SideJacking: Simpler than MiTM

Robert Graham
Errata Security

David Maynor
Errata Security

Abstract: Web 2.0 applications are not safe to use in WiFi hotspots. In this paper, we will show how session information can easily be sniffed, then replayed in order to hijack a user's account. We will show how to sniff session IDs from WiFi connections, then how to play them back through a proxy in order to hijack accounts. The consequence of this is that users should never use a WiFi hotspot unless they are using VPN or SSL to access their accounts.

Introduction

While the tools for "man-in-the-middle" attacks are fairly easy as hacking tools go, they still are a little complex. Imagine a newspaper writing doing a story on the subject: they would still need a lot of training in order to be able to do the attack.

However, for most Web 2.0 applications, man-in-the-middle is overkill. A much simpler technique involves sniffing the connection information, and replaying it as our own. All that we need is a proxy server that will overwrite the cookie information with the cookies that we sniff from the network.

Web 2.0 provides rich applications on remote websites.

Examples of Web 2.0 applications are e-mail services are:

- rich e-mail applications like Google or Yahoo! Mail
- blogging sites like BlogSpot
- social networking like MySpace, FaceBook, or LinkedIn
- integrated advertising like Google AdSense
- software-as-a-service such as Salesforce.com

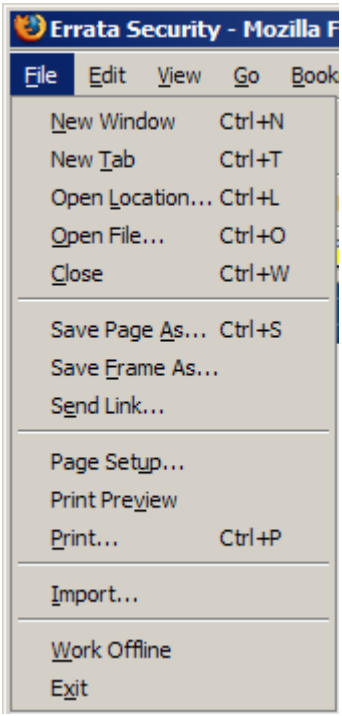
These applications share a common underlying architecture. Running them over SSL would be expensive, so they instead run unencrypted. Usually, only the initial login is done via encryption. Some use a challenge-response using JavaScript. Others do the login over SSL, then switch to non-SSL for the rest of the session.

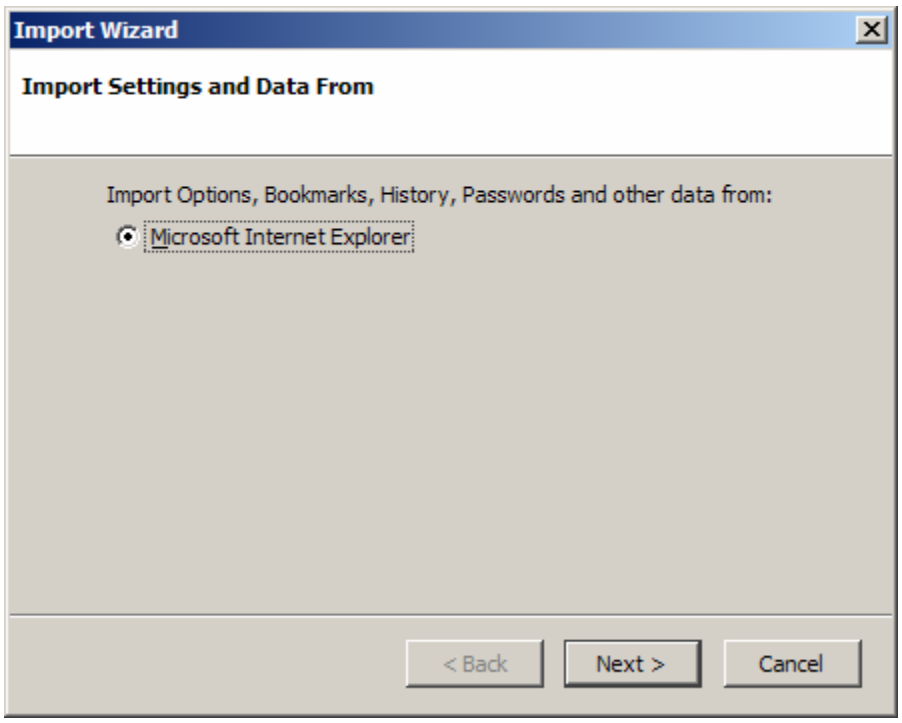
Once the login has completed, the servers send a unique "session identifier" back to the browser. This is either a sent as part of the URL or as a cookie. This information is sufficiently unique and complex that hackers are unlikely to guess such information.

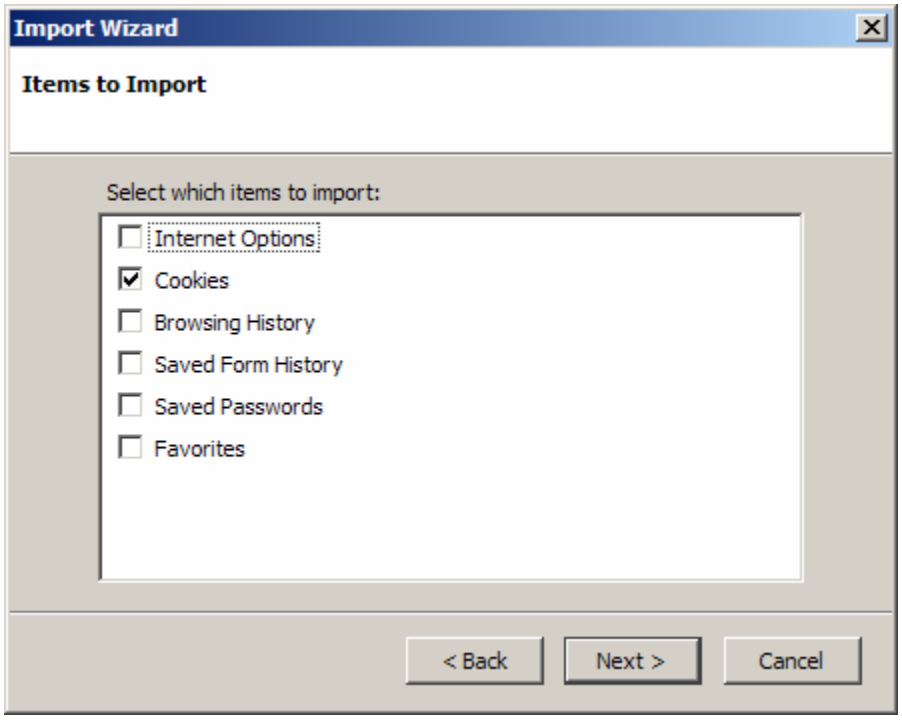
However, since this information is sent in the clear, a hacker who can eavesdrop on the network with a packet sniffer can easily grab this information and gain access to the account.

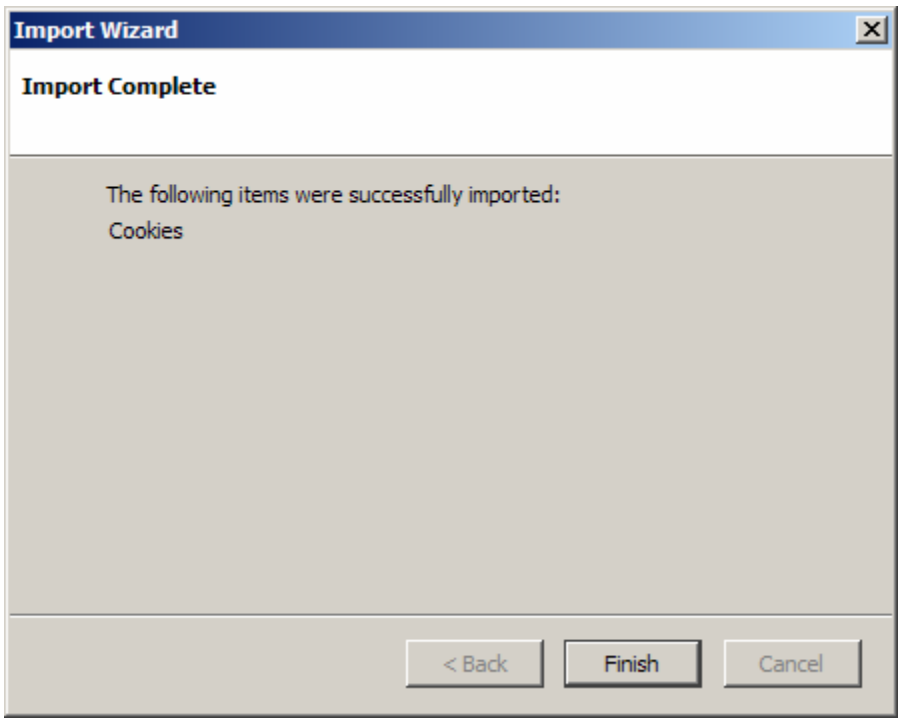
Proving the concept

Conceptually, the process is the same as the “Import” feature in Firefox.




Import Wizard
Import Settings and Data From
Import Options, Bookmarks, History, Passwords and other data from:
Microsoft Internet Explorer
< Back
Next >
Cancel


Import Wizard
Items to Import
Select which items to import:
Internet Options
Cookies
Browsing History
Saved Form History
Saved Passwords
Favorites
< Back
Next >
Cancel



After importing these cookies, you'll find that your Firefox webbrowser just sidejacked your Interne Explorer.

Examples

FaceBook is a social networking site. People post their pictures, link to their friends, and so forth.

The following is an example request sent from Internet explorer to access the main home page for the site:

```
GET /home.php? HTTP/1.1
Accept: */*
Accept-Language: en-us
UA-CPU: AMD64
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT
   5.2; Win64; x64; SV1; .NET CLR 2.0.50727)
Host: www.facebook.com
Connection: Keep-Alive
Cookie: ab_loff=B; test_cookie=1;
  login=edwilliams6%40yahoo.com;
  xs=9ce8f057e644123cc0a9c79c580dedab; c_user=668863853
```

In the case of FaceBook, the unique session information is wholly contained within the cookie. Somebody could copy this cookie information to their own machine in order to hijack this connection. Note that this cookie is built from both the user's e-mail address as well as unique numbers created for this session.

Example: MySpace

```
GET /index.cfm?fuseaction=user&Mytoken=bIyhfs,bdikmvboAla
HTTP/1.1
Host: home.myspace.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.2; en-US;
rv:1.8.0.12) Gecko/20070508 Firefox/1.5.0.12
Accept:
text/xml,application/xml,application/xhtml+xml,text/html;q=
0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Cookie: MSCulture=IP=68.19.21.167&IPCulture=en-
US&PreferredCulture=en-
US&Country=US&timeZone=0&ForcedExpiration=0&USRLOC=QXJlYUNv
ZGU9NzcwJkNpdHk9QXRsYW50YSZDb3VudHJ5Q29kZT1VUyZDb3VudHJ5TmF
tZT1Vbml0ZWQgU3RhdGVzJkRtYUNvZGU9NTI0JkxhdGl0dWRlPTMzLjgwMD
QmTG9uZ2l0dWRlPS04NC4zODY1JlBvc3RhbENvZGU9JlJlZ2lvbk5hbWU9R
0E=; NGUserID=a2825d3-5776-1183134349-1;
MYUSERINFO=MIICxQYKKwYBBAGCN1gD7qCCArUwggKxBgorBgEEAYI3WAMB
oIICoTCCAp0CAwIAAQICZgMCAgDABAhB4mo%2f1h9xBgQQYvXJVPdX%2bQX
TfjbGRkFp4wSCAnCGVO7%2begm3aHjvbIeWBvfnpxxOK8KrLRJ9FFJcESTo
aUn6Zs5UCv%2f9O37XED89XEw1QHq%2bxJq%2fQ8k4oxLShqPrIGJtChKSf
bzCdpi1ambW%2f7cUp2l1%2fyNPaOLuuSFD53e3yCkNTucmXTDxTkpxm9%2
bbXlTPDm%2fCTYnjEAh5bCw%2fNt70t8As0v6MUGBYMRG%2bOsve9Ou30A%
2fZRBTON1QRwDi7OK0dhR65qr%2ftV8KIWO9hUWX0je1BkLVNMjGrkgmFRn
6BcyHU47cHg42BKC4PCTvyss7fNTZMWD1pexGlFJHdxIpRQe3WvGVPBXFEG
Yl%2fO%2fHr3pzP0yH2L%2fkm2sKhFLWJoSp0eCR5r6a%2fmtBgv%2fBaGf
7b52udVLIV1%2fE9RHQLSDtNkliY1BnuW%2fH%2bzHRWYbrNiEbU%2bDwHg
KhWbhZrEnZRcrDsARg7YxWKjDE5RcEg37Kq%2fTlLyvMnjkZq4adSpc4oAq
YeDASnS2T%2fxhqPIAMs9r9d16y8XgNNs%2ftLZS9GXfj6Njiju4J2nkiYl
w5f%2b0O%2bmiZzfJkKz22ecW0RAsx7C8ASuVFrMpYbcmu1K5BZn7JSOR3K
5jOj7LLUL4aOXWvAJunlgQBYF5WdZ7NYHOlWWENSLnFMzfmso4Qph%2bjI2
d7yZ1AWaeY1reZMrQjH6U2lTz8XbUr%2bRA5Lw5icuAANHhO4tB4tedt97I
n%2fOLzywHIJJGjVqA0bHyuLpqgtX1D9%2bDg3c6AW%2ftU5na5C%2bCnVZ
ebL46zxSGJxAUQWY6bos6aD3Da8U79Hj%2fkYx2G%2bbKkSxcNmPiWba41c
k6AV1lP1WTFSFUTS2w7bSFh0lGPqPRI%3d;
LASTUSERCLICK=%7bts+'2007-06-29+09%3a26%3a33'%7d;
SplashDisplayName=Edward;
```

```
DERDB=ZG9tYWluPS5teXNwYWNlLmNvbSZ0bGQ9Y29tJnNtb2tlcj0mc2V4c
HJlZj0mdXR5cGU9MiZyZWxpZ2lvbmlkPSZyZWdpb249MTEmcG9zdGFsY29k
ZT0zMDMyNyZtYXJpdGFsc3RhdHVzPVMmaW5jb21laWQ9JmhlaWdodD0mZ2V
uZGVyPU0mZnJpZW5kcz0wJmV0aG5pY2lkPSZhZ2U9MjMmYm9keXR5cG0mY2
hpbGRyZW5pZD0tMSZjb3VudHJ5PVVTJmRhdGluZz0wJmRyaW5rZXI9JmVkd
WNhdGlvbmlkPSZyZWxhdGlvbnNoaXBzPTAmbmV0d29ya2luZz0wJmRpc3Bs
YXluYW1lPUVkd2FyZCZmcmllbmRpZF9pbnQ9MjA3MDg4ODM3JmlwYWRkcmV
zcz0nNjguMTkuMjEuMTY3JyZzY2hsPTAmc2NobD0wJnNjaGw9MCZncncA9MC
ZncncA9MCZncncA9MCZjdWx0dXNlcnByZWY9MTAzMyZyc2lfd2FudD0w;
ME=edwilliams6@yahoo.com;
GADC=EUD=0:0:NjlhNDc3YTIxM2U3OGU3YbfPsEnAjrXt2QwevOmw7WVG-
vABWJkpHNDCM8g1H-
4uRhxdc3HKIzgk0tszUJczcpe9tPZA9q2rhcOdRnxEhq3HKxzKoZc2uBoE3
jxL5R74; UNIQUELOGINTAKEOVER_207088837=%7Bts%20%272007-06-
29%2009%3A26%3A16%27%7D; AUTOSONGPLAY=0;
CMSCAP=CDATE=633187059724355385&633147489413520857_ct=1&633
147489413520857_exp=633192243724355385
Pragma: no-cache
Cache-Control: no-cache
```

This information is much more complex than FaceBook. In previous presentations, we have presented this information as an example of “data seepage” – the fact that such we can decode this information we sniff on the wire to find more details about you. However, in this case, we can use this information to hijack your MySpace account.

Here is a Yahoo! Mail session:

```
GET
/dc/launch?action=welcome&YY=1940796309&.rand=cbg06l0h42ue0
HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg,
image/pjpeg, */*
Accept-Language: en-us
UA-CPU: AMD64
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT
5.2; Win64; x64; SV1; .NET CLR 2.0.50727)
Host: us.f351.mail.yahoo.com
Connection: Keep-Alive
Cookie: YM.stck=1183135228;
YM.CGP_edwilliams6=suc=405&btph=281&spp=1&res=932x662;
YM.LF=; YM.C=1; YM.Gen=; B=buvfkfd2ptjfp&b=3&s=3h;
F=a=4k_UMHYMvT3qIwYWUv81QWdVI.YEeqXFCXiPMrDEgdwstKhZwzo9cIj
kt8fq1rZf9UxDcuM-&b=rQwy; C=mg=1; YLS=v=1&p=1&n=0;
Y=v=1&n=97jgv0t1k6h63&l=43m8bb80ciwxw/o&p=m2e0tjo013000000&
r=f3&lg=en-US&intl=us; PH=fn=F5SQeqmSRCJ15Oc-;
T=z=l7ShGBlP6lGBCESrpPYVCyQNDI2BjU0TjcxMTMyTk4-
```

```
&a=QAE&sk=DAA71mlem698UC&d=c2wBTXpVeEFUSXpPVEEyTmpRMU9Uay0B
YQFRQUUBdGlwAUVOUkZuQwF6egFsN1NoR0JBN0U-
```

Here is a Blogger connection:

```
GET / HTTP/1.1
Accept: */*
Accept-Language: en-us
UA-CPU: AMD64
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT
5.2; Win64; x64; SV1; .NET CLR 2.0.50727)
Host: sidejacking.blogspot.com
Connection: Keep-Alive
```

Notice that it doesn't actually send cookies to the site at blogspot.com, but instead sends the cookie information when it gets the JavaScript, stylesheets, and images over at blogger.com:

```
GET /dyn-css/authorization.css?blogID=37798047 HTTP/1.1
Accept: */*
Referer: http://erratasec.blogspot.com/
Accept-Language: en-us
UA-CPU: AMD64
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT
5.2; Win64; x64; SV1; .NET CLR 2.0.50727)
Host: www.blogger.com
Connection: Keep-Alive
Cookie: S=blogger=pv-C9F5CqCOAQ_WL97qk4A; __utmc=150635877;
blogger_SID=DQAAAG4AAABi6K2SvC280UW5vEo9wyeUrlX_AsA9Lt7dnh7
lVsl6dpihMLshQw-
dEWylxJ0DksCXhCT4wZhMyIFbxN06Wpy614MqN_TmmtUTaXhtNT3tiViscd
jIwBtvXbLxKx6SO6p6HQB9tVXl1ZxbV1BvStXg; B2I=Sidejacking
```

The above example is interesting. What I did was create a blog "sidejacking.blogspot.com", and then went to surf the blog "erratasec.blogspot.com". This meant that surfing any blog at *.blogspot.com would allow the test blog to be hijacked.

How to build a tool

Building a tool is fairly straightforward.

The easiest tool to create would be to dump the cookie information into Internet Explorer files, then use the Mozilla Import tool to import them in. This works when the session can be “saved” to the computer (persistent cookies). However, a lot of websites require you to log back in every time you restart your computer, and do not save the cookies to the disk. Therefore, you need a live tool.

First, we need a packet sniffer that can eavesdrop on the network. In our own tool, we are going to use the “Ferret” codebase. However, anybody else can do it pretty simply. Just sniff all TCP packets. When the TCP packet starts with “GET /”, then parse out the URL, “Host:”, and all the “Cookie:” fields. This doesn’t need any complicated TCP reassembly in order to work.

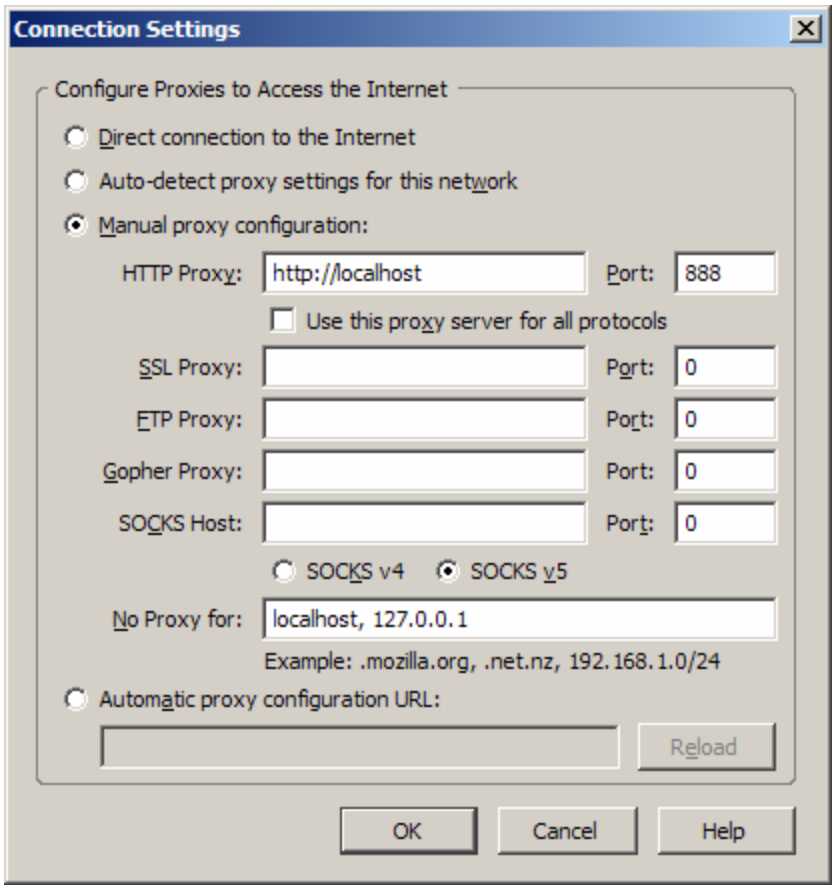
The easiest way is just to sniff the “Cookie:” fields from the client-side. You may also want to sniff the “Set-Cookie:” fields coming back from the server. However, client-side sniffing is probably the best because you sometimes need session information from the URLs as well.

Next is to grab a proxy server and hook it up with the sniffer. The purpose of the proxy server is to change the cookie information to conform to what we’ve sniffed from the wire. You can either write your own simple proxy, or change one like ‘squid’.

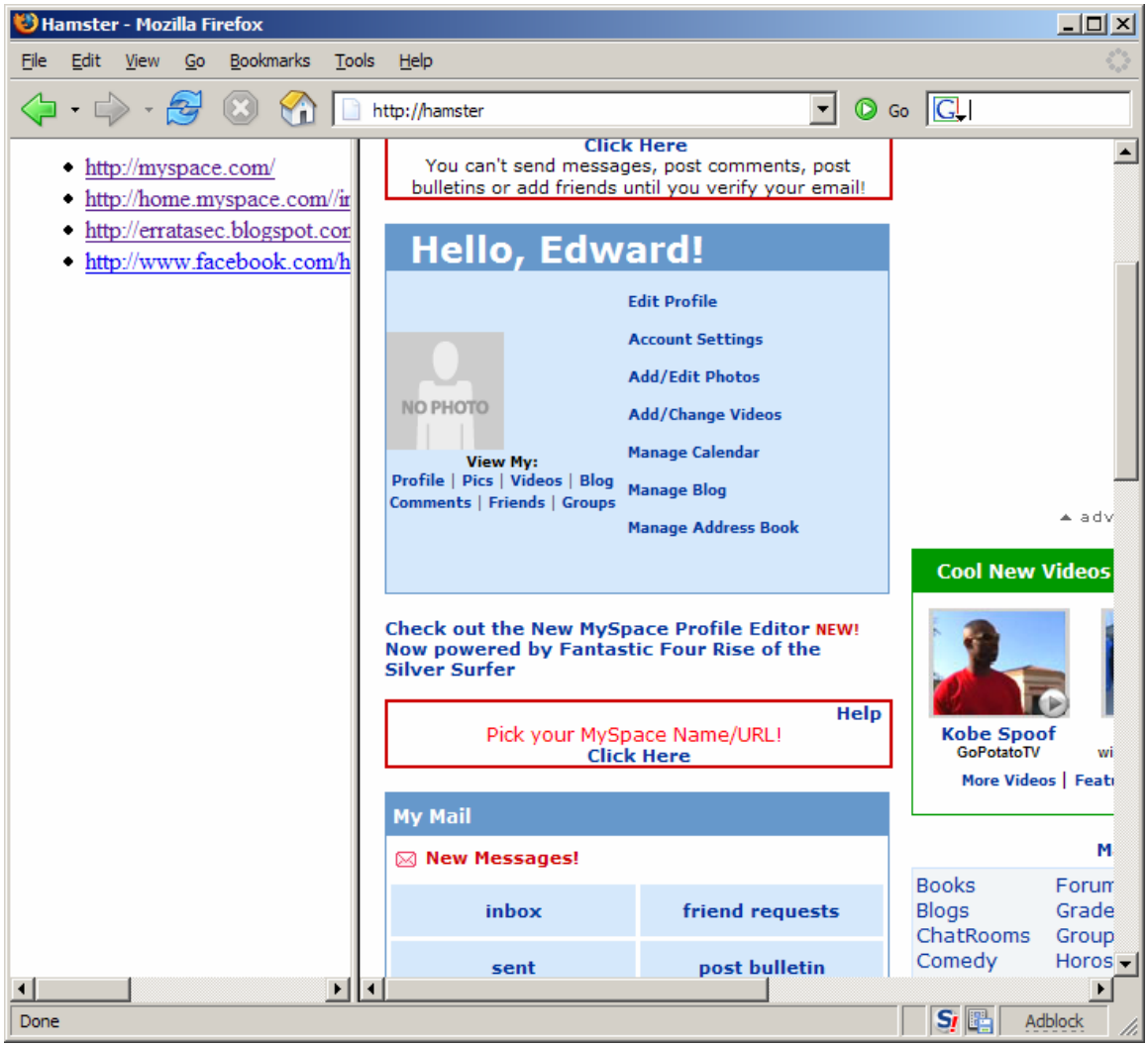
The Hamster Tool

We’ve built our own little proxy + sniffer combination called “Hamster”.

The user starts the sniffer and the proxy. The user then goes into their web browser to point it at the proxy:



At this point, the user visits the web page “http://hamster”. This web page will not go out to the Internet. Instead, the it will open a page that shows the list of current websites that the system knows about from sniffing the network. When the user clicks on one of those websites, they will get a “sidejacked” version of the webpage.



This tool has a number of problems. For example, when multiple people are surfing a wifi connection, it just uses the latest cookie information from all of them. This means that if you are trying to use one person's webpage, you may suddenly get somebody else's in the middle of a session.

The practical dangers of sidejacking

What can a sidejacker do with a page?

Most Web 2.0 applications are pretty robust. In order to get to the core account settings, such as changing the password, the user must re-enter the password and/or go through an encrypted connection.

However, most other changes are allowed. The sidejacker will be able to post blogposts, upload pictures, read e-mail, and that sort of thing.

Defense against sidejacking

If you can, go through a VPN connection back home. This slows down your connection though.

Another defense is to use SSL. If you manually edit the URL to point to “https:” instead of “http:”, you’ll get an encrypted connection. Unfortunately, this only works for a few sites (such as Gmail), and you’ll often have already sent your session id across the wire before you get a chance to change it.

Conclusion

The dangers of Web 2.0 over public WiFi has long been known, but it has been assumed that special man-in-the-middle tools were needed. We have shown that rather than hijacking them by intercepting and redirecting traffic, we can “sidejack” them by eavesdropping and replaying traffic.