

[Jagatic et al.]

- Reconstructed social network by crawling sites like Facebook, MySpace, LinkedIn and Friendster
- Sent 921 Indiana University students spoofed email (apparently from their friend)
- Email redirected to spoofed site asking user to enter his/her secure university credentials
  - Domain name clearly distinct from indiana.edu
- 72% of students entered real credentials
  - Males more likely if email sender is female

42

Victims' Reactions (1)

[Jagatic et al.]

- Anger
  - Subjects called the experiment unethical, inappropriate, illegal, unprofessional, fraudulent, self-serving, useless
  - Called for researchers conducting the study to be fired, prosecuted, expelled, or reprimanded
- Denial
  - No posted comments with admission that writer was victim of attack
  - Many posts stated that poster did not and would never fall for such an attack, and they were speaking on behalf of friends who had been phished

43

Victims' Reactions (2)

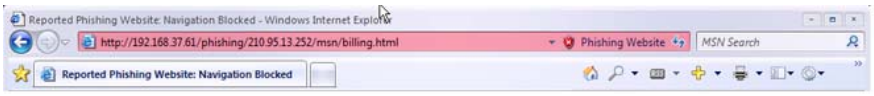
[Jagatic et al.]

- Misunderstanding
  - Many subjects were convinced that the experimenters hacked into their email accounts. They believed it was the only possible explanation for the spoofed messages.
- Underestimation of privacy risks
  - Many subjects didn't understand how the researchers obtained information about their friends, and assumed that the researchers accessed their address books
  - Others, understanding that the information was mined from social network sites, objected that their privacy had been violated by the researchers who accessed the information that they had posted online

44

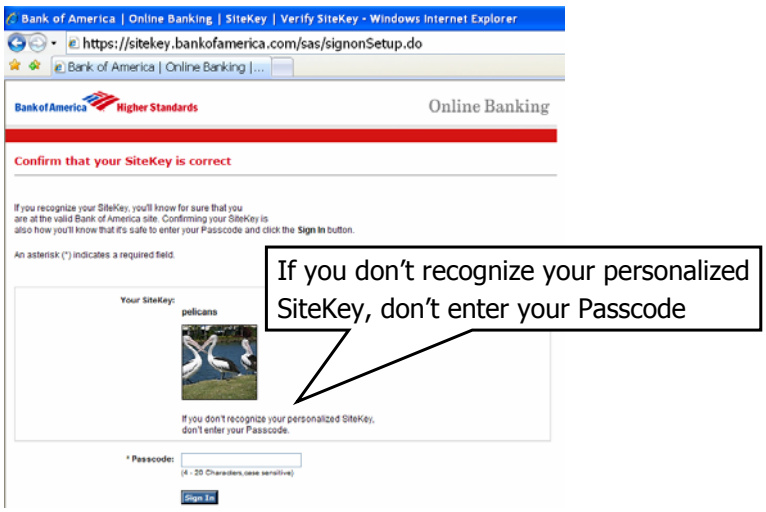
Defense #1: Internet Explorer 7.0

- "White list" of trusted sites
- Other URLs sent to Microsoft
  Responds with "Ok" or "Phishing!"



45

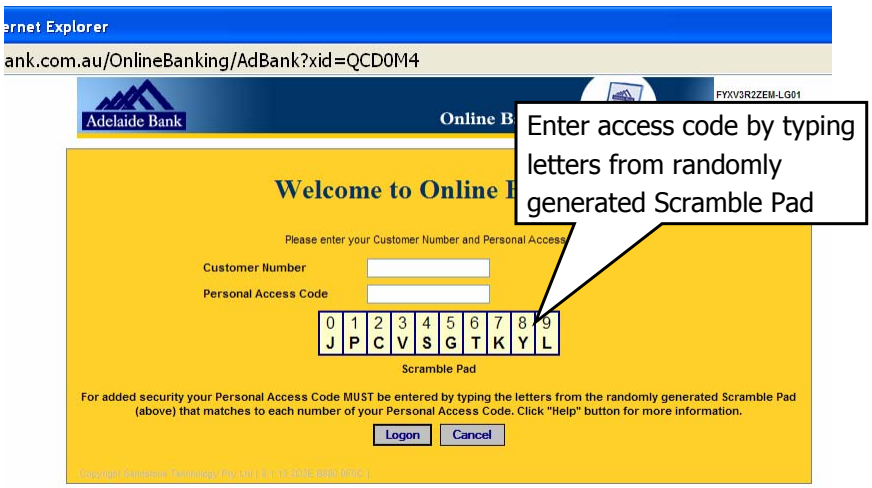
Defense #2: PassMark / SiteKey



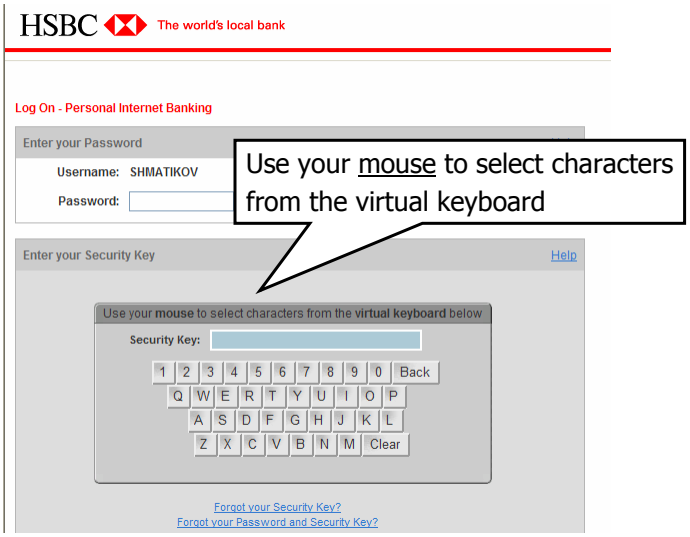
46

Defense #3: PIN Guard

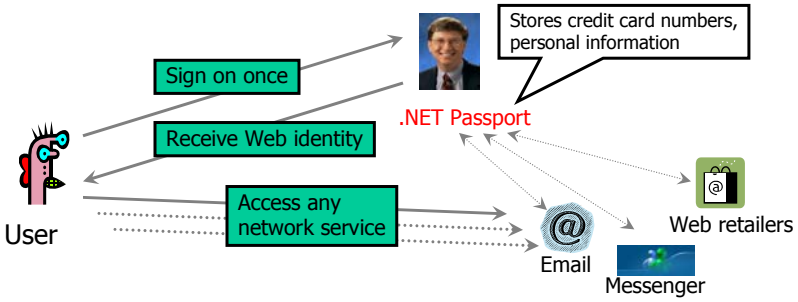
Defense #3A: Scramble Pad



Defense #4: Virtual Keyboard



Microsoft Passport



- Idea: authenticate once, use everywhere
- Trusted third party issues identity credentials
- User uses them to access services over the Web

History of Passport

- Launched in 1999
  - 2002, Microsoft claims > 200M accounts, 3.5 billion authentications each month
- Passport: Early Glitches
  - Flawed password reset procedure
  - Cross-scripting attack
- Current status
  - From Directory of Sites at http://www.passport.net: "We have discontinued our Site Directory…"
  - Monster.com dropped support in October 2004
  - eBay dropped support in January 2005
  - Seems to be fizzling out

51

Liberty Alliance

- Open-standard alternative to Passport

http://www.projectliberty.org

- Promises compliance with privacy legislation
- Long list of Liberty-enabled products

52

Defenses

- Use mutual authentication
- Non-Reusable credentials
  (not sufficient against man-in-the-middle attacks)

- Basic technical mechanism available
- Human interaction with these is a challenge!
- Security is a systems problem

53