Web application security

Ing. Stefano Zanero

What I will not explain here

- What is HTTP and how it works
  - You **will** need to know this!
- What a 3-tier web application is
  - You **will** need to know this!
- How to code in a specific web-application oriented language or framework
  - You don't need any specific one, but you **will** need to know at least one to understand

HTTP is a problem

- Stateless protocol not designed for sessions (guess what it's being used for?)
- A public protocol designed to serve anonymous users (weak authentication)
- HTTP servers have an ominous security track record

Web applications are **the** problem

- The “new model” of computing
- A wild environment (often exposed to the public Internet)
- Staggering results: estimates say that 4 web applications out of 5, roughly, have some serious vulnerabilities
  - Security often an afterthought
  - Redesign is costly and time consuming
  - Interaction between server, application server, database, and programming frameworks is a mess

Where the vulns are in webapps

- Buffer Overflow
- Eavesdropping
- Race Condition
- Man in the Middle
- Input Validation
- Session Hijacking
- Memory Residue
- Replays
- Path Manipulation
- Backdoors

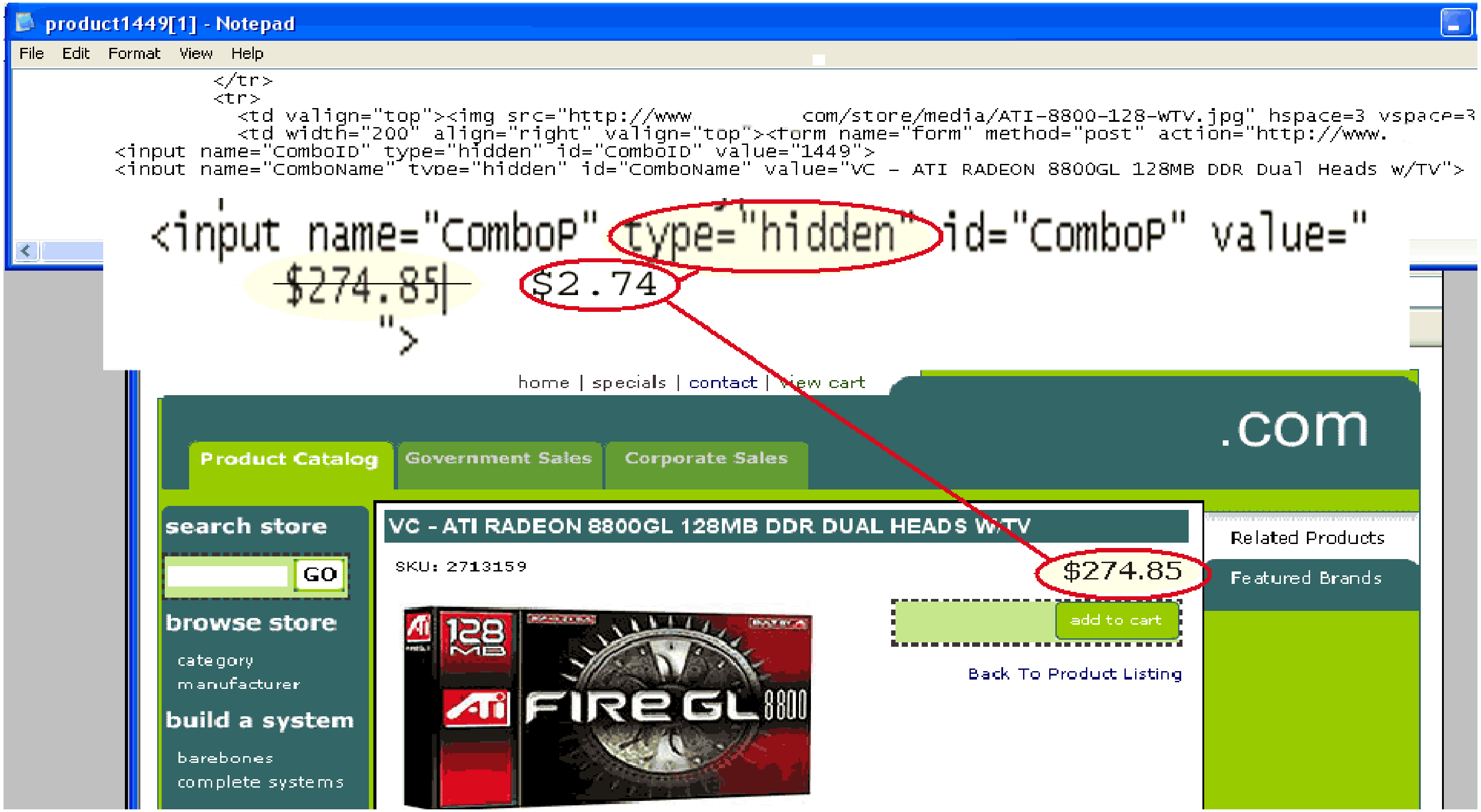
- Buffer Overflow (-)
- Eavesdropping (+)
- Man in the Middle (+)
- **Input Validation (+++++)**
- Session Hijacking (++)
- Replays

First rule: the client is not trustworthy

Clients are not trustworthy

- So we cannot trust:
  - To validate inputs or perform actions on the client side, e.g. through javascript
  - Variables, such as REFERER, that the client is sending us
  - In general, any data the client is giving us
- The challenge is that the instinct of the programmer is to think that the client is a part of their application infrastructure

True example, with hidden identity



```
            </tr>
            <tr>
              <td valign="top"><img src="http://www          com/store/media/ATI-8800-128-wTV.jpg" hspace=3 vspace=3
              <td width="200" align="right" valign="top"><form name="form" method="post" action="http://www.
      <input name="ComboID" type="hidden" id="ComboID" value="1449">
      <input name="ComboName" type="hidden" id="ComboName" value="VC - ATI RADEON 8800GL 128MB DDR Dual Heads w/TV">
```

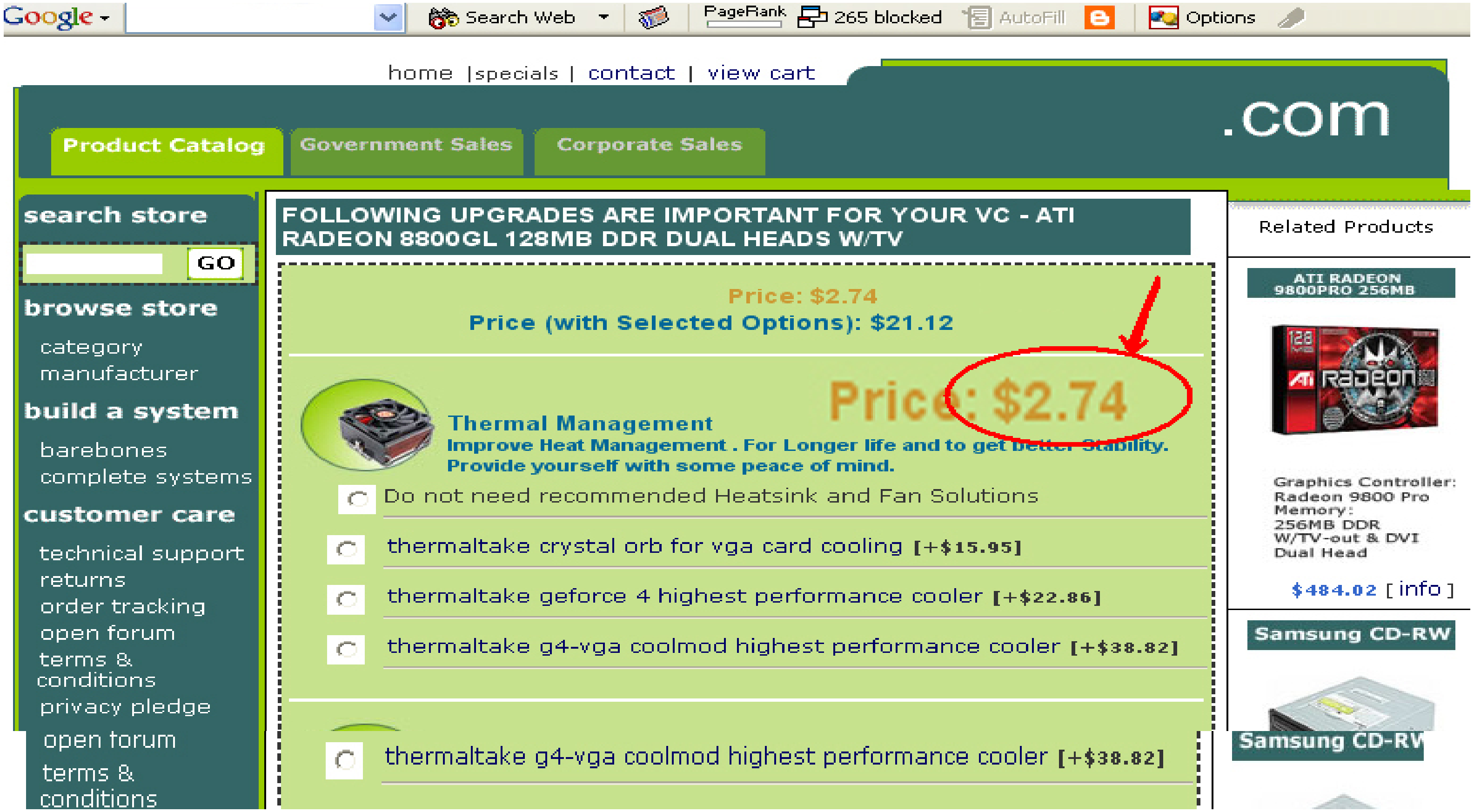
`<input name="ComboP" type="hidden" id="ComboP" value="` ~~$274.85~~ $2.74 `">`

VC - ATI RADEON 8800GL 128MB DDR DUAL HEADS W/TV

SKU: 2713159

$274.85

The price is just about right!



FOLLOWING UPGRADES ARE IMPORTANT FOR YOUR VC - ATI RADEON 8800GL 128MB DDR DUAL HEADS W/TV

Price: $2.74

Price (with Selected Options): $21.12

Price: $2.74

Thermal Management

Improve Heat Management . For Longer life and to get better Stability. Provide yourself with some peace of mind.

○ Do not need recommended Heatsink and Fan Solutions

○ thermaltake crystal orb for vga card cooling [+$15.95]

○ thermaltake geforce 4 highest performance cooler [+$22.86]

○ thermaltake g4-vga coolmod highest performance cooler [+$38.82]

○ thermaltake g4-vga coolmod highest performance cooler [+$38.82]

Graphics Controller: Radeon 9800 Pro
Memory: 256MB DDR
W/TV-out & DVI
Dual Head

$484.02 [info]

Second rule: always validate your input

Validate it. Validate it all.

- Web application inputs are **always** untrusted
- Anything you get must always be checked
- There's no filter that is too paranoid, while there are entire graveyards of filters which were “not paranoid **enough**”
- It ain't easy.

A validation sequence

- Whitelisting: let through only recognized things
- Blacklisting: from this, filter out anything that you can recognize as bad
- Escaping: from what is left, escape special characters appropriately
- Parsing: only after all this, elaborate input

Whitelisting is good

- There's always two approaches to filtering in security
  - Blacklisting: take away what is wrong
  - Whitelisting: let through only what is right
- The general principle is that, in security, **blacklisting is bad**
  - That being said, at times you simply cannot whitelist because you don't know enough
  - That being said, if you have already done whitelisting, then doing some additional blacklisting may help

Escaping

- Substituting special characters with a version that will not trigger bad behaviors
- E.G. a free text field in a web application should not allow a user to input HTML tags, so we can:
  - Replace > with &gt;
  - Replace < with &lt;
  - Replace " with &quot;
  - Replace & with &amp;

Escaping: what and how?

- What? many things; examples:

../	(Directory Transversal)
(*, ?, +)	(globbing)
";"	(command append)
">" "<" "|"	(data piping)
" and ‘	(string terminators)

- How? Depends on the system, the input and the output

And REMEMBER!

- The client is NOT TRUSTWORTHY
- Validation MUST BE DONE on the server side
- If you are using scripting to validate data, you are in a big sea of trouble

A consequence of lack of validation: Cross Site Scripting

Cross-Site Scripting (XSS)

- Insertion of unauthorized scripting code on a webpage
- “So what? Scripting code is harmless in its sandbox, right?”. Yeah, right, but:
  - Cookie theft
  - Session hijack
  - Manipulation of a session and execution of fraudulent transaction
  - Snooping on private information

First example: stored XSS

- A blog comment platform which doesn't perform any filtering
- Any attacker can insert scripting code in his comment, it will be **stored** in the backend and displayed to any subsequent user

Second example: reflected XSS

- A feedback page which doesn't perform any filtering
- The feedback is then showed just once to the original poster, for acknowledgement
- Any attacker can insert scripting code in his feedback but it will be displayed back to him... so what's the point?
- Craft a url submitting malicious feedback with evil javascript, and then social engineer a user to click on it

The nightmare of HTML filtering

- Ever wondered why most web application restrict HTML formatting in input fields?
- (Or, at very least, restrict you to a subset of delimiters, often replaced by nonstandard ones)
- That's because filtering out “bad stuff” in this case is a nightmare

Enemies of the state

- Tags:
  - <APPLET>
  - <BASE>
  - <BODY>
  - <EMBED>
  - <FRAME>
  - <FRAMESET>
  - <HTML>
  - <IFRAME>
  - <IMG>
  - <LAYER>
  - <META>
  - <OBJECT>
  - <P>
  - <SCRIPT>
  - <STYLE>

- Attributes:
  - STYLE
  - SRC
  - HREF
  - TYPE

- And if you have a problem in understanding why, follow me through the next few slides...

“Fatta la legge, trovato l'inganno”

- An old Italian saying...
- Suppose we are tossing out just the <SCRIPT> tag:

  ```
  <SCRIPT>alert('JavaScript Executed');</SCRIPT>
  ```
- What about these equivalent tags?

  ```
  <IMG SRC="javascript:alert('JavaScript Executed');">
  <ANYTHING SRC="javascript:alert('JavaScript
  Executed');">
  ```
- Solution: “strip out” the “javascript:” keyword from the SRC attribute!
- Oh, really? Too bad that...

A whitespace problem...

- My filter strips out “javascript” from SRC... but what if I write:

```
<IMG SRC="javasc

ript:alert('JavaScript Executed');">
```

- It works ! :-(
- Solution (updated): filter out CR-LF, CR, tab, spaces, etc. inside tags, then apply previous filter

HTML entities

- Now what happens if I add an HTML entity \ 09-12 ?

  ```
  <IMG SRC="javasc&#09;ript:alert('JavaScript
  Executed');">
  ```

- ... it works AGAIN.
- Solution: filter null entities, then apply previous filter. Easier said then done though:
  - I can do it with hexes

    ```
    <IMG SRC="javasc&#X0A;ript:alert('JavaScript
    Executed');">
    ```

  - I can add a bunch of zeroes

    ```
    <IMG SRC=javasc&#000010;ript:alert('JavaScript
    Executed');>
    ```

… browser craziness ...

- OK. Now we filter out whitespaces and blablas, we filter entities, we filter out the javascript keyword... what if I write: **`<IMG SRC="&{alert('JavaScript Executed')};">`**
- ... some browsers execute it, for no reasons! :-(
- Solution: filter out &{

Recursion problem...

- OK. Now we filter out whitespaces and blablas, we filter entities, we filter out the javascript keyword and the funny &{ thing.
- What happens if I write:

```
<IMG SRC="java&{script{alert('JavaScript
Executed')};">
```

- The filter takes out **&{** and the remaining string works
- Solution: instead of stripping, mangling it (e.g. replacing with something else) or throwing away the whole thing

A difficult case

- Obviously, you remembered to make the filters case-insensitive, right?

Another instance: style sheet

- Let's take the “STYLE” tag...

```
<style TYPE="text/javascript">JS EXPRESSION</style>
```

- We have a “text/javascript” to strip... same story as before. But is it enough?
- No, it isn't:

```
<STYLE type=text/css>
@import url(http://server/very_bad.css);
@import url(javascript:alert('JavaScript
Executed'));
</STYLE>
```

- Another round of filtering vs @import !

STYLE is also an attribute...

- And unsurprisingly, javascript works from within
  ```
  <P STYLE="left:expression(eval('alert(\'JavaScript
  Executed\');window.close()'))" >
  ```
- Here deciding how to filter is difficult, best choice is to drop STYLE altogether
- Do you get what I meant by **nightmare**?

Another lack of validation consequence: SQL injection

SQL Injection

- We define “SQL Injection” a bug which allows an attacker to infiltrate SQL queries to the backend database
- It can happen if unfiltered user input is naively used to compose a SQL query
- Depending on the specific database and on the privileges with which the application is authenticating itself, the attacker can have a lot of fun

SQL Injection – Example 1

```
public void OnLogon(object src, EventArgs e){
  SqlConnection con = new SqlConnection(
    "server=(local);database=myDB;uid=sa;pwd;" );

  string query = String.Format(
    "SELECT * FROM Users WHERE " +
    "username='{0}' AND password='{1}'",
    txtUser.Text, txtPassword.Text );
  SqlCommand cmd = new SqlCommand(query, con);
  conn.Open();
  SqlDataReader reader = cmd.ExecuteReader();
  try{
    if(reader.HasRows())
      IssueAuthenticationTicket();
    else
      TryAgain();
  }
  finally{
    con.Close()
  }
}
```

SQL Injection – Explanation 1

What the programmer had in mind:

```
username: abc
password: test12

The resulting query is:
select * from users where username='abc' and password = 'test12'
```

What he didn't think about:

```
username: abc'; --
password:

The resulting query is:
select * from users where uname='abc’; --' and password=''
```

What if...

- What if I didn't know a specific username?
- What if I used COUNT(*) as opposed to * and then “hasrows”?
- What if I checked “exactly 1”?

- How could I avoid the issue?

Preventing the injection

- Use of PreparedStatements
- Appropriate validation
- In any case, last resort stripping or escaping of special sequences
- Don't use DB field names as names of form fields (can you see why?)
- Limitations on query privileges (e.g. connecting with different users for different forms and privileges)

Conclusions on validation

- **Filter** inputs. Filter **all** inputs. **Always** filter **all** inputs
- Whitelisting is good, blacklisting is bad
- HTML is bad. Worse than you would think.

Error Management

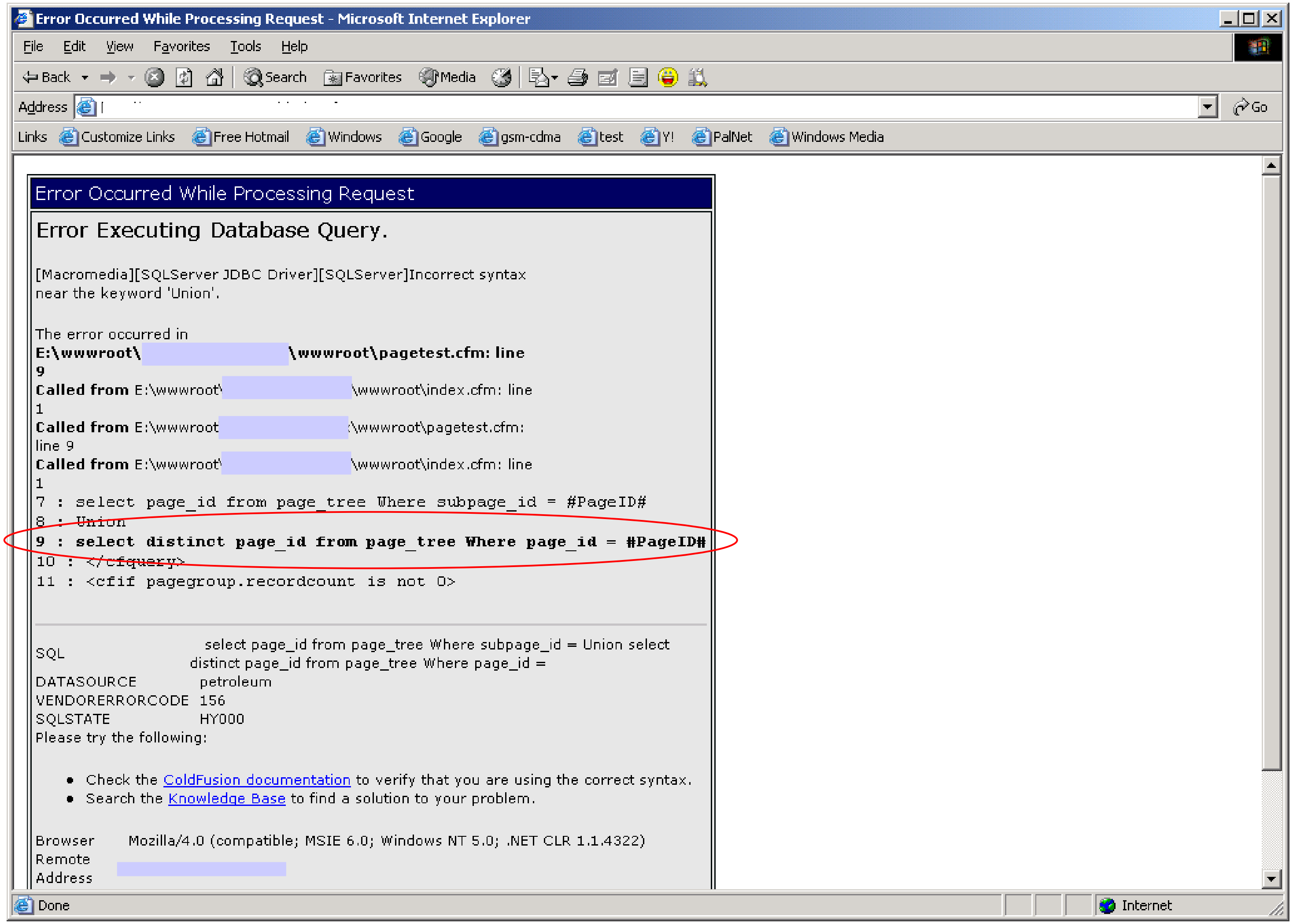
Errors in errors

- A nice error message is good HCI practice
- However, an error which echoes user-inserted strings and data is dangerous
- Errors can be dangerous also by creating side channels (e.g. “wrong password” as opposed to “wrong username”)

Freudian slip

- By default, application servers and many applications print out informative debug traces
- They are called “debug” traces as opposed to “production” traces, because they should not be used in production environments ;-)
- What we don't want to reveal:
  - Server and application versions
  - Database names, structure and credentials
  - Pathnames

Enumeration example


Error Occurred While Processing Request - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address
Go
Links Customize Links Free Hotmail Windows Google gsm-cdma test Y! PalNet Windows Media
Done
Internet

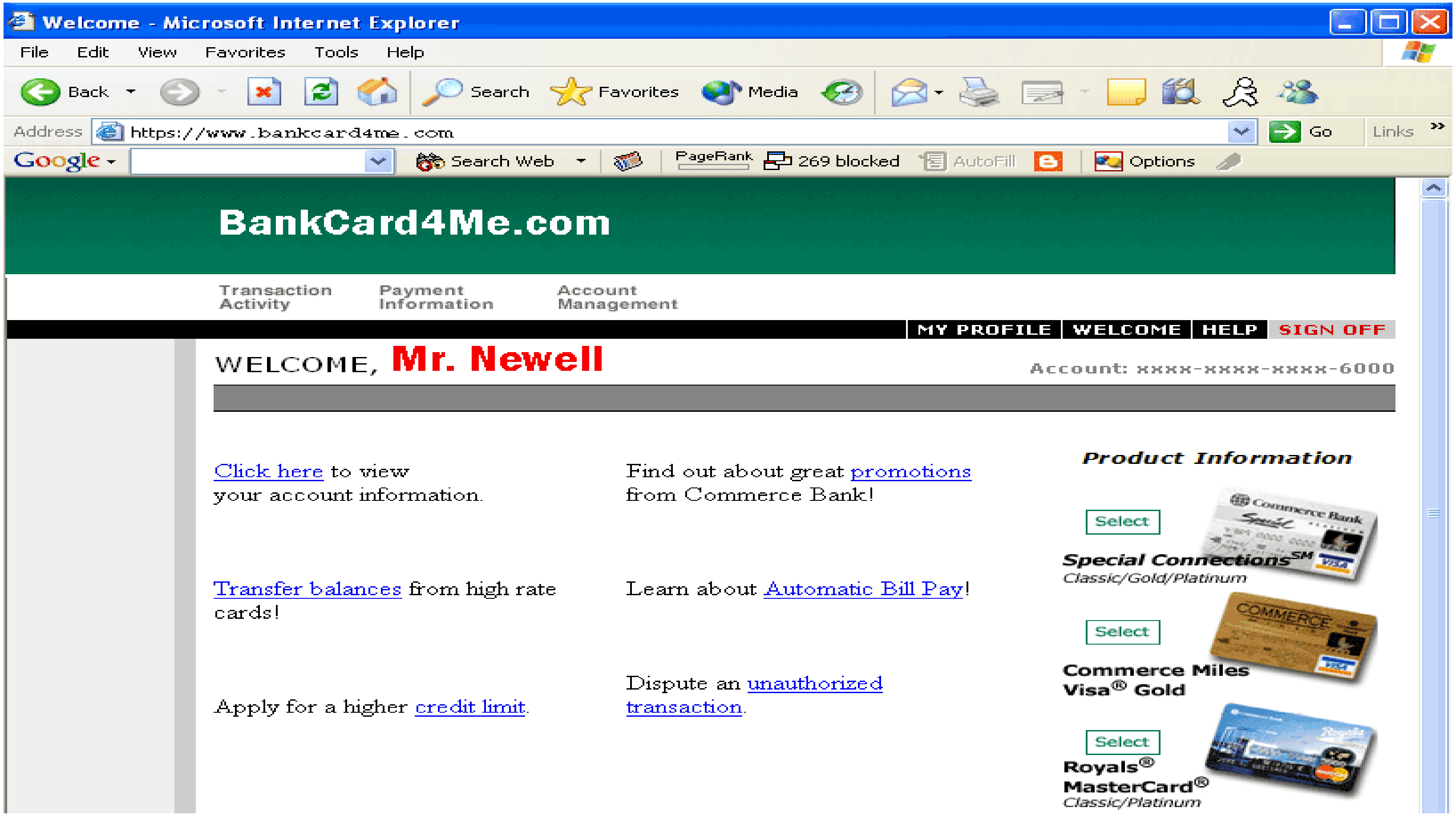
```
Error Occurred While Processing Request

Error Executing Database Query.

[Macromedia][SQLServer JDBC Driver][SQLServer]Incorrect syntax
near the keyword 'Union'.

The error occurred in
E:\wwwroot\[redacted]\wwwroot\pagetest.cfm: line
9
Called from E:\wwwroot\[redacted]\wwwroot\index.cfm: line
1
Called from E:\wwwroot[redacted]\wwwroot\pagetest.cfm:
line 9
Called from E:\wwwroot\[redacted]\wwwroot\index.cfm: line
1
7 : select page_id from page_tree Where subpage_id = #PageID#
8 : Union
9 : select distinct page_id from page_tree Where page_id = #PageID#
10 : </cfquery>
11 : <cfif pagegroup.recordcount is not 0>

SQL              select page_id from page_tree Where subpage_id = Union select
                 distinct page_id from page_tree Where page_id =
DATASOURCE       petroleum
VENDORERRORCODE  156
SQLSTATE         HY000
Please try the following:

  • Check the ColdFusion documentation to verify that you are using the correct syntax.
  • Search the Knowledge Base to find a solution to your problem.

Browser     Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0; .NET CLR 1.1.4322)
Remote
Address     [redacted]
```

Insecure Storage

Cookie Poisoning

- HTTP is stateless (grrrr!)
- HTTP is almost unidirectional
  - Client passes data to the server, but the server cannot "store" something on the client, except...
- Except for "cookies": user side information storage
  - Original idea: site customization
  - Abuse: privacy violations
  - Dangerous ideas: user authentication (expire time can be extended; simple IDs can be bruteforced or otherwise reversed)

Cookie Poisoning (naive)



Welcome - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address https://www.bankcard4me.com Go Links
Google Search Web PageRank 269 blocked AutoFill Options

BankCard4Me.com

Transaction Activity | Payment Information | Account Management

MY PROFILE | WELCOME | HELP | SIGN OFF

WELCOME, **Mr. Newell**

Account: xxxx-xxxx-xxxx-6000

Click here to view your account information.

Find out about great promotions from Commerce Bank!

Transfer balances from high rate cards!

Learn about Automatic Bill Pay!

Apply for a higher credit limit.

Dispute an unauthorized transaction.

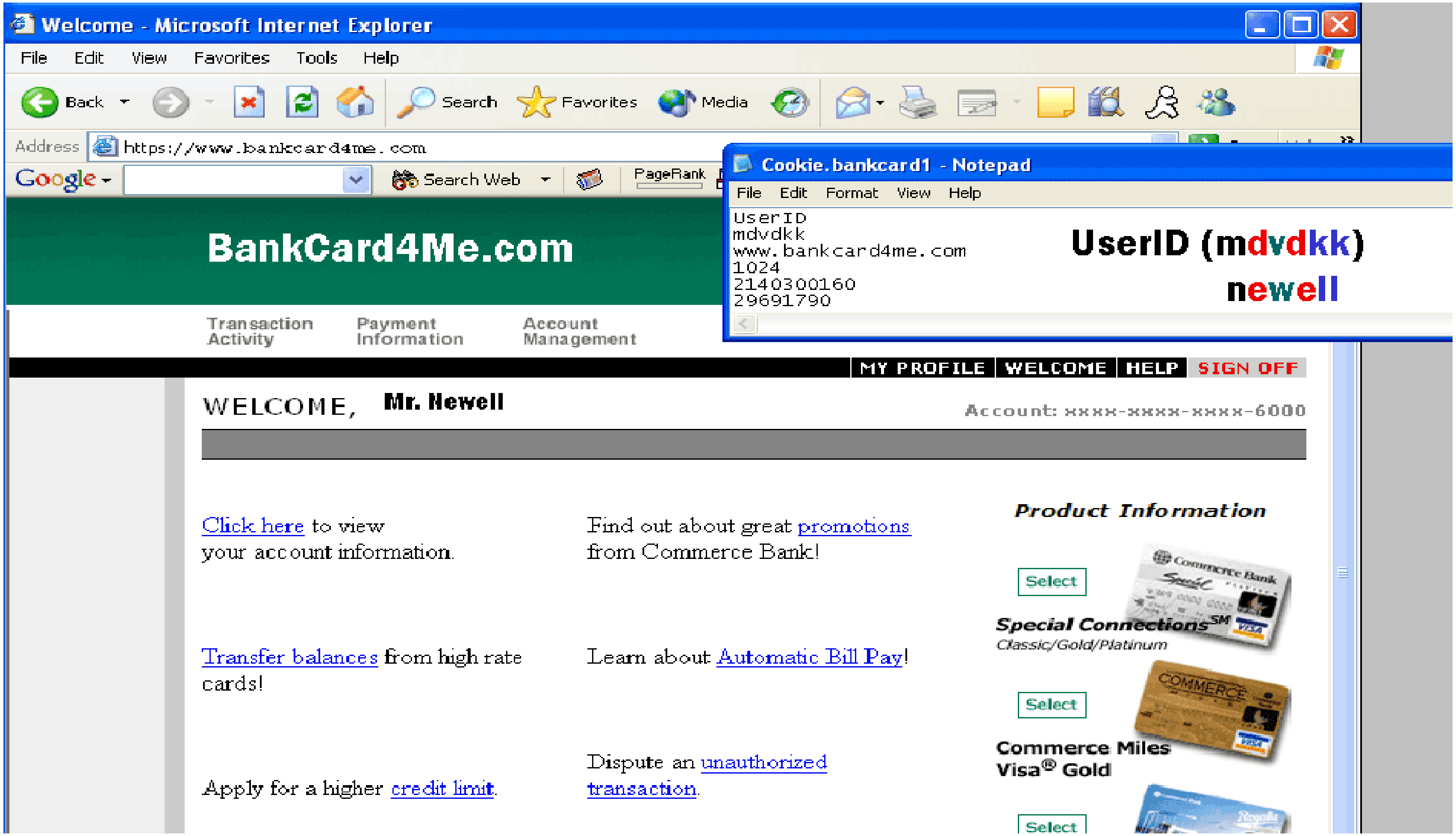
Product Information

Select
Special Connections℠
Classic/Gold/Platinum

Select
Commerce Miles Visa® Gold

Select
Royals® MasterCard®
Classic/Platinum

Cookie Poisoning



```
UserID
mdvdkk
www.bankcard4me.com
1024
2140300160
29691790
```

UserID (mdvdkk)

newell

WELCOME, Mr. Newell

Account: xxxx-xxxx-xxxx-6000

Click here to view your account information.

Find out about great promotions from Commerce Bank!

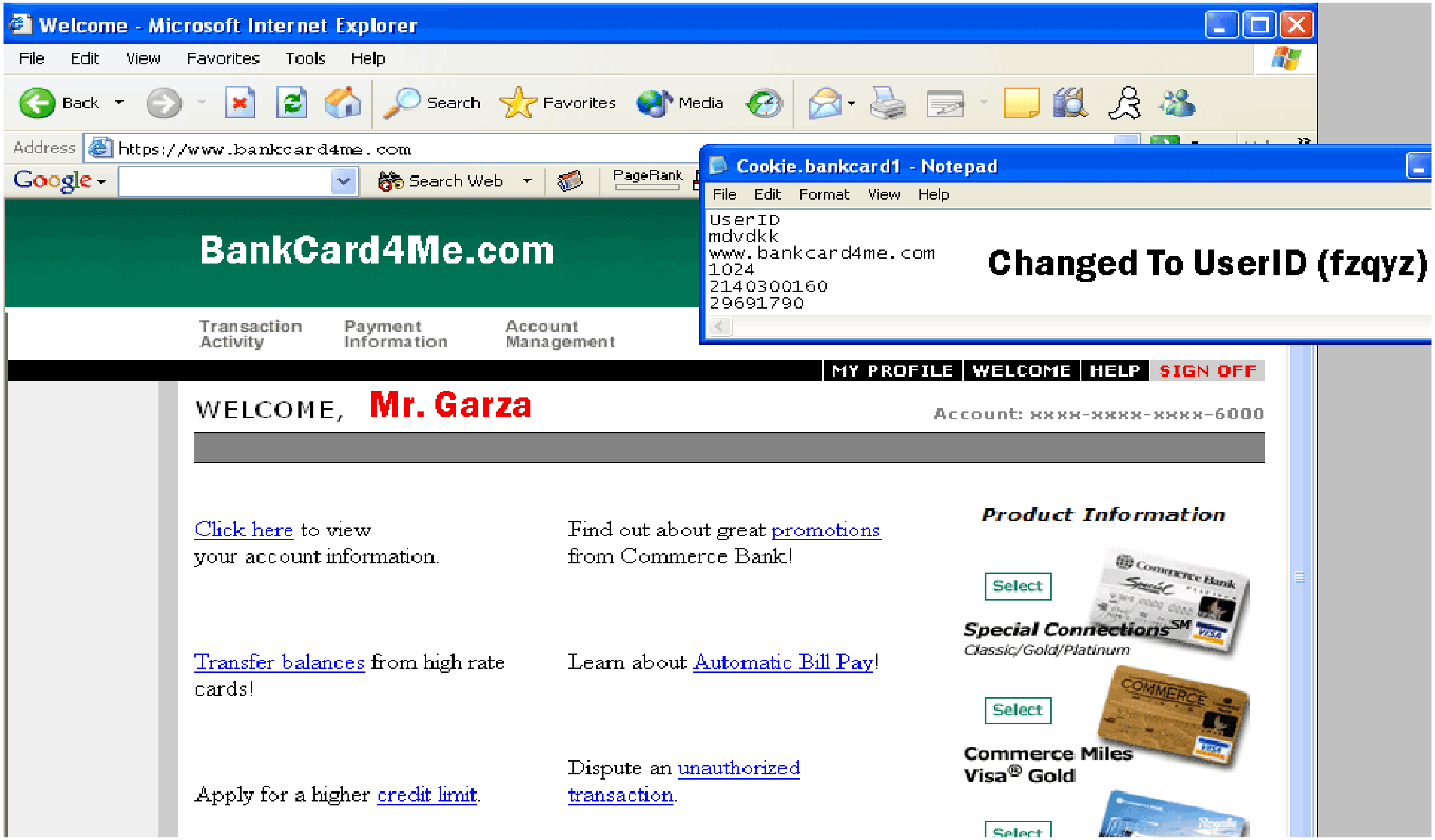
Transfer balances from high rate cards!

Learn about Automatic Bill Pay!

Apply for a higher credit limit.

Dispute an unauthorized transaction.

Cookie Poisoning



```
userID
mdvdkk
www.bankcard4me.com
1024
2140300160
29691790
```

Changed To UserID (fzqyz)

BankCard4Me.com

WELCOME, **Mr. Garza**

Account: xxxx-xxxx-xxxx-6000

Click here to view your account information.

Find out about great promotions from Commerce Bank!

Transfer balances from high rate cards!

Learn about Automatic Bill Pay!

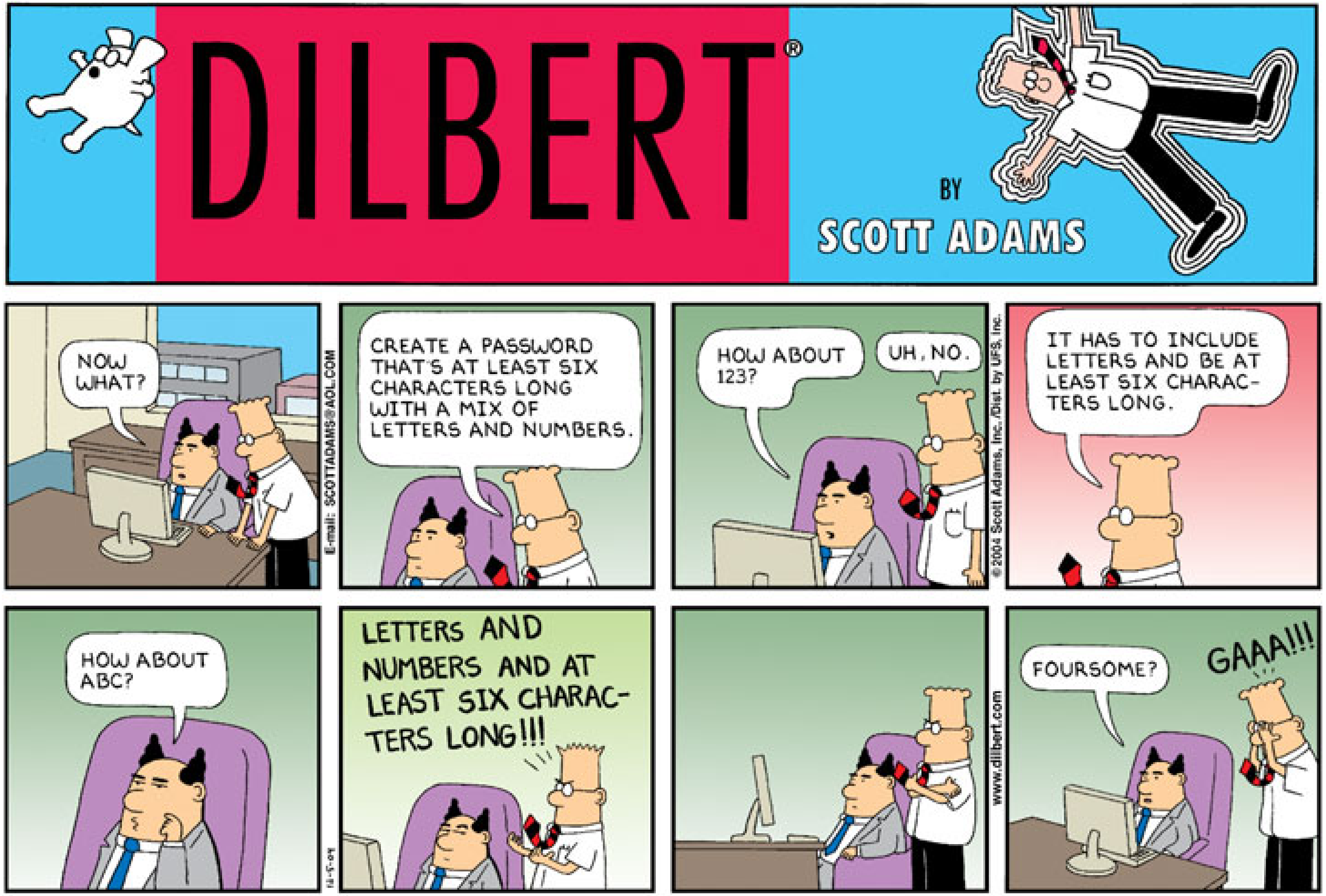
Apply for a higher credit limit.

Dispute an unauthorized transaction.

Authentication and access control

Passwords (again!)

- import(everything we said in lesson 2)
- In a web application authentication scheme
  - Passwords must not be stored in plaintext
  - Encryption must be used at protocol level (see lesson on SSL)
  - Passwords should really expire
  - Password restore schemes deserve extra attention


DILBERT®
BY SCOTT ADAMS
NOW WHAT?
E-mail: SCOTTADAMS@AOL.COM
CREATE A PASSWORD THAT'S AT LEAST SIX CHARACTERS LONG WITH A MIX OF LETTERS AND NUMBERS.
HOW ABOUT 123?
UH, NO.
© 2004 Scott Adams, Inc./Dist. by UFS, Inc.
IT HAS TO INCLUDE LETTERS AND BE AT LEAST SIX CHARACTERS LONG.
HOW ABOUT ABC?
12-5-04
LETTERS AND NUMBERS AND AT LEAST SIX CHARACTERS LONG!!!
www.dilbert.com
FOURSOME?
GAAA!!!

© UFS, Inc.

Bruteforcing on web applications

- Naïve solution: after *n* failed logon attempts, lock account
- Reverse bruteforcing: fix *n-1* attempts and bruteforce accounts
- Make accounts not-enumerable, and block IP address?
- IP address? Really?
- Is this a good idea at all? (hint: proxies)

Cookies for authentication

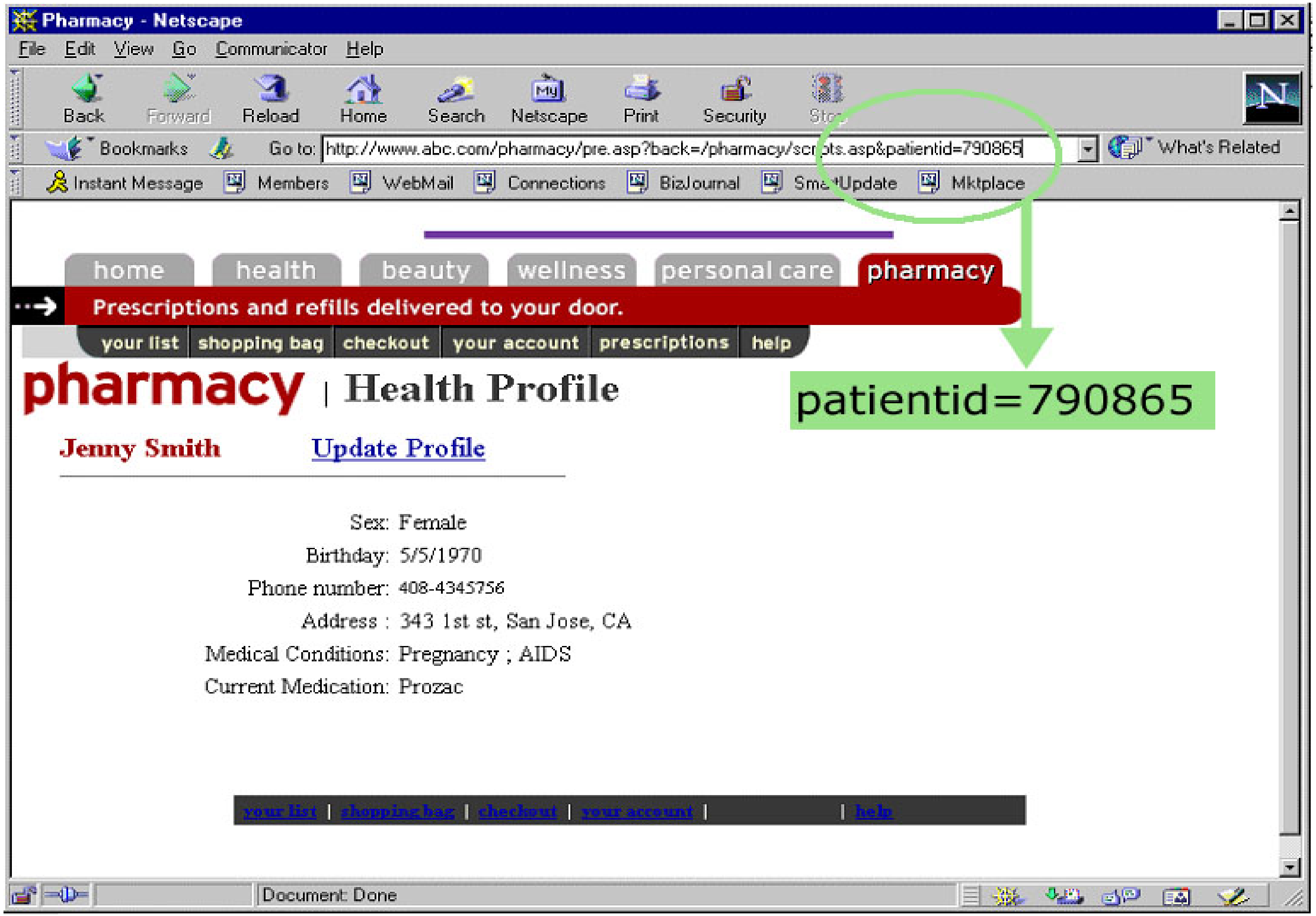
- Mixing of two unstable things leads to an explosive one
- Don't save credentials in cookies (as they can be stolen)
- Prevent reuse of stolen cookies by connecting them to IP addresses and other non-forgeable data
- Don't use cookie duration to force logout (as it can be modified on the client)

Useful reading resource

- Kevin Fu, Emil Sit, Kendra Smith, e Nick Feamster: “Do's and Don'ts of Client Authentication on the Web” http://cookies.lcs.mit.edu/pubs/webauth.html

URL manipulation

URL Manipulation



Pharmacy - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks
What's Related
Instant Message Members WebMail Connections BizJournal SmartUpdate Mktplace
home health beauty wellness personal care pharmacy
your list shopping bag checkout your account prescriptions help
Document: Done

Go to: http://www.abc.com/pharmacy/pre.asp?back=/pharmacy/scripts.asp&patientid=790865

Prescriptions and refills delivered to your door.

patientid=790865

pharmacy | Health Profile

Jenny Smith   Update Profile

Sex: Female

Birthday: 5/5/1970

Phone number: 408-4345756

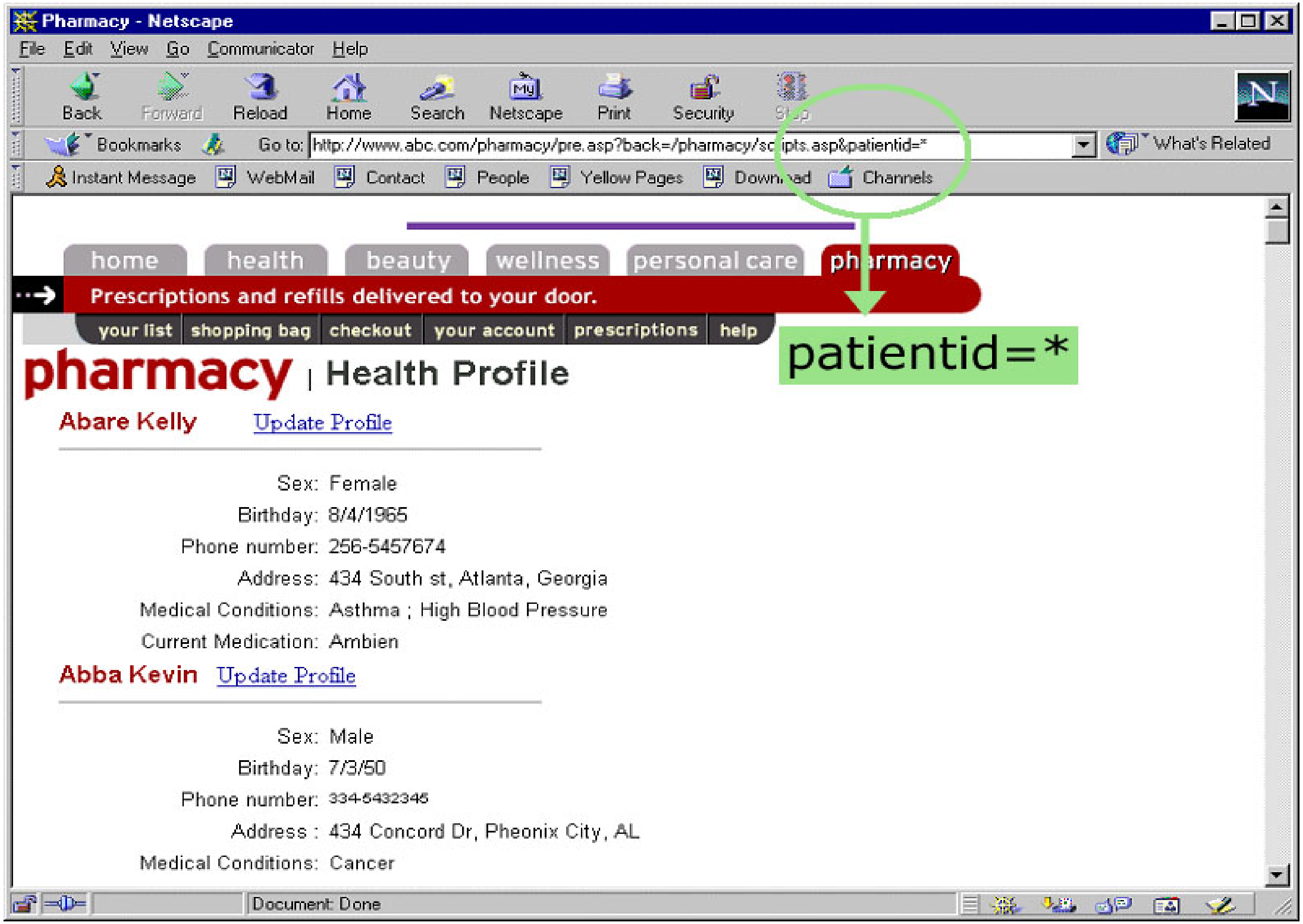
Address : 343 1st st, San Jose, CA

Medical Conditions: Pregnancy ; AIDS

Current Medication: Prozac

your list | shopping bag | checkout | your account | | help

URL Manipulation


Pharmacy - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks
Go to:
What's Related
Instant Message WebMail Contact People Yellow Pages Download Channels
home health beauty wellness personal care pharmacy
your list shopping bag checkout your account prescriptions help
Document: Done

http://www.abc.com/pharmacy/pre.asp?back=/pharmacy/scripts.asp&patientid=*

patientid=*

Prescriptions and refills delivered to your door.

pharmacy | Health Profile

Abare Kelly Update Profile

Sex: Female
Birthday: 8/4/1965
Phone number: 256-5457674
Address: 434 South st, Atlanta, Georgia
Medical Conditions: Asthma ; High Blood Pressure
Current Medication: Ambien

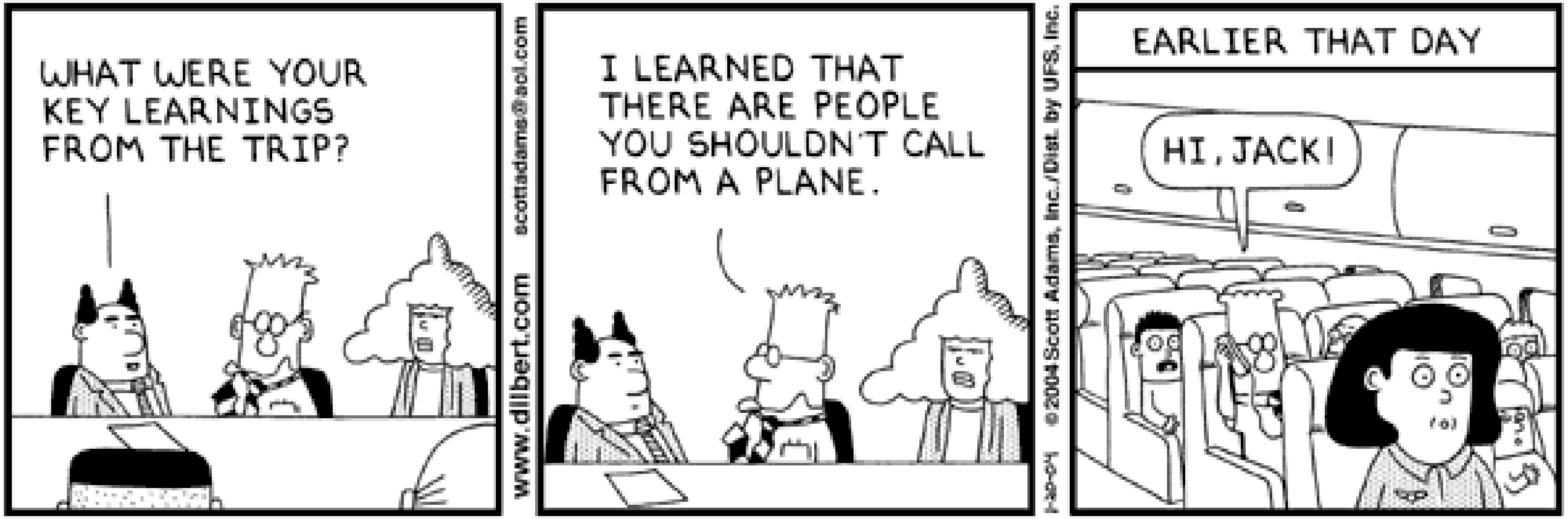
Abba Kevin Update Profile

Sex: Male
Birthday: 7/3/50
Phone number: 334-5432345
Address : 434 Concord Dr, Pheonix City, AL
Medical Conditions: Cancer

URL Manipulation

- This is simple because it's a GET request, however POST requests are also modifiable
- GET requests also stored in history, take care
- Validate parameters against user session
- Validate parameters by using hashing to prevent easy tampering

Session hijacking


WHAT WERE YOUR KEY LEARNINGS FROM THE TRIP?
I LEARNED THAT THERE ARE PEOPLE YOU SHOULDN'T CALL FROM A PLANE.
EARLIER THAT DAY
HI, JACK!
scottadams@aol.com
www.dilbert.com
© 2004 Scott Adams, Inc./Dist. by UFS, Inc.

© UFS, Inc.

Session hijacking

- As we will see, hijacking on network can happen if a MITM attack occurs
- However, since HTTP is stateless, hijacking can occur:
  - By stealing a cookie with an XSS attack
  - By brute forcing a weak session id parameter
- Defenses
  - Using HTTPS
  - Match IP address and session/cookie
  - Use large session IDs possibly changing per interaction