Computer Science 161 – Computer Security
Instructor: Tygar
3 November 2010

Midterm 2

Instructions: Keep answers concise

NAME_____ SECTION (TA/Hour) _____

Question 1	*Question 2*	*Question 3*	*Total*

Computer Science 161 – Computer Security
Instructor: Tygar
3 November 2010

Instructions: Keep answers concise (write the answer to question 1 only on this sheet – use back if necessary).

NAME_____ SECTION (TA/Hour) _____

Midterm 2 - Question 1

One problem that software vendors face is providing patches to update software. Consider the following patching process: when a user runs a program, it will contact the vendor's web site and see if an update is available – if it is, it will download and install the update.

(a) Using the set of 13 principles that we discussed, give a threat analysis of major security problems in this software patching process. Use at least four of the 13 principles.

(b) Can we use SSL to address some of the problems identified above? Identify which ones, and explain how we could use SSL to solve the problems. For those that are not addressed by SSL, explain why.

(c) In addition to SSL, what other security steps can we take to solve these problems? Try to give a complete solution to your list of threats.

Computer Science 161 – Computer Security
Instructor: Tygar
3 November 2010

Instructions: Keep answers concise (write the answer to question 2 only on this sheet – use back if necessary).

NAME_____ SECTION (TA/Hour) _____

Midterm 2 - Question 2

Look in the appendix to Question 2 and read the attached article on Firesheep and a proposed "solution" by Tom Cross and Takehiro Takahashi

(a) Draw a diagram showing the Cross-Takahashi proposed "solution." The diagram should include all essential parties in a session including the client, server, and any security relevant parties. Indicate which portions of the communication are over secure channels and which are not.

(b) Sketch a handshake protocol for the Cross-Takahashi protocol, using the SSL handshake as a model.

(c) Describe some ways of attacking the Cross-Takahashi "solution."

Appendix for problem 2
(from PC World http://www.pcworld.com/printable/article/id,208727/printable.html)

Firefox Add-on Firesheep Brings Hacking to the Masses

By Ian Paul

Oct 25, 2010 12:14 pm

Want to hack someone else's Amazon, Facebook, Twitter or Windows Live account in just one click? A Firefox extension called Firesheep claims you can by hijacking a person's current user session over an open Wi-Fi connection. I tested the extension out and to my horror it works as advertized - almost that is.

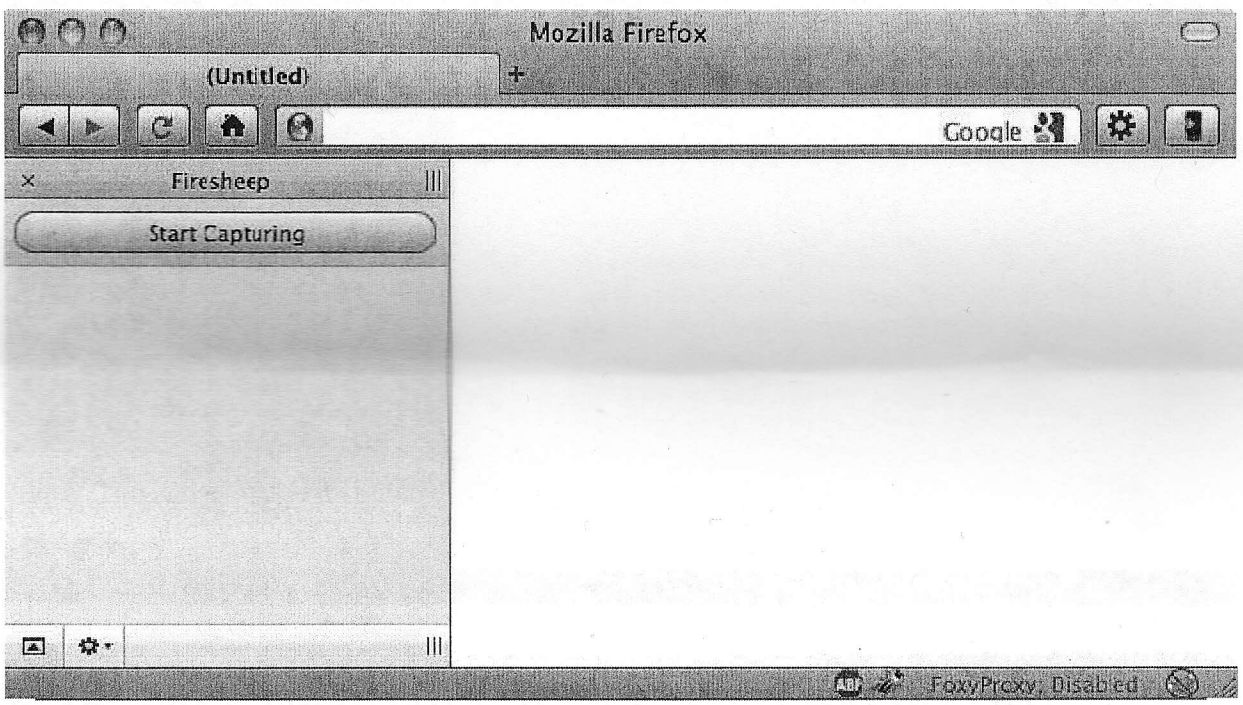
Firesheep was created by Seattle-based software developer Eric Butler who said he created the extension to highlight the security risks associated with session hijacking, also known as sidejacking. Firesheep targets 26 online services, and includes many popular online services such as Amazon, Facebook, Foursquare, Google, The New York Times, Twitter, Windows Live, Wordpress and Yahoo. The extension is also customizable allowing a hacker to target other Websites not listed by Firesheep. While Firesheep sounds scary (and once again highlights the security concerns of using open Wi-Fi) the new Firefox extension is not as frightening as it sounds.

Firesheep is basically a packet sniffer that can analyze all the unencrypted Web traffic on an open Wi-Fi connection between a Wi-Fi router and the personal computers on the same network. The extension waits for someone to log in to any of the 26 sites listed in Firesheep's database. When you log in to Amazon, for

example, your browser's Amazon-specific cookie communicates with the site and contains personally identifying information such as your user name and an Amazon session number ID.

As your browser swaps cookie information back and forth with the Website a third party can hijack that communication and capture info including your user name and session ID. Typically, the cookie will not contain your password. But even without your password, the fact that Firesheep has snagged your session cookie means that a hacker can, at least in theory, access your account and gain virtually unrestricted access. If the hacker got your Yahoo Mail cookie they could send an e-mail, if it was Facebook they may be able to post a message and so on. Any operations that require your password, however, such as accessing your credit card information on Amazon should not be possible using Firesheep.

Firesheep put to the test



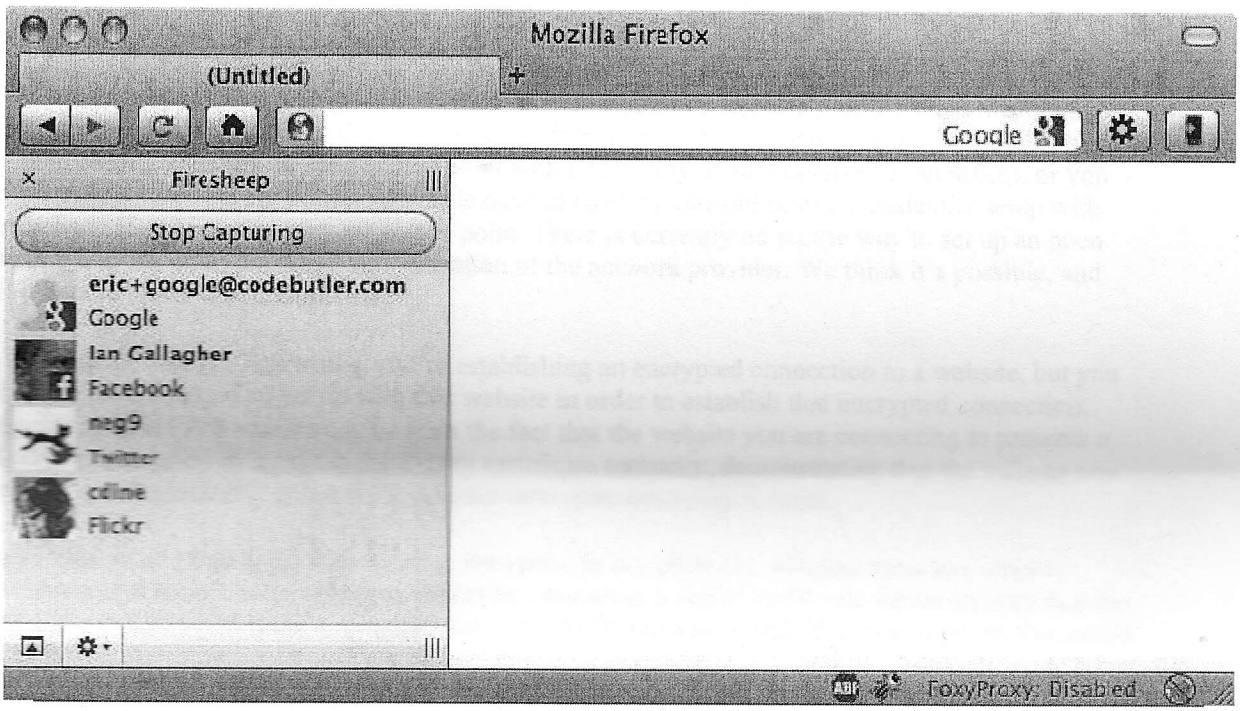
Since I wasn't close to a public Wi-Fi hotspot today, I tested Firesheep on my own home network using Firefox 3.6 for Mac OS X. The problem is I use WPA2 encryption at home, a Wi-Fi security standard that encrypts all user traffic going between your PC and the router. So the only way I could test Firesheep was on my own machine, which I did by browsing on both Firefox and Chrome.

To get started I installed Firesheep on Firefox, and then opened it up by clicking on View>Sidebars>Firesheep. I then saw a blank sidebar with a button at the top that said "Start Capturing." Once I clicked the button to start snooping, the extension asked for my computer's master password so that the extension could access and make changes to my machine. Needless to say, this is not something I would recommend you try on your own computer.

After the sidebar was working it started grabbing user IDs as promised for sites I logged in to including Amazon, Facebook, Google and The New York Times. Firesheep was able to grab my user name and profile photo (when available) and then display each account in the sidebar.

Theoretically, if I had tested this system over an unencrypted Wi-Fi network at a cafe, I should have been able to simply click on any of the accounts I saw in the Firesheep sidebar and then gain almost unrestricted access to the account. But in my tests that's not what happened.

Firesheep gets corralled



After the sniffing was done, I was supposed to be able to click on each user ID listed in my sidebar and then see my online accounts. Obviously, I was able to do this when using an account I'd logged in to using Firefox since the browser contained my actual session IDs as well as the stolen cookies sitting in Firesheep. But when I tried to gain access to my New York Times account that I'd logged in to using Chrome, Firesheep couldn't give me access to my account in Firefox. This was despite the fact that my user name and profile picture appeared in Firesheep. It's also important to note that once I logged out of any of the online services I tested, I could not use Firesheep's stolen cookie to log back in. . . .

(from IBM Frequency X blog http://blogs.iss.net/archive/WirelessSolution.html)

A new solution to wireless security issues

Posted by **Tom Cross and Takehiro Takahashi** on October 27, 2010 at 12:43 PM EDT.

Eric Butler's Firesheep plugin for Firefox has kicked off a firestorm of discussion. It's a pretty nicely designed demonstration of the security risks that you face when you connect to open wireless networks. The plugin displays all of the potential victims who are using the wireless network you are connected to and gives you one click access to their accounts on services like Facebook and Twitter. The release of this tool has resulted in renewed warnings about open wifi as well as calls for greater use of HTTPS by websites.

X-Force Research has been working on a more fundamental solution to these problems and this blog post is the first time we've ever discussed our approach on the Internet. Wifi network security is a hobson's choice. You can either set up an open access point with no security at all that anyone can access, or you can set up an encrypted access point, but people have to have a password or other credential setup with you beforehand in order to access your access point. There is currently no secure way to set up an open access point that has encryption and authentication of the network provider. We think it's possible, and we call it Secure Open Wireless Access.

If you think about how HTTPS works, you're establishing an encrypted connection to a website, but you don't have to have a password set up with that website in order to establish that encrypted connection. The security of an HTTPS session comes from the fact that the website you are connecting to presents a digital certificate, signed by a trusted third party certificate authority, demonstrating that the website you are connecting to legitimately controls the domain name you are trying to reach.

You can do the exact same thing with wireless networks. In our proposal, wireless networks would establish encrypted connections with their clients by presenting a digital certificate demonstrating that the operator of the access point is the legitimate user of the SSID associated with that access point. You could even use domain names as SSIDs and use off the shelf SSL certificates.

For example, IBM could set up an open wireless network with the SSID "ibm.com." When you connect, our access point would send down a digital certificate for "ibm.com," and your wireless client would establish an encrypted connection with us, knowing that because the name in the certificate is the same as the SSID, the network you are connecting to must be run by IBM.

The result would be that when you open up your wireless client you could establish secure, encrypted connections to networks operated by people (or companies) that you trust, knowing that those networks are really operated by the people (or companies) that they claim they are operated by without needing to have a password.

We've been thinking about this for a long time, we've looked at it from a lot of different angles, and we're pretty sure it would work, and that it would not interfere with any legitimate thing people are doing with wifi today. Unfortunately the process associated with moving from an idea like this to an actual implementation in real networks can be frustratingly slow. We have a patent pending and we have a paper with a complete technical discussion that we have been trying to publish. The wheels of the peer review machine turn slowly, but hopefully we will publish our paper within a few months and you will read about it here.

However, publishing a paper is not enough to get this idea into your laptop. If you are reading this post and you are a certificate authority, wireless access point manufacturer, wireless client software developer, or wireless network operator, and you'd like to understand this idea in more detail, please get in touch with us. We would like to see this idea become a reality. If you can help, let's talk.

Computer Science 161 – Computer Security
Instructor: Tygar
3 November 2010

Instructions: Keep answers concise (write the answer to question 3 only on this sheet – use back if necessary).

NAME_____ SECTION (TA/Hour) _____

Midterm 2 - Question 3

For years Alice Airlines has let its passengers check in at self-service stations inside the airport—only now are they adapting their software to support online check-in. However, in moving from a kiosk environment (which tightly regulates the user's input) to an online system that allows for a wider variety of input, the airline's software has become vulnerable to a code injection attack.

(a) What is a code injection attack, and what is the root cause of code injection vulnerabilities?

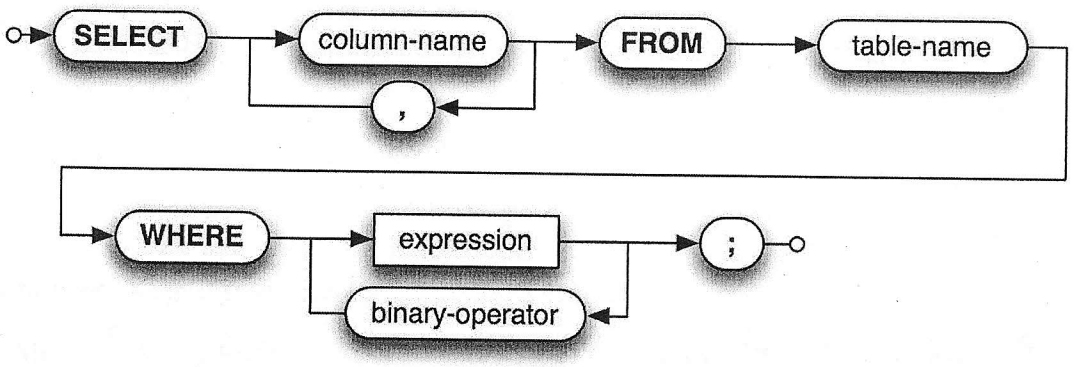
(b) Consider the pseudo-code below that contains a SQL code injection vulnerability. Craft an input string (a `confirmation_code`) that lets an attacker tell whether a specific passenger named 'Bob Loblaw' is on any flight today. Consult the attached appendix for help with SQL syntax.

(1) `confirmation_code` ← *get input from the user*
(2) `todays_date` ← *get today's date from the system (e.g., '11/03/2010')*
(3) `query` ← `"SELECT name, flight FROM passengers WHERE conf = '" + confirmation_code + "' AND flight_date = '" + todays_date + "';"`
(4) `name, flight` ←*executeQuery*(`query`)
(5) *display* `name` *and* `flight` *to user for check-in confirmation*

(c) Some SQL code injection attacks like the one above require the attacker to know a little about the structure of the database (e.g., the names of its tables and columns). What are some strategies an attacker could use to learn the column names (`name`, `flight`, `flight_date`, and `conf`) in the example above?

Appendix for problem 3b

The following syntax should define all the SQL necessary to solve problem 3b:



The SELECT statement returns any rows in a given table that match the given expression. An expression may include column-names and literals, and make use of the usual operators. Below is a list of valid binary operators, *in order of precedence from highest to lowest.*

||	String concatenation
* /	Arithmetic
+ -	
< <= > >=	Inequality
= !=	
AND	Logic
OR	

Example **passengers** table:

Name	flight	date	conf
'Eve'	155	'10/31/2010'	'JUD824'
'Barry'	177	'11/02/2010'	'SSR832'
'Frank'	177	'11/03/2010'	'JJK921'

Example SELECT statements. Note that text following the double-dash (--) is considered a comment.

```
SELECT name FROM passengers WHERE date <= '11/02/2010';-- returns the
names Eve and Barry.
```

```
SELECT name FROM passengers WHERE date <= '11/02/2010' AND flight =
177; -- returns only the name Barry.
```