Half a Century of Practice: Who is Still Storing Plaintext Passwords?

Erick Bauman, Yafeng Lu, Zhiqiang Lin

Department of Computer Science, The University of Texas at Dallas
800 W. Campbell RD, Richardson, TX 75080 USA
{firstname.lastname}@utdallas.edu

Abstract. Text-based passwords are probably the most common way to authenticate a user on the Internet today. To implement a password system, it is critical to ensure the confidentiality of the stored password—if an attacker obtains a password, they get full access to that account. However, in the past several years, we have witnessed several major password leakages in which all the passwords were stored in plaintext. Considering the severity of these security breaches, we believe that the website owners should have upgraded their systems to store password hashes. Unfortunately, there are still many websites that store plaintext passwords. Given the persistence of such bad practice, it is crucial to raise public awareness about this issue, find these websites, and shed light on best practices. As such, in this paper, we systematically analyze websites in both industry and academia and check whether they are still storing plaintext passwords (or used to do so). In industry, we find 11 such websites in Alexa's top 500 websites list. Also, we find this is a universal problem, regardless of the profile of the websites according to our analysis of almost 3,000 analyzed sites. Interestingly, we also find that even though end users have reported websites that are storing plaintext passwords, significant amounts of website owners ignore this. On the academic side, our analysis of 135 conference submission sites shows that the majority of them are also still storing plaintext passwords despite the existence of patches that fix this problem.

1 Introduction

To access sensitive information (e.g., online financial accounts, social networks, and email services) protected by a computer system, end users often need to provide a password, which is a secret string of characters that is matched with a stored value in a database to authenticate a user. While password based schemes are disliked by users [16] and have many alternatives [18], they are still the de facto standard for authentication, especially in today's Internet, due to their easier deployment (no need of special hardware), low cost, and simplicity.

However, implementing a secure password system is still complicated, and many things can go wrong. Among these password threats, some of them can be fixed easily. For instance, we should always store passwords as hashes instead of as plaintext. Note that the systems and security field began discussing password secrecy half a century ago [32,37], and it was decided early on that using hashes provided needed security and storing plaintext passwords was unwise. As a statement of fact, a website should

never store plaintext passwords, but this statement can be made even stronger by adding that sites should also not store passwords with reversible encryption. While encrypting passwords is stronger than nothing, the key in many cases may be on the same server that is compromised, thereby defeating its purpose. Therefore, regardless of the way a password is being stored on the server, the salient fact is that for proper security a website should never be able to retrieve a user's original password. This should preclude the fact that under no circumstances should a user's password be sent to them in plain text.

Unfortunately, in practice, website owners ignore these facts and continue to store and send passwords insecurely. Regardless of how the passwords are being stored on the server side, if a server sends a user's plaintext password back to them, it is following bad password practices. While some sites may consider sending a user's password to them a matter of convenience, we define a site following such practices as "insecure" regardless of motivations. Note that insecure here means "subject to password leakage".

In the past several years, we have witnessed a number of major password leakage incidents. More specifically, as illustrated in Table 1, we can notice that many of the leaked passwords were actually in plaintext, and in total more than 100,000,000 plaintext passwords have been leaked. Given that end users often tend to reuse their passwords for different websites [28,24,21], such leakage forces users to change their passwords or risk someone breaking into their accounts.

Site Address	Amount	Year	Plaintext?	Source
myspace.com	34,000	2006	✔	[25]
rockyou.com	32,603,388	2009	✔	[39]
hotmail.com	8,931	2009	✔	[20]
gawker.com	1,300,000	2010	✗	[35]
csdn.net	6,428,632	2011	✔	[9]
tianya.cn	31,761,424	2011	✔	[9]
webo.com	4,765,895	2011	✔	[9]
7k7k.com	6,541,991	2011	✔	[9]
renren.com	4,768,600	2011	✔	[9]
17173.com	18,333,776	2011	✔	[9]
duowan.com	8,305,005	2011	✔	[9]
uuu9.com	5,577,553	2011	✔	[9]
ieee.org	100,000	2011	✔	[10]
rootkit.com	81,4501	2011	✗	[11]
youporn.com	1,566,156	2012	✔	[15]
voices.yahoo.com	453,000	2012	✔	[14]
militarysingles.com	163,792	2012	✗	[13]
linkedin.com	6,500,000	2012	✗	[12]
adobe.com	150,000,000	2013	✗	[43]

Table 1. Notable Examples of Recent Password Leakages.

Do website owners learn lessons from the mistakes of others and quickly fix their websites in response to reports of millions of leaked plaintext passwords? Table 1 shows that is clearly not always the case. For instance, the massive password leakage of rockyou.com in 2009 [39] resulted in substantial media coverage, but we still observe that over 70 million plaintext passwords were stolen in 2011 [9], which is a historic record. While the most recent password leakage was not plaintext (the leaked passwords from Adobe were encrypted), we can still find websites that store plaintext passwords. This is in fact true in both academia and industry. Specifically, we find many academic conference paper submission sites (especially those using HotCRP [29]) that store plaintext passwords, and we also find several websites from Alexa's top 500 websites list that store plaintext passwords as well.

Therefore, given the existence of such a bad practice, it is crucial to raise public awareness, find these websites, and shed light on best practices. As such, in this paper, we systematically analyze websites in both industry and academia, investigate whether they are insecure (or used to be insecure), and discuss why they are still insecure. In industry, we find 11 insecure websites in Alexa's top 500 websites list. Interestingly, we also find that even though end users have reported that a website is insecure, significant

numbers of website owners appear to ignore this. On the academic side, we examine 135 academic conferences in security, systems, and networking from the past five years. Our results reveal that many conference paper submission sites remain insecure and store plaintext passwords, in spite of the existence of patches to fix this problem.

2 Background

Password security has been an important security issue for decades. Users trust service providers with valuable information that can be reached by anyone with the correct username and password; maintaining password secrecy is paramount. While the use of password hashing can be traced back to Multics and UNIX in the 1960s and 1970s[1], we focus on the web. We do not discuss password transmission, although we note that the client-server connection should be encrypted at the very least. We also do not discuss alternatives to password authentication, because we want to focus on the most popular authentication mechanism.

2.1 How to Store a Password

There are many ways to store a password. At a high level, there are three basic ways: (1) plaintext, (2) encrypted plaintext, (3) hashed plaintext. In the following, we review these approaches and discuss their pros and cons.

Plaintext. The most straightforward approach to implement a password system is to store the user's password in plaintext and perform a simple comparison to authenticate a user. While this approach is easy to implement, it creates a lot of security problems.

First, anyone with access to the database can view all users' passwords. Given that many Internet users tend to reuse their passwords [28,24,21], it may allow malicious owners of a site to login to user accounts on other sites. Second, even if we trust the website owners, if an attacker gains database access, all passwords will be leaked. This also makes password collection trivial, and is the reason that millions of plaintext passwords have been revealed.

The advantages of plaintext are appealing to an inexperienced developer. It is simple authentication without the need to use cryptography, and it gives an illusion of security. The disadvantages are obvious and serious enough that plaintext should never be used.

Encrypted Plaintext. A natural solution is to encrypt the user's password. One can either directly encrypt the password with a single key, or encrypt a constant with a user supplied password as the key.

For the first approach, the server must store the key so it can encrypt the password. Therefore, passwords are at risk of being revealed if the key is compromised. While one can separate the key from the data by putting the key in the code and the passwords in the database, this only slightly reduces the attack vectors. Also, if the key is ever revealed, all passwords encrypted with the key are compromised, and the owners will know the

[1] An incident on the CTSS system in the mid-1960s, in which the contents of the password file were displayed on login, was inspiration for password hashing [32]. Such an algorithm was in use in Multics since at least the early 1970s [37] and in UNIX since Version 3 in 1973 [31].

passwords since they have the key. Finally, an attacker with access to only ciphertext can still tell if users have the same password.

The second approach operates in a similar manner to hash functions. It produces a fixed-length ciphertext. It is better than the first approach but can reveal identical passwords like the first approach does. The strength of the password will rely on the encryption, which was not designed specifically for such use.

Hashed Plaintext. One-way cryptographic hash functions exist for the specific purpose of authentication. There is no need to decrypt a password; the server can compare the final hash with the value in its database. By default, we should use the hash function to transform user provided passwords and then store the hashes. Security then depends on which hash function is used. Some well-known hashing algorithms are MD5, SHA-1, SHA-2, and SHA-3. MD5 and SHA-1 are no longer recommended for cryptographic purposes. These functions can produce hashes relatively quickly.

Using a cryptographic hash function (with salts) has been proven to be the right approach for storing passwords, but it still remains susceptible to certain attacks (§2.2). The use of salts makes precomputing passwords for rainbow table attacks impractical, slows dictionary attacks, and hides the presence of identical passwords, and therefore it notably improves the process. Also, hashing costs can be increased with algorithms such as PBKDF2 or bcrypt.

2.2 Attacks Against a Stored Password

Large password databases are tempting targets to hackers and crackers, and therefore server administrators must take care to secure their servers as best they can. Here are some ways that attackers can obtain and crack passwords:

Password Stealing. The first objective of an attacker is usually to gain access to the server or to user accounts. One client-side strategy that is near-impossible for a site to defend from is phishing attacks, in which an attacker sets up a fake version of the real site to fool users[2]. Other client-side attacks include keyloggers or other malware that steals a user's password.

An issue of greater concern to the owners of a site is maintaining the security of their database of passwords and user data. One significant attack vector for sites is SQL injection, which exploits mistakes made in sanitizing user data before passing it to the database. Other attacks include obtaining server dumps (as an inside job) or convincing the server to divulge its memory contents, which is what happened with the Heartbleed SSL vulnerability.

Password Cracking. If passwords are stored in plaintext, then once an attacker retrieves the contents of the database their job is done. Hashing forces an attacker to perform extremely difficult calculations in order to retrieve usable passwords, and in theory the calculations require a high enough time complexity to render them infeasible. Unfortunately, there are several strategies attackers can take to make cracking the hashes easier.

[2] While a site can hope that its SSL certificate will help a user distinguish an untrusted impostor from the real site, if a user is fooled, the real site may have no way of knowing the attack even happened.

Hashing algorithms are one-way functions; they are easy to calculate one way and extremely hard to calculate the other way. Therefore, the most straightforward attack on hashes is to simply attempt to compute hashes for every possible password by brute force instead of trying to derive passwords from the hashes, which is nearly impossible [32]. The disadvantage to this attack is that the password space is enormous and slower hashing algorithms make computing many hashes a very time-consuming process.

Instead of choosing every possible password combination, it is much faster if an attacker instead selects candidate passwords from a dictionary or database of common passwords or phrases. Since many users choose weak passwords, it is much more likely that an attacker will find a valid password sooner in their search than with simple brute forcing. In addition, attackers can make changes to the dictionary words to reflect common modifications that users perform on their passwords to make them more "secure." Again, one way to try to defend against this is by making it computationally expensive to calculate a large number of hashes, but an attacker would likely still be able to obtain the weakest passwords despite this.

3 Practice in Industry

Since storing plaintext passwords is a very bad practice, it is important to make sure website owners properly store users' passwords. However, from a client perspective, we cannot tell how the passwords are being stored in the server's database. However, there is a straightforward method to determine if the website is able to retrieve a user's original password. In particular, we can use the "forgot password" option, in which website owners mail the user a password reset link, a new (random) password, or the user's previous password. If it is the user's previous password, then the server must be storing user passwords in plaintext or reversible encryption with the server having access to the decryption key, and they are sending the plaintext password via an unencrypted channel (email). We therefore consider websites that send a user's password back to them as insecure.

Experiment Setup. Therefore, to determine whether a particular website is subject to password leakage or not, we must first register a user and then trigger the password mailing option. While we wish we could write a robot to automatically scan these websites and perform a large scale study, captchas and special verification such as using SMS require manual inspection, which limits the number of sites that can be collected.

Fortunately, we noticed that there is a website dedicated to reporting websites of this type called PLAINTEXTOFFENDERS [5], which allows Internet users to submit examples of websites that store user passwords. Therefore, those submitted websites could serve as first hand raw data and enable us to extract insights from industry practice. However, these insecure sites are submitted in an ad-hoc manner, and therefore the archive may be missing certain important websites. Given the fact that it is not feasible to register to a large number of sites, but it is still desirable to obtain a large sample, we decided to inspect the PLAINTEXTOFFENDERS archive first (§3.1), and then manually inspect the top 500 websites (§3.2). In the following, we present our findings.

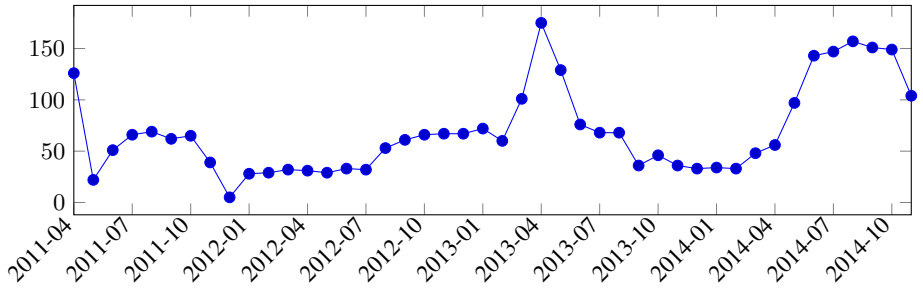

Fig. 1. Plain Text Offenders website reporting rate by year and month. Some posts had multiple URLs. The data shows no downward trend in the number of websites being reported.

3.1 Results from PLAINTEXTOFFENDERS

PLAINTEXTOFFENDERS [5] is a website that allows users to submit examples of websites that store user passwords. It has a substantial archive of sites with submissions that contain a URL for each insecure site, and an image of an email containing the user's password, demonstrating that the site is storing user passwords without hashing them. However, at the time of our original study, PLAINTEXTOFFENDERS did not provide any aggregated statistics or analysis of its submissions. Therefore, we developed a script to automatically scrape the entire site contents. We perform four types of studies that aim to understand (1) the trend of industry practice, (2) site rankings, (3) site classification, and (4) the properties of reformed sites.

The Trend. We first wanted to see whether industry practice has been improving. Since PLAINTEXTOFFENDERS's archives go back to 2011, we decided to obtain information on the reporting rate. We retrieved the date the site was posted, the URL, and the description of the site from PLAINTEXTOFFENDERS. In total, the site yielded 2,914 URLs. We calculated the number of websites reported per month over the lifetime of PLAINTEXTOFFENDERS; the results are shown in Figure 1. The number of submissions to the site shows no clear indication of a trend. Ideally, the graph would show a downward trend to indicate less websites storing plain text passwords. However, due to external factors such as increases in the number of global websites and potential increasing popularity of PLAINTEXTOFFENDERS, it is difficult to accurately determine the change in percentage of such websites on the Internet.

Ranking	Site Count
1-500	41
501-1000	41
1001-10000	297
10001-100000	747
100001-1000000	989
1000001-10000000	491
>10000000	88
UNKNOWN	224

Table 2. PLAINTEXTOFFENDERS websites grouped by their Alexa rankings. The majority of the sites are in the top million sites.

Site Ranking. It is important to determine how popular the insecure sites are, because the problem is more significant if popular, high-traffic sites are also insecure. Therefore, we decided to rank the sites by using Alexa to determine the popularity of the listed insecure sites. By retrieving their Alexa rankings, we could group them to show the distribution of the sites by number of users. We show this result in Table 2. The UNKNOWN category

consists of sites that did not rank in Alexa's system. Note that ranks in Alexa's ranking system are not considered statistically meaningful beyond 100,000, but there is still a substantial number of them in that range—over 1,000.

Observation 1 *Over 350 sites are in the top 10,000, 82 sites in the top 1,000, and 41 in the top 500. Some of the world's most popular sites are showing passwords in plaintext.*

It is important to note that Alexa rankings change frequently. While the rankings listed were accurate at the time of retrieval, the distribution will change over time, even if the underlying list of sites stays the same.

Site Classification. We then ran each URL through uClassify [8], a free API for text classification. We developed a script allow it to analyze the contents of the home page of each site. The contents were classified into one of thirty business categories. The UNKNOWN category consists of sites that did not respond when uClassify attempted to query them. The top categories are shown in Table 3. There are important sectors dealing with potentially sensitive data, and PLAINTEXTOFFENDERS offers hundreds of concrete examples of websites in these sectors. There are URLs for many online retailers, over ten banks, and sites dealing with tax forms, which shows that the risk to consumers in these sectors is real.

Category	Site Count
Information Technology	494
UNKNOWN	282
Marketing and Advertising	281
Retail Trade	227
Telecommunications	192
Investing	157
Hospitality	147
Accounting	142
Consumer Goods and Services	122
Arts and Entertainment	107
Opportunities	97
Cooperatives	69
Food and Related Products	67
Business Services	65
International Business and Trade	65
Environment	62

Table 3. PLAINTEXTOFFENDERS websites categorized by uClassify business categories, sorted by greatest to least. The bottom 15 categories are omitted.

Observation 2 *Password plaintext storage is common in industries working with sensitive data.*

Reformed Sites. One important consideration is whether sites that once stored passwords insecurely have improved. If they have, then this indicates growing awareness and willingness of site owners to change and improve their sites.

The sites that were reported on PLAINTEXTOFFENDERS in 2011 have had three years to improve their password storage systems. We decided that this set of sites was a good candidate for investigating to see if they have stopped storing plaintext passwords. We performed a manual investigation of the sites reported on PLAINTEXTOFFENDERS in 2011 to determine if they had fixed their password systems, and in total there are 483 sites. Among them, 293 (i.e., 60.7%) could be tested. There are several reasons why the sites could not be tested: some were no longer online, and some needed a phone number or other special information that could not be provided. Of the 293 sites that were tested, 205 had fixed the problem and 88 remain insecure, meaning that approximately 70% of the testable sites no longer store passwords in plain text.

PLAINTEXTOFFENDERS has a list of "Reformed Offenders," which consists of sites that have been confirmed to no longer send passwords in plaintext. Surprisingly, the list is extremely short; people may be less inclined to verify existing sites in the database than to add new ones. As indicated by our results, there may be a significant number of

sites reported on PLAINTEXTOFFENDERS that are no longer insecure, and so the actual number of still-insecure sites may not be as serious as it first appears. We will submit the list of reformed sites that we found to PLAINTEXTOFFENDERS after publication.

We took the 205 fixed sites and grouped them by category like we did for the entire set of PLAINTEXTOFFENDERS sites. This way, we could compare the demographics of the reformed sites and the entirety of the PLAINTEXTOFFENDERS archives to determine if there are any specific kinds of sites that improved. . Interestingly, the sites fell into similar category groupings. While the order varied slightly, these sites had very similar distributions to the overall distribution of the PLAINTEXTOFFENDERS archives.

There was no specific type of site that indicated that the site would be more or less likely to be fixed. Less new sites may adhere to best practices than long-established sites, but many websites do eventually stop storing passwords in plaintext. Therefore, we can conclude that

Insight 1 *There is no specific website profile that can definitively indicate the probability of whether it is handling user passwords correctly, or its likelihood to fix the problem if it is not following best practices. However, a website is more likely to have fixed the problem the longer they have existed.*

This result corresponds well with previous studies; it has been found that longer-lived sites tend to follow better security practices [19].

3.2 Result of the Manual Analysis of Top 500

As discussed earlier, insecure sites are submitted to PLAINTEXTOFFENDERS in an ad-hoc manner, as the existence of a submission depends on a user knowing about PLAINTEXTOFFENDERS, using the insecure site, and choosing to report it. To have a better view, we decided to perform a manual analysis systematically on a given number of sites. We chose Alexa's top 500 sites as input, and manually analyzed whether there are any websites that still store plaintext passwords. Recall earlier we found 41 insecure sites on PLAINTEXTOFFENDERS in Alexa's top 500 sites (Observation 1), and this study would also allow us to confirm how many of these top 500 sites have fixed the problem.

We used the manual methodology discussed earlier to determine whether a given website is insecure by registering a user and checking the site's response to the "forgot password" option. For the current top 500 sites, we were able to manually inspect 393 (i.e., 78.6%) of them (the rest cannot be verified due to constraints such as requiring special verification, a special account, special utilities or services, a referee, an existing account number,

Site Address	Rank.	Category	Country
fc2.com	58	Business Services	Japan
lists.wikimedia.org	135	Cooperatives	United States
badoo.com	164	Cooperatives	Italy
espncricinfo.com	188	Arts & Entertainment	India
liveinternet.ru	192	Arts & Entertainment	Russia
rutracker.org	301	Arts & Entertainment	Russia
corriere.it	380	Arts & Entertainment	Italy
extratorrent.cc	415	Arts & Entertainment	India
jrj.com.cn	456	Investing	China
kooora.com	464	Arts & Entertainment	SaudiArabia
xywy.com	484	Healthcare	China

Table 4. The sites from our study of the top 500. None of these have been fixed yet. The countries were determined by Alexa based on popularity.

or a safe code). Among them, we found at least 11 reported insecure sites have not been patched yet. Among the remaining 30 top 500 sites from PLAINTEXTOFFENDERS , they could either not be verified due to requiring personal information, or they were fixed or incorrectly reported. The sites found on PLAINTEXTOFFENDERS and manually verified by us are presented in Table 4.

Observation 3 *There are likely insecure sites that have not been reported, and there may be several more in the top 500 that we were unable to verify.*

However, the number we were able to confirm is a small percentage (2.2%) of the top 500 sites, and even the total number reported on PLAINTEXTOFFENDERS (8.2%) is not nearly as serious as it could be. This matches the pattern found in previous studies that more popular sites tend to have better password security [19].

4 Practice in Academia

We have observed the mistakes made by industry. Does academia perform better? Since academic researchers often need to set up conference websites to manage paper submissions and reviews, we investigated how conference submission sites manage user passwords. In this section, we present our findings regarding this.

Software Name	Centralized	Checked		Plaintext	
		#	%	#	%
HotCRP (HC)	✗	106	78.5	103	97.2
EasyChair (EC)	✔	19	14.1	7	36.8
SoftConf (SC)	✔	4	3.0	3	75.0
EDAS (ED)	✔	5	3.7	3	60.0
CMT (CM)	✔	1	0.7	0	0

Table 5. Conference Management Software Inspected in Our Study.

4.1 Experiment Setup

Conference paper submission and review is an important activity for academic researchers, especially in computer science. During a paper submission, an author/reviewer often needs to register on a submission site and choose a username (or email) and password for authentication. There are a number of frequently used examples of conference management software, such as HotCRP [29], EasyChair [1], Softconf [7], EDAS [2], and CMT [4], and they are hosted on web servers to provide this service.

As presented in Table 5, some of the conference management software is managed through a centralized service (e.g., EasyChair), some of them are hosted independently by individual conference submission sites (e.g., HotCRP), and some of them are mixed, with both centralized hosting and individual hosting (e.g., SoftConf). Centralized services host the conference software themselves and perform all setup and maintenance for a conference. However, many conference organizers choose to host their own paper submission sites, which gives them more control, but also puts the responsibility of patching and updating the conference software on each individual conference.

In our study, we selected these five conference management software packages for study, and observed how they managed their users' passwords. Except for CMT, a closed source service from Microsoft which we cannot confirm, all others have been identified as having been insecure in the past, but they have all been patched within the last few

years. In total, we examined 135 conferences from security, systems, and networking, using the past five years as the time window. For each conference, we checked which conference software it used and whether it was insecure. If the website was still live, we used the same "forgot password" option as we did in the industry study to validate whether it was insecure. Otherwise, we checked our own email history as well as the email of our colleagues to confirm whether they were insecure.

4.2 Our Findings

The detailed inspection results for these conferences are presented in Table 6. Interestingly, we notice that the majority (78.4%) of them use HotCRP, especially in systems conference management, with the second being EasyChair, which accounts for 14.2%. Surprisingly we found 97.1% of HotCRP sites were insecure. We would have assumed the conference sites managed by security researchers should be more secure, but there was no exception for the security conferences.

Observation 4 *Surprisingly, website owners often do not follow best security practices, even when they are well-educated and understand the risks.*

Also, it is important to note that the trend is moving towards good practice, as we have seen Oakland, SIGCOMM, and MobiCOM in 2014 store hashes when they use HotCRP. However, why were so many conferences before 2014 insecure, especially those using HotCRP? Fortunately, HotCRP is open source, so we decided to inspect its source to determine how it handled passwords and when it started storing hashed passwords.

We examined the code and configurations on the git repository of the most recent release of HotCRP. Interestingly, we noticed that it contains a flag called "safePasswords", which enables the storage of password hashes instead of plaintext. This flag is enabled by default. However, work on this feature did not begin until August 2013, as documented in the changelog [3]. This explains why all conferences before 2013 that used HotCRP were insecure: HotCRP did not have the ability to store password hashes until late summer 2013. Then why are some conferences in 2014 still insecure? We suspect they are running an older version of HotCRP (such as all the conferences hosted by USENIX) or the administrators did not configure it correctly. If it is the latter case, our Observation 4 is further reinforced. We would like to emphasize that the confidentiality of research is considered important; if someone obtained the password to an account for a conference, they could view papers or modify reviews.

Finally, we noticed from our investigation that EasyChair and SoftConf fixed this problem in 2011, and EDAS fixed in 2012. Also, we did find that one user submitted a report to PLAINTEXTOFFENDERS exposing EasyChair's plaintext practice in 2011.

5 Discussions and Implications

After analyzing insecure sites in both industry and academia, we must answer why websites are still storing plaintext passwords. To this end, we would like to ask the following questions:

Conference Name	2014		2013		2012		2011		2010	
	Software	Plaintext?	Software	Plaintext?	Software	Plaintext?	Software	Plaintext?	Software	Plaintext?
CCS	EC	✗	EC	✗	EC	✗	EC	✔	EC	✔
CSF	EC	✗	EC	✗	HC	✔	HC	✔	EC	✔
ESORICS	EC	✗	EC	✗	EC	✗	EC	✔	EC	✔
NDSS	HC	✔	EC	✗	EC	✗	HC	✔	SC	✔
Oakland	HC	✗	HC	✔	HC	✔	HC	✔	HC	✔
RAID	HC	✔	EC	✗	HC	✔	HC	✔	EC	✔
USENIX-Security	HC	✔	HC	✔	HC	✔	HC	✔	HC	✔
ASPLOS	HC	✔	EC	✗	SC	✗	SC	✔	SC	✔
EuroSys	HC	✔	HC	✔	HC	✔	HC	✔	HC	✔
FAST	HC	✔	HC	✔	HC	✔	HC	✔	HC	✔
HPCA	HC	✔	HC	✔	HC	✔	HC	✔	HC	✔
ISCA	HC	✔	HC	✔	HC	✔	HC	✔	HC	✔
LISA	HC	✔	HC	✔	HC	✔	HC	✔	HC	✔
MICRO	HC	✔	HC	✔	HC	✔	HC	✔	HC	✔
OSDI	HC	✔	-	-	HC	✔	-	-	HC	✔
PACT	HC	✔	HC	✔	HC	✔	HC	✔	HC	✔
SenSys	HC	✔	HC	✔	HC	✔	HC	✔	HC	✔
SoCC	HC	✔	HC	✔	HC	✔	HC	✔	CM	✗
SOSP	-	-	HC	✔	-	-	HC	✔	-	-
USENIX-ATC	HC	✔	HC	✔	HC	✔	HC	✔	HC	✔
VEE	HC	✔	HC	✔	HC	✔	EC	✔	HC	✔
CoNEXT	HC	✔	HC	✔	HC	✔	ED	✔	ED	✔
IMC	HC	✔	HC	✔	HC	✔	HC	✔	HC	✔
MobiCom	HC	✗	HC	✔	HC	✔	HC	✔	HC	✔
MobiHoc	ED	✗	ED	✗	HC	✔	HC	✔	ED	✔
MobiSys	HC	✔	HC	✔	HC	✔	HC	✔	HC	✔
NSDI	HC	✔	HC	✔	HC	✔	HC	✔	HC	✔
SIGCOMM	HC	✗	HC	✔	HC	✔	HC	✔	HC	✔

Table 6. Statistics on the Conference Submission Sites. Symbol ✔ denotes the corresponding site was plaintext, ✗ denotes not plaintext, and − denotes the conference was not held in that year.

Q1. Does the fact that a website is physically isolated make it more secure?

Insight 2 *Site owners may think that their sites are sufficiently secured or isolated from attackers that they do not need to worry about intruders. However, repeated security breaches of websites that considered themselves secure have proved this is unrealistic.*

Q2. Why have operating systems been hashing passwords for almost 50 years, as in Multics or UNIX, but some websites today do not?

Insight 3 *Such systems were designed by experts. They quickly discovered that they could not eliminate the threat of a breach and therefore came up with a provably secure solution. Unlike with the development of an OS, a website can be created by someone with no experience. Therefore, some websites are developed without following any best practices.*

Q3. Is a password a critical piece of private data that is as sensitive as bank accounts and SSNs?

Insight 4 *Some websites do not store any valuable personal data like addresses or credit card numbers. Therefore, owners of those sites may think that there is no need to protect passwords. However, due to password reuse, an attacker can use passwords retrieved*

from a breached site to retrieve valuable data from a more secure one. This opens up an attack vector that very secure websites cannot protect against. This threat is significant enough that even sites that do not store other sensitive data should consider passwords as highly sensitive. However, many of these sites do not.

It is important to note that these insights are not revolutionary, and have been the subject of previous discussions of password security. However, we wish to very clearly emphasize these conclusions; it is crucial that both academia and industry unequivocally condemn plaintext password storage so as to provide correct guidance to website owners.

6 Mitigations and the Future

Getting websites to change their password practices is clearly difficult. Here are several mitigations for users and potential solutions for the future.

Password generator and manager. One solution is to use a password generator to create unique passwords for each website, and then use a password manager to store each password. However, this requires a master password or a key and (in the case of an online password manager), introduces the challenge of secure password transmission across the Internet. There are many password managers, however, and there are ones that do take security seriously [6]. Therefore, this is likely a significant security improvement for the average user if they can keep their master password safe.

Client-side hashing. With a client-side program or browser extension [36], a user can hash their password before sending it to a website, effectively making that hash their de facto password for that site. If the user adds a salt to their password based on the site in question (by, for example, using the site domain), the password's hash will be different for every site. From the site's perspective, the user is sending a long string of random characters, with no hint as to the original password and salt. Therefore, even if a site with poor security is compromised, the only user information revealed is the hash of their password, which is unique to—and only usable for—that specific site.

Single sign-on and related technologies. Another alternative for securing login credentials is to trust them with a central authority and use a framework like OpenID, OAuth, or Facebook Connect. Such a system requires the user to log in to the central authority, which then authenticates the user to any third-party site integrated with the system. An advantage of this is that the user never has to give their credentials to any third-party site. However, these systems are not foolproof [40], and the user must trust that the central authority is storing their password securely.

Established standards. Many aspects of password security lack an agreed-upon standard [19]. If a comprehensive standard were developed for password storage, it could provide a baseline to compare sites to. In addition, a reference implementation from a trusted authority could discourage developers from designing their own potentially flawed authentication, and provide developers with an authoritative, publicly audited starting point for making secure design decisions.

7 Related Work

There has been a substantial amount of research centered around password security. In general, the research mostly focuses on how to generate (e.g., [23,38,30]), transfer (e.g., [34,44]), store (e.g., [26]), manage (e.g., [45,24,17]), and attack and defend passwords from numerous attack vectors (e.g., [32,28,27,46]). In this section, we review related work on password storage attacks and defenses.

In 2010, Bonneau et al. performed a large-scale general analysis of how websites handle passwords. They covered many aspects of password security, including password storage. In their study of 150 sites, 29% emailed user passwords in plaintext [19]. To motivate websites to fix their bad password practices, PLAINTEXTOFFENDERS [5] was launched in April 2011. Our work is closely related to the site in that we both aim to raise public awareness. The difference is that we also perform a more systematic study of the problem and its solutions.

Since security experts advocate storing hashes, there has been a great amount of interest in efficiently cracking hashes and obtaining plaintext. The earliest attempts stem from brute forcing and evolved into dictionary attacks. Recently, there were also efforts to make dictionary attacks smarter by employing Markov models (e.g., [33]), probabilistic context free grammars (e.g., [42]), and history based guessing (e.g., [46]).

Interestingly, the large amounts of plaintext passwords revealed from recent password leakages have also enabled many valuable password studies. Dell'Amico et al. [22] analyzed the strength of passwords from leaked MySpace passwords and two other websites. Weir et al. [41] leveraged the 32M passwords leaked from RockYou to test the metrics (especially the entropy) for password creation policies. Das et al. [21] investigated several hundred thousand leaked passwords and conducted a password reuse survey, finding that about 43-51% users reuse their passwords for several websites. In addition, they have also developed a password guessing algorithm to guess cross-site passwords.

8 Conclusion

Despite almost 50 years of practice of storing and managing passwords, we still find many websites storing plaintext passwords. In this paper, we systematically analyzed the insecure sites from both industry and academia to investigate the reasons behind this issue. We found 11 of the most popular 500 websites and over 100 conference submission sites from the past five years that do not hash user passwords. Finally, we discuss how the illusion of security, lack of security experts, and lack of attention to secure passwords on sites not storing other sensitive data are likely the root causes of why today we are still storing plaintext passwords.

Acknowledgements. We thank the anonymous reviewers for their feedback. This research was supported in part by AFOSR grant FA9550-14-1-0119. Any opinions, findings, conclusions, or recommendations expressed are those of the authors and not necessarily of the AFOSR.

References

1. Easychair home page. http://www.easychair.org/.
2. Edas. https://www.edas.info/index.php.
3. Hotcrp begin work on safe password storage. https://github.com/kohler/hotcrp/commit/127613bfbee196c06b6e799bbc4705b7be37cc06.
4. Microsoft's conference management toolkit. https://cmt.research.microsoft.com/.
5. Plain text offenders. http://plaintextoffenders.com/.
6. Security - keepass. http://keepass.info/help/base/security.html.
7. Softconf start v2 conferencemanager. http://www.softconf.com/.
8. uclassify. http://www.uclassify.com/.
9. Hackers released the passwords of over 70 million chinese internet accounts. https://dazzlepod.com/rootkit/, 2011.
10. Ieee data breach: 100k passwords leak in plain text. http://www.neowin.net/news/ieee-data-breach-100k-passwords-leak-in-plain-text, 2011.
11. rootkit.com cleartext passwords. https://dazzlepod.com/rootkit/, 2011.
12. Linkedin password hack: Check to see if yours was one of the 6.5 million leaked. http://www.huffingtonpost.com/2012/06/07/linkedin-password-hack-check_n_1577184.html, 2012.
13. Militarysingles.com hack exposes over 160,000 users' information. http://www.databreaches.net/militarysingles-com-hack-exposes-over-160000-users-information/, 2012.
14. Yahoo hack leaks 453,000 voice passwords. http://www.darkreading.com/attacks-and-breaches/yahoo-hack-leaks-453000-voice-passwords/d/d-id/1105289?, 2012.
15. Youporn passwords available for download, thousands of users exposed. http://nakedsecurity.sophos.com/2012/02/22/youporn-password-download/, 2012.
16. A. Adams and M. A. Sasse. Users are not the enemy. *Commun. ACM*, 42(12):40–46, Dec. 1999.
17. H. Bojinov, E. Bursztein, X. Boyen, and D. Boneh. Kamouflage: Loss-resistant password management. ESORICS'10, pages 286–302, Berlin, Heidelberg, 2010. Springer-Verlag.
18. J. Bonneau, C. Herley, P. C. v. Oorschot, and F. Stajano. The quest to replace passwords: A framework for comparative evaluation of web authentication schemes. SP '12, pages 553–567, Washington, DC, USA, 2012. IEEE Computer Society.
19. J. Bonneau and S. Preibusch. The password thicket: Technical and market failures in human authentication on the web. In *WEIS*, 2010.
20. B. Calin. Statistics from 10,000 leaked hotmail passwords. http://www.acunetix.com/blog/news/statistics-from-10000-leaked-hotmail-passwords/, 2009.
21. A. Das, J. Bonneau, M. Caesar, N. Borisov, and X. Wang. The Tangled Web of Password Reuse. In *NDSS '14*, February 2014.
22. M. Dell'Amico, P. Michiardi, and Y. Roudier. Password strength: An empirical analysis. INFOCOM'10, pages 983–991, Piscataway, NJ, USA, 2010. IEEE Press.
23. D. Florencio and C. Herley. A large-scale study of web password habits. WWW '07, pages 657–666, New York, NY, USA, 2007. ACM.
24. S. Gaw and E. W. Felten. Password management strategies for online accounts. SOUPS '06, pages 44–55, New York, NY, USA, 2006. ACM.
25. R. A. Grimes. Myspace password exploit: Crunching the numbers.

26. J. Hart, K. Markantonakis, and K. Mayes. Website credential storage and two-factor web authentication with a java sim. WISTP'10, pages 229–236, Berlin, Heidelberg, 2010.
27. T. Holz, M. Engelberth, and F. Freiling. Learning more about the underground economy: A case-study of keyloggers and dropzones. ESORICS'09, pages 1–18, Berlin, Heidelberg, 2009.
28. B. Ives, K. R. Walsh, and H. Schneider. The domino effect of password reuse. *Commun. ACM*, 47(4):75–78, Apr. 2004.
29. E. Kohler. Hotcrp conference management software. http://read.seas.harvard.edu/~kohler/hotcrp/, 2014.
30. S. Komanduri, R. Shay, P. G. Kelley, M. L. Mazurek, L. Bauer, N. Christin, L. F. Cranor, and S. Egelman. CHI '11, pages 2595–2604, New York, NY, USA, 2011. ACM.
31. M. D. McIlroy. A research unix reader: Annotated excerpts from the programmer's manual. 1971.
32. R. Morris and K. Thompson. Password security: A case history. *Communications of the ACM*, 22(11):594–597, 1979.
33. A. Narayanan and V. Shmatikov. Fast dictionary attacks on passwords using time-space tradeoff. CCS '05, pages 364–372, New York, NY, USA, 2005. ACM.
34. M. Peyravian and N. Zunic. Methods for protecting password transmission. *Computers& Security*, 19(5):466–469, 2000.
35. J. Raphael. Gawker hack exposes ridiculous password habits. http://www.pcworld.com/article/213679/Gawker_Hack_Exposes_Ridiculous_Password_Habits.html, 2010.
36. B. Ross, C. Jackson, N. Miyake, D. Boneh, and J. C. Mitchell. Stronger password authentication using browser extensions. In *Proceedings of the 14th Usenix Security Symposium*, volume 31, 2005.
37. J. H. Saltzer. Protection and the control of information sharing in multics. *Commun. ACM*, 17(7):388–402, July 1974.
38. R. Shay, S. Komanduri, P. G. Kelley, P. G. Leon, M. L. Mazurek, L. Bauer, N. Christin, and L. F. Cranor. Encountering stronger password requirements: User attitudes and behaviors. SOUPS '10, pages 2:1–2:20, New York, NY, USA, 2010. ACM.
39. M. Siegler. One of the 32 million with a rockyou account? you may want to change all your passwords. like now. http://techcrunch.com/2009/12/14/rockyou-hacked/, 2009.
40. R. Wang, S. Chen, and X. Wang. Signing me onto your accounts through facebook and google: A traffic-guided security study of commercially deployed single-sign-on web services. In *SP '12*, pages 365–379. IEEE, 2012.
41. M. Weir, S. Aggarwal, M. Collins, and H. Stern. Testing metrics for password creation policies by attacking large sets of revealed passwords. CCS '10, pages 162–175, New York, NY, USA, 2010. ACM.
42. M. Weir, S. Aggarwal, B. d. Medeiros, and B. Glodek. Password cracking using probabilistic context-free grammars. SP '09, pages 391–405, Washington, DC, USA, 2009. IEEE Computer Society.
43. C. White. Adobe leaks 150 million passwords; facebook and others impacted. http://www.neowin.net/news/adobe-leaks-150-million-passwords-facebook-and-others-impacted, 2013.
44. C.-C. Yang, T.-Y. Chang, and M.-S. Hwang. Security of improvement on methods for protecting password transmission. *Informatica*, 14(4):551–558, Dec. 2003.
45. K.-P. Yee and K. Sitaker. Passpet: Convenient password management and phishing protection. SOUPS '06, pages 32–43, New York, NY, USA, 2006. ACM.
46. Y. Zhang, F. Monrose, and M. K. Reiter. The security of modern password expiration: An algorithmic framework and empirical analysis. CCS '10, pages 176–186, New York, NY, USA, 2010. ACM.