Internet Security

By Bradley Frank

Harvard-MIT Data Center at The Institute for Quantitative Social Science

Presented on 18 July 2013

www.hmdc.harvard.edu

www.iq.harvard.edu

bfrank@hmdc.harvard.edu

Online Threats and Attacks

Wealth, revenge, and power are as irresistible to today's criminals as much as they were the criminals of an era before computers. Only the tools have changed over time.

Social Engineering and Malware

The act of social engineering is a *confidence trick*, similar to what a confidence artist might employ. The goal, in the context of computer security, is to leverage a person's assumptions, biases, or lack of knowledge about a system in order to obtain information about, or access to, that system. This can be accomplished electronically, over the phone, or even in person. I will detail a few examples here.

While no means the solitary method of infecting a computer,[1] *trojans* are aptly named as they bypass a system's security by tricking the owner into believing the payload is benevolent, beneficial, or required. Fake antivirus is an example of this behavior: a malicious website runs a fake virus scan[2] and reports the computer as infected. The trojan poses as a legitimate antivirus program for purchase and installation. A more nefarious version of this scheme is *ransomware*. In this scenario, for example, a website (either previously compromised or surreptitiously redirected to) displays a warning that the user has been caught breaking the law, and must pay a fine.[3]

Another common venue of attack is the *phishing* email. It is seen frequently on a personal and corporate level. At home, the email may masquerade as an official document from say, Paypal. At the business, phishing emails may take the form of a request from the local IT department. There are two important aspects to making deceit successful: imitation of the legitimate source (often achieved by pasting corporate logos within the email[4]), and masking URLs (so it appears legitimate, but actually redirects to a fake website[5] to collect credentials).

Viruses and other forms of malware bypass human interaction altogether, taking advantage of software bugs, or holes in security, relying on people that neglect to update their systems. This type of attack can be completely silent, never drawing attention to the fact the system is infected, as to keep the victim from removing the malware. When this happens, it is an indication that the malware is communicating

[1] http://www.veracode.com/blog/2012/10/common-malware-types-cybersecurity-101/
[2] Screenshot: https://docs.google.com/file/d/0B1coFwtPhG3XVG53anVjOVFtZ3c/edit?usp=sharing
[3] Screenshot: https://docs.google.com/file/d/0B1coFwtPhG3XQWJOXzJ6SmRsc1k/edit?usp=sharing
[4] Screenshot: https://docs.google.com/file/d/0B1coFwtPhG3XUDRjWFl1ck10VjQ/edit?usp=sharing
[5] Screenshot: https://docs.google.com/file/d/0B1coFwtPhG3XcXgyVDJKVFpDbTQ/edit?usp=sharing

with other infected computers to coordinate an action, such as attacking a server to bring down a website.

In the *Microsoft Tech Support Scam*, the criminal poses as a Microsoft technician and contacts people over the phone. The victim is shown a relatively accessible Windows feature, the *Event Log* (which logs computer activity) and told that completely innocuous messages are in fact proof of infection. Naturally, a small fee is required to clean the computer, or renew a fictitious warranty. That fee is usually kept inline with legitimate software costs, as a way to further legitimize the scam. Grammatically poor English (such as “You have 100 hacking files on your computer, you are very high risk"[6]) may have been a tip off in the past, but in the global economy, we have become accustomed to language barriers in the tech support field.

In email variants, such as *The Nigerian Scam*, the person is promised enormous returns on a small investment. The details differ and continually evolve, but the basic premise remains the same: the victim is lead to believe he or she has inherited money from a long-lost relative, or a wealthy foreigner needs help moving funds out of the country. The scam is so named due to the laughable (and thus memorable) English grammar of the scam emails,[7] often originating from Nigeria. However, this hides the much more sinister motive of filtering out all but only the most gullible of individuals, ensuring success.

Incompetence and Indifference

Proper security is a difficult and expensive process to implement. Some website developers will mean well but forget a crucial step, others may be underpaid, or simply not have time for completeness. A popular phrase “security by obscurity” is given to situations where weaknesses and poor practices are simply hidden, or not easily encountered. This will never stop a malicious hacker—they will leverage every bug, every cut-corner, and every oversight to gain access.

The Secure Socket Layer (SSL) protocol is the basis for security on the web. It provides a secure communication channel between the server, and the person visiting the website. When visiting a website, the “https” in front of a URL denotes HTTP[8] running over SSL. “...HTTPS gives us assurance of identity (we know who we’re connecting to), it ensures data integrity (we know the content hasn’t been modified) and finally, it gives us privacy (the data is encrypted and can’t be read by others).” [15]

[6] http://www.wired.co.uk/news/archive/2013-04/11/malwarebytes
[7] Screenshot: https://docs.google.com/file/d/0B1coFwtPhG3Xd1FSMzloT0ZOMkU/edit?usp=sharing
[8] Hypertext Transfer Protocol: the language browsers use to talk to servers

When transmitted over HTTPS, packets of information can only be read by both sides—if a third party obtained a copy of the data, they would be unable to decrypt its contents. In order for HTTPS to be effective, the entire page must be encrypted. If elements of the page are loaded over plain HTTP, the entire page is rendered vulnerable. Modern browsers are able to detect this and warn users appropriately.[9] Cookies are a standard method of storing a visitor's authentication status, but can be compromised if handled inappropriately. For example, cookies holding personal data should never be accessed outside of HTTPS. [15]

When a website employs SSL, the server gives the browser a digital certificate (which is a public key, discussed later) so they can communicate securely. These certificates are validated by a *Certificate Authority*; all browsers keep a list of CA's for this express purpose. If a website does not submit their certificate to a CA, or there is a mismatch, the person opening the website is presented with the certificate so she may manually accept or decline it.[10] If criminals successfully submit a fake certificate to a CA, their malicious website can masquerade as the legitimate website. [11]

Breaches and Non-Disclosures

In the last five years, between 2009 and 2013, counting only high-profile cases, there has been a little over 240 million stolen passwords lifted from servers with weak or no security.[12] [1] Approximately 55% of those were stored completely unencrypted—when the accounts were compromised the passwords were immediately available to the attackers. For those that were encrypted, password cracking is quickly becoming non-trivial. Once usernames and passwords are paired, criminals can test the credentials on other websites, such as bank accounts. It's an effective tactic because people tend to reuse the same password on about four different websites, on average. [1] This is to say nothing about the personal and identifiable data (social security numbers, addresses, phone numbers, etc.) being stolen along with the credentials.

This scenario is an immediate threat to a person's well being and livelihood. Should personal information become compromised, it's important to know immediately so appropriate action can be taken, for example, credit monitoring, updating passwords, and alerting banks. But companies may not be inclined to acknowledge

[9] Screenshot: https://docs.google.com/file/d/0B1coFwtPhG3XZ3R2aHFRWmZlMTg/edit?usp=sharing
[10] Screenshot: https://docs.google.com/file/d/0B1coFwtPhG3XSmpOci1ES1dCa3c/edit?usp=sharing
[11] http://www.theregister.co.uk/2013/01/04/turkish_fake_google_site_certificate/
[12] https://docs.google.com/spreadsheet/ccc?key=0AlcoFwtPhG3XdHo1a2J5cjAteGVlbmh6SlNTa3dqMGc&usp=sharing

the theft: bad PR, loss of customers, payments to customers for damages, and time and money to fix the original security issue, are deterrents to publicizing the theft. Thankfully, 46 states now have laws that force disclosure if personal data is compromised. Only Alabama, Kentucky, New Mexico, and South Dakota lack such regulation. [16]

Online Insecurity

Passwords are the most common facet of security that we have control over. They are the beginning and end of our online lives, yet most people pay them no mind. Since the dawn of the web, password rules have been difficult, limiting, and confusing. That has forced us into poor habits when making and storing the most important line of defense we have against online criminals.

Basic Password Protection: Hashing

Storing passwords in plain text (i.e. human readable, without encryption) is the poorest of practices. If security is breached in any manner, passwords have become less than useless—they are a liability. The solution is a one-way hash. This is the process of taking variable length data (e.g. a password), and storing it as fixed length data. It is the basis for modern cryptography.

A one-way hash has two properties: the process should never be reversible (the password cannot be decrypted), and two different passwords will never have the same resulting hash. [9] If either or both condition can be proven true, that hashing process (algorithm) is broken. In the real-world, a website's "forgot my password" function should create a temporary password, and then request a new one be created. If the original password is known and sent, passwords are not being encrypted properly, or at all.

One of the original one-way hash functions was the *Message-Digest* algorithm, which includes the popular MD5 (version 5). It was eventually replaced with the *Secure Hash* algorithm. There have been several iterations in this family as well, becoming incrementally stronger: SHA-0, SHA-1, SHA-2 (made up of SHA256 and SHA512), and most recently, SHA-3. Both of these algorithms were designed to calculate hashes quickly, and with low overhead. [12] Processes faster than *brute force* methods have developed to find collisions (duplicate hashes) within Message-Digest, SHA-0, and SHA-1, so they are now considered "broken". [9]

Passwords Are Broken

The brute force method of cracking passwords is the process of iteratively hashing all password combinations until a match is found within a list of passwords hashes (i.e. a stolen list). However, the difficulty of this is measured in exponential scale; it becomes computationally prohibitive after passwords reach five characters (depending on hardware), and after eight characters, even the most powerful computers hit the proverbial wall. [1] Resourceful hackers were able to overcome the brute force limitation by using two different tools: *rainbow tables* and *dictionary attacks*.

Despite its cheery name, rainbow tables include pre-calculated hash data, eliminating the need to calculate the hash of every possible password, and ultimately decreasing the amount of time and computing resources it takes to crack passwords. [8] Testing for dictionary words also proved to be effective at finding passwords. In 2009, the breach[13] of RockYou.com saw 32 million passwords leaked to the internet. From that list, the dictionary words *Password* and *princess* were among the top ten passwords, appearing approximately 62 thousand and 35 thousand times, respectively. RockYou.com only required five-character passwords and stored them in plain text. [7]

This negligence is compounded with the the *"first-day" password problem*, which is the practice of using a default password (such as *changeme*) for all new accounts. [5] The danger lies in that some people will never change this password, exposing not only their own account, but others as well. It is also an example of a security hole that is completely preventable by the administrators, and goes to back to the ignorance or incompetence problem.

In response to these threats, the *salted password* was introduced. The process is relatively simple, in that a string of unique characters is added to the password before it's run through a hashing function. It was successful in rendering rainbow tables useless, and slowing down dictionary and brute force attacks, because password hashes once again had to be calculated and compared one at a time. [1] But salt was hardly a cure-all for password woes.

To authenticate a user, the salt is added to the provided password to get the final hash, thus the salt itself can never be encrypted. If the hacker has a list of the salts and hashes, it's just a matter of incorporating the known salt into the brute force and dictionary attack algorithms. Unfortunately, salts are usually found stored alongside the hashes. [12] Ultimately, they will only slow down the cracking process by a multiple of the number of unique salts in a given list. To be effective, the salts must follow the same rules as user passwords: randomized, of a certain length, and never reused. [1] Naturally, this requires additional complexity and resources to implement.

For a long time, the best way to beat the password crackers was to follow a strict ruleset for creating strong passwords: 8 characters, mixed case letters, numbers, and

[13] Via a SQL injection (manipulating a SQL database by injecting malicious code into the website).

symbols. Combined with salt, it was near impossible to break. But an unlikely piece of hardware has invalidated all of the old theories.

Putting the Video Card to Work

In the past, testing password hashes was a function of the computer's *central processing unit*, or CPU. As graphics became more integral to the computer industry, those processes were moved to a dedicated chip on the video card called the *graphics processing unit*, or GPU. The GPU renders polygons: two dimensional figures that when combined, make up a three dimensional image. Better 3D images are made up of more numerous, smaller polygons, a result of having hundreds of "cores" calculating polygons in parallel. Without the overhead of managing the rest of the system, the GPU was perfectly suited to the task of dedicated password cracker.

The AMD Radeon 7970 graphics card is aimed at the consumer-level hardware enthusiast, selling for only a few hundred dollars. It is capable of calculating 4.8 billion MD5 hashes per second, or 2.2 billion SHA1 hashes per second. And those limitations exist only because the hash lists simply can not be fed to the GPU any quicker. [1, 12]

For passwords that can not be cracked with the classical methods described above, attackers have built machines with 8 to 25 GPU's for an increased number of cores. A 25 GPU cluster can churn out 180 billion MD5 hashes per second, 63 billion SHA1 hashes, and 350 billion NTLM hashes (Microsoft's old cryptographic algorithm, mercifully retired some years ago). [8, 17] Stacking so many GPU's generates a lot of heat, so much so that it can damage equipment. To keep these monsters from overheating, owners go so far as to submerge them in mineral oil. [3]

The advantages rainbow tables once had is diminished as it's now quicker to calculate hashes one at a time; salts are simply cracked through sheer brute force. Furthermore, holistic attack methods that were once too computationally complex are now feasible with the GPU. Password crackers stack multiple dictionary words (a combinator attack), and test for common expressions, phrases, or patterns. For example, the password *qeadzcwrsfxv1331* is easily broken because it follows a pattern on the keyboard. [1, 2]

Taking Back Security

The path to fixing security lies with not only the systems' designers, but the individual as well. Both sides are responsible for combating poor practices and bad habits.

Stronger Hashing Algorithms

As mentioned earlier, the Message-Digest and Secure Hash algorithms were designed to be calculated quickly, and with low overhead. This makes them attractive to websites that have high traffic and usage; large amounts of users can be authenticated with no discernable lag. But this efficiency has also been their undoing: the speed advantage is exploited by password crackers, as evidenced by the billions of hashes able to be created every second. Thus, the old algorithms were no longer suitable to cryptography. What was needed were more computationally expensive algorithms, designed specifically for encrypting passwords, rather than data integrity checking. [15]

Three algorithms have become popular for meeting the demands of modern security: *PBKDF2*, *bcrypt* and *scrypt*. Their commonality is a concept known as *work factor*. The work factor is a variable within the algorithm that allows for adjustments in its computational complexity. As the work factor is increased, the hashing will get slower, and the time an attacker must invest, per password, will increase. Since the work factor is embedded in the result, the hashes generated are backward and forward-compatible (new versions can read old hashes, and vise versa). [12]

The hurdle to overcome is the cost in implementing these functions. GPU's are used by attackers to crack passwords, so they could be used just as effectively for legitimate hashing. But servers (web, authentication, etc.) don't come with GPU hardware—graphics are typically the domain of a personal computer. Implementing a solution on existing servers will undoubtedly carry a non-trivial cost. It may be a cost that website administrators and businesses are unwilling to incur.

Properties of Strong Passwords

With password cracking being as efficient as it is, it may seem impossible to come up with a the perfect password. But it doesn't have to be perfect, just good enough that it would deter an attacker. Combined with proper habits, it's possible to keep your data safe and secure.

The strength of a password can be measured in bits of *entropy*. Entropy is a lack of order, or randomness, and that is the key to making a strong password. High entropy, lengthy passwords are resilient to dictionary, heuristic and brute force attacks. There are several properties of a strong password: it needs to be over 9 characters long (but ideally 13-20), mixed case letters with numbers and symbols, no words, phrases, or patterns, and no common symbol substitution. This ensures a password entropy of at least 50 bits. [17]

A 50 bit password will take 2^{50} (1 quadrillion) brute force attempts to try all possibilities (although on average it would be found in half that many attempts). But if a website is still employing MD5 or SHA1, that password will still be broken relatively quickly. It's predicted that passwords reaching 65 to 70 bits will require systems that cost over $1 million, for the next few years, regardless of the algorithm used. So the bar for minimally safe passwords has been set. [13]

Since entropy is by definition, randomness, it's impossible to define the exact number of bits of entropy a password has, but there are methods for approximating. Two examples include the liberal Rumkin[14] and the more conservative zxcvbn.[15] Both algorithms analyze the characteristics of a password; some of those methods are found in both, while others may be unique to one or the other. But these differences will result in different bits of measured entropy. To be safest, always assume the lower number result is correct.

Generating Secure Passwords

Within the security field, there are competing methods and ideas for creating high entropy passwords. Some are more classical, as stated just above, in that they call for chaotically generated passwords, using numbers, mixed case alphabet, and special characters. [2] Others advocate for simpler, longer passwords rather than shorter, complex passwords, for two reasons: there may not be enough good guessing data for longer passwords (due to the lack of long password requirements), and common long passwords are still less common that common short passwords. [14]

To help generate the former, Bruce Schneier developed a method which would make such a password more memorable. The concept is simple: select a relatively long sentence that you know or can easily memorize, and turn it into a password. For example, *the quick brown fox jumps over the lazy dog*, turns into TqbFoxjotlDog. Then it's further strengthened with numbers and symbols: *TqbF0x,jotlD0g!*.[16] The two number substitutions, and two punctuation marks are relatively straightforward, aiding in memorization, and works out to be about 70 bits of entropy. This method is captured in Schenier's password generator app, Password Safe.[17] [4]

To test the later theory, one study provided people with multiple rule sets for creating a strong password, one which required 16 characters, and a second that required only 8 characters, but also different cases, numbers, letters and symbols. Based on

[14] http://rumkin.com/tools/password/passchk.php
[15] https://dl.dropboxusercontent.com/u/209/zxcvbn/test/index.html
[16] This is an example, do not use it. The sentence is also inherently bad because it is so well-known.
[17] http://passwordsafe.sourceforge.net/

the results, people created higher entropy passwords under the first rule set (44.68 bits) versus the stricter second rule set (34.30 bits). [14] If in fact representative of a person's predilection toward generating passwords, the longer, less complex method would be ideal for two reasons: people are inherently poor randomness generators, and complex passwords tend to be written down as they are easily forgotten.

Randall Munroe, author of the comic *xkcd*,[18] is an advocate of the longer, simpler password generation method. He subsequently drew a comic to express how little effort it takes to improve password strength, and how the public has been instilled with poor security practices over the years. This comic has become known as the XKCD method of creating passwords. [6] It presents the idea of randomly selecting four distinct words, separated by spaces. Randall calculates the entropy at 44 bits, but as discussed earlier, the minimum should be at least 65 bits. The XKCD method still stands, but for proper security, the length should be five words. Both xkpasswd,[19] and Diceware,[20] are websites that can generate strong passwords in this vein.

Real World Constraints

The most difficult aspect of adopting secure practices is putting those practices into use. There are two roadblocks to fully realizing this goal: arbitrary restrictions on passwords, and memorization.

Many websites still impose an eight character limit, in the hope that people will create passwords with numbers, symbols, and mixed case, and still be memoizable. However, this is in fact an impediment to that very goal as eight letter password entropy is likely to be under 55 bits. Many sites also limit special characters (and potentially how many times they can be used) further weakening the resultant password.

The issue of individual password memorization was addressed, but trying to remember unique passwords for all sites is near impossible. A possible solution is selecting a strong base password, and coupling it with a personal algorithm for customizing and adding to it based on the website where it's used (e.g. adding the website URL somewhere in the password).

But to make unique, truly random, highest-possible entropy passwords for the quantity of websites that require accounts these days, a *password manager* is the best option.

[18] http://xkcd.com
[19] https://www.xkpasswd.net/c/index.cgi
[20] http://world.std.com/~reinhold/diceware.html

Best Practices

Managing Passwords

Password managers are safe alternatives to memorizing dozens of passwords. By encrypting the database of credentials with a *master password*, a person only need to memorize one single strong password. The methods presented earlier can work in this scenario as well, in particular Diceware or xkpasswd. AgileBits also offers guidelines for creating a good master passwords.[21] Another critical component includes the ability to generate strong passwords, aside from simply securing them, a feature that mitigates the insecure human element.

There are two popular options for password managers: 1Password[22] and LastPass.[23] (Both of which use the PBKDF2 algorithm for encrypting the master password.) [10] KeePass[24] introduces an additional layer of security with a master key, a required file for decrypting the database.

Two-Factor Authentication

Minimum website authentication consists of a username and a password, but this may not be enough when dealing with sensitive data, such as banking information. Take for example the *forgot my password* feature, which is meant to supply a new password should it be lost. A special URL is sent to the person's email, so when used, the website identifies the URL as belonging to that specific *email address*. If that email account is not accessible, the website support techs will often do password resets over the phone. This usually requires some personally identifiable information, such as answers to password-reset questions, partial credit card numbers, billing information, or even just a phone number. But this is all information that can be discovered about a person by digging through publicly available records such as Facebook, or looking at other online accounts.

If an attacker engages in social engineering to obtain personal information, he may be able to impersonate someone for the purposes of a password reset, thus gaining access to their sensitive accounts without actually ever stealing the password directly. A way of mitigating this loophole is by using an additional *one time password*. As the name suggests, this password works only once, becoming useless

[21] http://blog.agilebits.com/2011/06/21/toward-better-master-passwords/
[22] https://agilebits.com/onepassword
[23] https://lastpass.com/
[24] http://keepass.info/

after it's been used. For website authentication, the OTP takes the form of a PIN. There are three popular methods of distributing PIN's: small hardware tokens, SMS (text message) and smart phone app.

Once a person has authenticated to a website using his or her password, they are prompted to enter the PIN. In theory, only that person will have access to the PIN delivery mechanism, and can thus prove identity. If an attacker does manage to steal or reset a password, there is now a secondary layer of security protecting the person. There is added responsibility and complexity to this system, and convenience is lost, but for potentially critical data, the added security is worth the cost. Many websites have begun to roll out TFA, including Google, Facebook, Dropbox, Twitter, Yahoo, and others.

Other Practices

There are several practices not directly password related that can play a role in keeping online personas and data secure:

1. Always keep the computer up-to-date. Attackers take advantage of unpatched systems to exploit security holes and bugs. This is a way to steal the owner's personal information or infect the computer. Personal computers are often used as part of a malicious *botnet*, a group of infected networked computers which can communicate to accomplish tasks not possible with a single machine. It may even include cracking passwords.
2. A corollary of the above is to ensure the web browser is also updated.
3. In case of theft, the computer should be encrypted to prevent any personal data from being lifted. Without encryption, user account passwords are useless. Disk drives can be removed and plugged into another computer, or special operating systems can be loaded from USB drives (or CD/DVD) that run independently of the main system, and have complete access to the information stored on disk. Encryption is included with popular operating systems: BitLocker for Windows, FileVault for Mac, and LUKS for Linux.
4. If sent unsolicited advice, or warnings, remain suspicious and never install unknown software.
5. Disable the built-in browser password manager. These are not as secure as the ones previously discussed, and are meant to be more for convenience rather than security. [11]
6. Consider browser add-ons such as HTTPS Everywhere,[25] or DoNotTrackMe.[26] Modern browsers can alert you to suspicious websites, but to decrease the

[25] https://www.eff.org/https-everywhere
[26] https://www.abine.com/dntdetail.php

chances of visiting a compromised or malicious website, employ a service such as OpenDNS.[27]

Conclusion

We can never be 100% secure in an online world. Attackers are continually finding new venues of attack, processes to exploit, and reasons to steal. For the average person, good habits and common sense will go a long way in protecting oneself. Having technical knowledge of the problems faced in the security industry will give anyone the extra edge against would-be attackers looking only for the least common denominator. I urge everyone to share and pass along the practices outlined here as to make a safer web for us all.

[27] http://www.opendns.com/

[1] D. Goodin. (2013, May 27). Anatomy of a hack: How crackers ransack passwords like "qeadzcwrsfxv1331". *Ars Technica* [Online]. Available: http://arstechnica.com/security/2013/05/how-crackers-make-minced-meat-out-of-your-passwords

[2] A. Rao, B. Jha, and G Kini. "Effect of Grammar on Security of Long Passwords," School of Comp. Sci., Carnegie Mellon Univ. Pittsburgh, PA, May 2012. Available: http://reports-archive.adm.cs.cmu.edu/anon/isr2012/CMU-ISR-12-113.pdf

[3] D. Goodin. (2013, June 4). Password crackers go green by immersing their GPUs in mineral oil. *Ars Technica* [Online]. Available: http://arstechnica.com/security/2013/06/password-crackers-go-green-by-immersing-their-gpus-in-mineral-oil/

[4] B. Schneier. (2008, Nov. 13). Passwords Are Not Broken, but How We Choose them Sure Is. *Schneier on Security* [Online]. Available: http://www.schneier.com/essay-246.html

[5] B. Schneier. (2012, Sept. 19). Recent Developments in Password Cracking. *Schneier on Security* [Online]. Available: http://www.schneier.com/blog/archives/2012/09/recent_developm_1.html

[6] R. Munroe. (2011, August 10). Password Strength. *xkcd* [Online]. Available: http://xkcd.com/936/

[7] Imperva, CA. Consumer Password Worst Practices. presented at http://www.imperva.com/ [Online]. Available: http://www.imperva.com/docs/wp_consumer_password_worst_practices.pdf

[8] D. Goodin. (2012, Aug. 20). Why passwords have never been weaker—and crackers have never been stronger. *Ars Technica* [Online]. Available: http://arstechnica.com/security/2012/08/passwords-under-assault/

[9] B. Schneier. (2004, Aug. 19). Cryptanalysis of MD5 and SHA: Time for a New Standard. *Schneier on Security* [Online]. Available: http://www.schneier.com/essay-074.html

[10] K. Young. (2013, March 19). Two-factor authentication? *AgileBits* [Online]. Available: http://discussions.agilebits.com/discussion/12407/two-factor-authentication

[11] Anonymous. (2013, June 20). How Browsers Store Your Passwords (and Why You Shouldn't Let Them). *RaiderSec* [Online]. Available: http://raidersec.blogspot.com/2013/06/how-browsers-store-your-passwords-and.html

[12] T. Hunt. (2012, June 26). Our password hashing has no clothes. *troyhunt.com* [Online]. Available: http://www.troyhunt.com/2012/06/our-password-hashing-has-no-clothes.html

[13] S. Rachev. (2013, March 8). Your Password Is No Longer Secret, Part 1. *Java Code Geeks* [Online]. Available: http://www.javacodegeeks.com/2013/03/your-password-is-no-longer-secret-part-1.html

[14] S. Komanduri, R. Shay, P. Kelley, M. Mazurek, L. Bauer, N. Christin, L. Cranor, and S. Egelman. Electrical and Computer Engineering., Carnegie Mellon Univ. Pittsburgh, PA, May 2011.

Available: http://www.ece.cmu.edu/~lbauer/papers/2011/chi2011-passwords.pdf

[15] T. Hunt. (2013, May 16). Hack yourself first – how to go on the offence before online attackers do. *troyhunt.com* [Online]. Available: http://www.troyhunt.com/2013/05/hack-yourself-first-how-to-go-on.html

[16] *State Security Breach Notification Laws*, National Conference of State Legislatures, Denver, CO, 2012. Available: http://www.ncsl.org/issues-research/telecom/security-breach-notification-laws.aspx

[17] D. Goodin. (2012, Dec. 9). 25-GPU cluster cracks every standard Windows password in <6 hours. *Ars Technica* [Online]. Available: http://arstechnica.com/security/2012/12/25-gpu-cluster-cracks-every-standard-windows-password-in-6-hours/