EECS 1012: Introduction to Computer Science

November 21, 2016

Security

- A substantive problem, of course with a range of issues
- Here we look at one example related to JavaScript and HTML
  - Cookies

Cookies

- One ‘problem’ with web pages is that they lack persistent data
- Can use some central server for data, but what about less persistent, local storage?

Cookies are the answer.

Cookies

- Persistent data (stays for a session, or for longer).
- Accessible to all web pages from the same domain
  - Although is can be circumvented slightly

Low-level API

- document.cookie = “name=value”;
  - Adds this (name,value) pair to the list of cookies
- document.cookie = “name=value;expires=date”;
  - Adds this (name,value) pair to the list of cookies and sets its expiry date
- document.cookie - all the pairs

Higher-level API

```
var ajax;
function getAllCookies() {
  var pairs = document.cookie.split("; ");
  var cookies = []
  for(var i=0;i<pairs.length;i++) {
    var pair = pairs[i].split("=");
    cookies[pair[0]] = pair[1];
  }
  return cookies;
}
```

```
function setCookie(cookie, value) {
  var date = new Date();
  date.setTime(date.getTime()+(365*24*60*60*1000));
  document.cookie = cookie + "=" + value + ";expires="+date.toGMTString();
}
```

Cookie persists for a year (why not)

Example

- Two web pages from the site site, they can share information (number of loads of the page)

```
function go()
{
   var cookies = getAllCookies();
   if(cookies['nvisitz'] == undefined) {
     setCookie('nvisitz', 1);
   } else {
     setCookie('nvisitz', Number(cookies['nvisitz']) + 1);
   }
   cookies = getAllCookies();
   var nvisitz = cookies['nvisitz'];
   document.getElementById('output').innerHTML = cookies['nvisitz'];
}

onload=go;
```

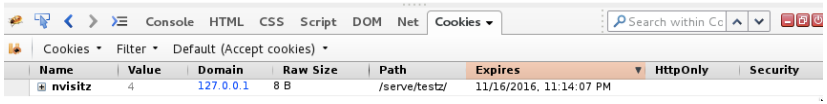
index.html

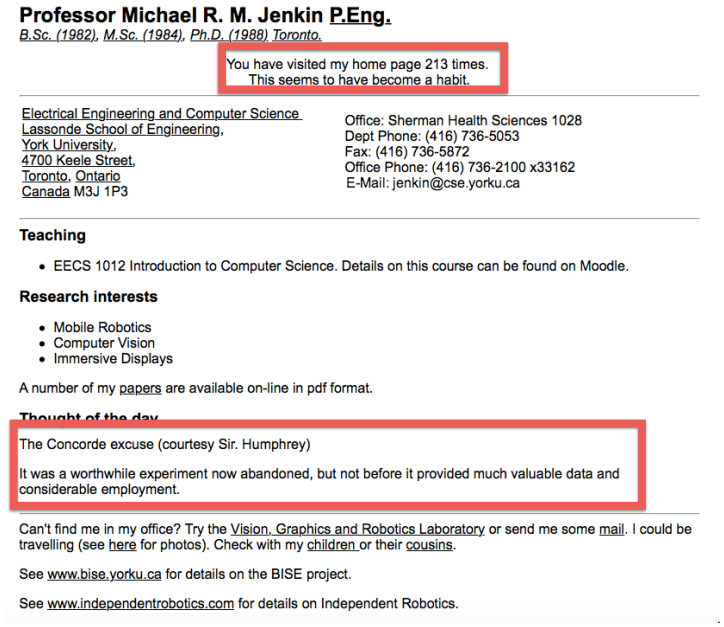
```
function go()
{
   var cookies = getAllCookies();
   document.getElementById('output').innerHTML = cookies['nvisitz'];
}

onload=go;
```

index2.html

Firefox Cookies





Professor Michael R. M. Jenkin P.Eng.

B.Sc. (1982), M.Sc. (1984), Ph.D. (1988) Toronto.

You have visited my home page 213 times.
This seems to have become a habit.

Electrical Engineering and Computer Science
Lassonde School of Engineering,
York University,
4700 Keele Street,
Toronto, Ontario
Canada M3J 1P3

Office: Sherman Health Sciences 1028
Dept Phone: (416) 736-5053
Fax: (416) 736-5872
Office Phone: (416) 736-2100 x33162
E-Mail: jenkin@cse.yorku.ca

Teaching

- EECS 1012 Introduction to Computer Science. Details on this course can be found on Moodle.

Research interests

- Mobile Robotics
- Computer Vision
- Immersive Displays

A number of my papers are available on-line in pdf format.

Thought of the day

The Concorde excuse (courtesy Sir. Humphrey)

It was a worthwhile experiment now abandoned, but not before it provided much valuable data and considerable employment.

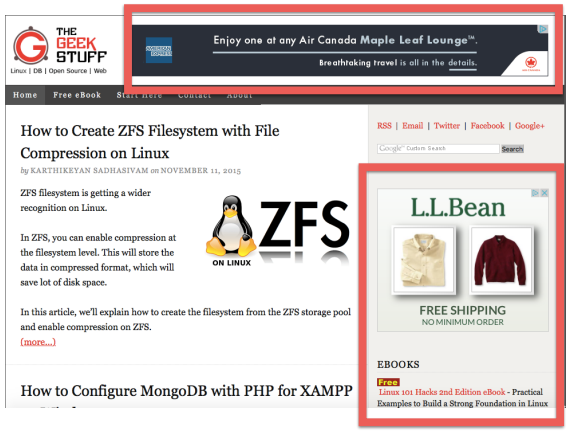
Can't find me in my office? Try the Vision, Graphics and Robotics Laboratory or send me some mail. I could be travelling (see here for photos). Check with my children or their cousins.

See www.bise.yorku.ca for details on the BISE project.

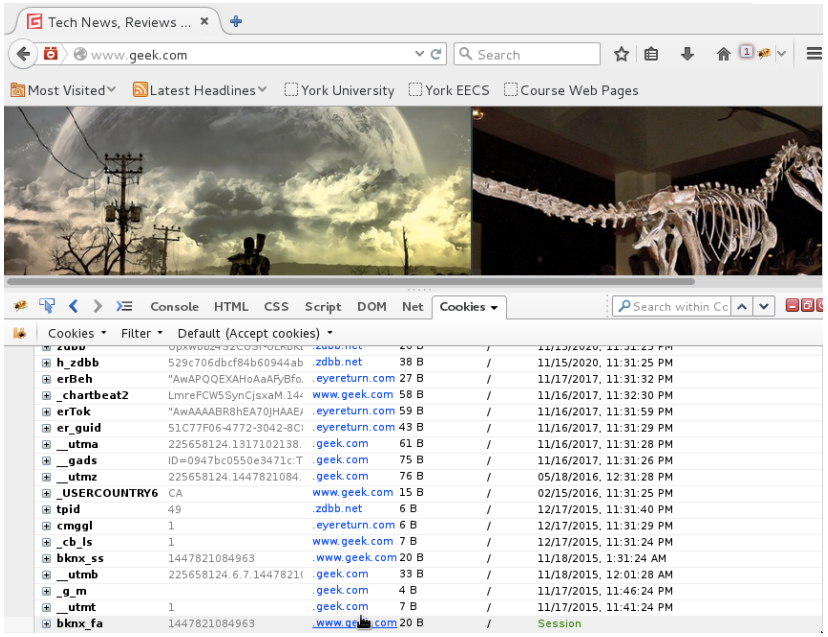
See www.independentrobotics.com for details on Independent Robotics.

Security…

- Now, in theory cookies are only accessible from the domain in which were created….but…
- A given page might have content from more than one site on it.



Each site may (likely) has its own cookies



Security

"On the Internet, nobody knows you're a dog."

Cookies

- Allow a company to know (track) all of your browser's operations within its page on your machine.
  - Want privacy? Make sure you clear your cookies.
- Potential for substantial security issues
  - 'Remember Me' services.

Web Security

- JavaScript runs in the target's machine.
- The 'sandbox' it runs in is intended to prevent any code from being malicious.
- But, poorly written code can still do damage.

Lets consider an example

- In the last lab you are writing a piece of code that uses a database of albums.
- You displayed the graphics associated with the album (or displayed the titles)

```
var ajax;
function go()
{
  ajax = new XMLHttpRequest();
  ajax.onreadystatechange = ajaxProcess;
  ajax.open("GET", "http://127.0.0.1:8000/sql?query=select * from collection");
  ajax.send(null);
}

function ajaxProcess() {
  alert(ajax.readyState);
  alert(ajax.status);
  if((ajax.readyState == 4)&&(ajax.status == 200)){
    ajaxCompleted(ajax.responseText)
  }
}

function ajaxCompleted(text) {
  var output = document.getElementById("output");
  var data = JSON.parse(text);
  var i;
  for(i=0;i<data.length;i++) {
    output.innerHTML += data[i].album + "<br>";
  }
  for(i=0;i<data.length;i++) {
    var p = document.createElement("img");
    p.src =data[i].cover;
    p.width = 64;
    p.height = 64;
    output.appendChild(p);
  }
}
```

Is this code safe to XSS? (no)

```
var ajax;
function go()
{
  ajax = new XMLHttpRequest();
  ajax.onreadystatechange = ajaxProcess;
  ajax.open("GET", "http://127.0.0.1:8000/sql?query=select * from collection");
  ajax.send(null);
}

function ajaxProcess() {
  alert(ajax.readyState);
  alert(ajax.status);
  if((ajax.readyState == 4)&&(ajax.status == 200)){
    ajaxCompleted(ajax.responseText)
  }
}

function ajaxCompleted(text) {
  var output = document.getElementById("output");
  var data = JSON.parse(text);
  var i;
  for(i=0;i<data.length;i++) {
    output.innerHTML += data[i].album + "<br>";
  }
  for(i=0;i<data.length;i++) {
    var p = document.createElement("img");
    p.src =data[i].cover;
    p.width = 64;
    p.height = 64;
    output.appendChild(p);
  }
}
```

Is this code safe to XSS? (no)

Suppose..

- That somehow the user can enter new album information
  - Makes sense, should be able to do that in some updated version of the software.

```
for(i=0;i<data.length;i++) {
  output.innerHTML += data[i].album + "<br>";
}
```

Album

- <button onclick="foo();">Button</button>
- Then the displayed 'album' will create a button in the output..
- Or we could just have a new page created over the background.

Summary

- Any software running anywhere has to deal with security issues
- This is even more critical for web-based software that when let out into the wild can (will) be run by other users, in other environments, and in ways that you might not have intended.