Analyzing Websites for User-Visible Security Design Flaws

Laura Falk
University of Michigan
Computer Science and Engineering
laura@umich.edu

Atul Prakash
University of Michigan
Computer Science and Engineering
aprakash@umich.edu

Kevin Borders
University of Michigan
Computer Science and Engineering
kborders@umich.edu

ABSTRACT

An increasing number of people rely on secure websites to carry out their daily business. A survey conducted by Pew Internet states 42% of all internet users bank online. Considering the types of secure transactions being conducted, businesses are rigorously testing their sites for security flaws. In spite of this testing, some design flaws still remain that prevent secure usage. In this paper, we examine the prevalence of user-visible security design flaws by looking at sites from 214 U.S. financial institutions. We specifically chose financial websites because of their high security requirements. We found a number of flaws that may lead users to make bad security decisions, even if they are knowledgeable about security and exhibit proper browser use consistent with the site's security policies. To our surprise, these design flaws were widespread. We found that 76% of the sites in our survey suffered from at least one design flaw. This indicates that these flaws are not widely understood, even by experts who are responsible for web security. Finally, we present our methodology for testing websites and discuss how it can help systematically discover user-visible security design flaws.

1. INTRODUCTION

Secure websites have become an integral part of our day-to-day-life. People conduct both their personal and job-related business using these sites. Many consumers purchase goods online using sensitive credit card information. A large number of people have also given up conventional banking in favor of online banking. One can even buy and sell stocks with the click of a button through broker websites. Due to the sensitive nature of these sites, security is a top priority. They all deploy protocols such as SSL and many of them hire security experts to conduct vulnerability assessments.

Despite all of the security mechanisms and rigorous testing, security is still a major concern both for institutions who offer secure websites and for potential users. Forbes.com conducted a survey, *The State of Customer Satisfaction with Online Banking* in which over 900 people responded. The results were published on April 17, 2007 [7]. The participants fell into three categories:

- Used online banking applications and paid bills online through their bank's website.
- Used online banking applications but not online bill payments.
- Used no online banking activities whatsoever.

Those who used online banking were satisfied with the services, satisfied with the financial institutions that offer them, and found them usable. However, those who chose not to use online banking cited security concerns as a reason why they did not use the services. Generally speaking, the article concluded that the security measures being employed now are securing online banking to a much greater degree than just a few years ago and that a major problem is in *educating* potential customers on the security and convenience of online banking. As Schechter et al. have shown, people tend to disregard SSL indicators, leaving them vulnerable to phishing attacks [15]. However, studies have not focused on design flaws that would prevent even the most educated user from being able to make the right security decisions.

In this paper, we analyze 214 U.S. financial institution websites for *design flaws* that prevent secure usage. Design flaws differ from typical software bugs that can be fixed by applying patches. Design flaws are a result of decisions made during the website design phase, such as how to implement security features. These design decisions promote insecure user behavior. Our study was conducted during November and December of 2006. The list of the 214 sites that we used in our study was obtained by getting a list of banks on the Internet in the United States from [9] excluding any non-working links and links to sites that obviously did not offer current financial services (e.g., historic banks such as First Bank of the United States). The final list we used can be found at [1]. We chose financial institutions in particular because they have a substantial stake in securing their websites and in providing secure access for their customers. We checked each of the sites for the following design flaws.

1. *Break in the chain of trust:* Some websites forward users to new pages that have different domains without notifying the user from a secure page. In this situation, the user has no way of knowing whether the new page is trustworthy.
2. *Presenting secure login options on insecure pages:* Some sites present login forms that forward to a secure page but do not come from a secure page. This is problematic because an attacker could modify the insecure page to submit login credentials to an insecure destination.
3. *Contact information/security advice on insecure pages:* Some sites host their security recommendations, contact information, and various other sensitive information about their site and company on insecure pages. This is dangerous because

Symposium on Usable Privacy and Security (SOUPS) 2008, July 23–25, 2008, Pittsburgh, PA USA

an attacker could forge the insecure page and present different recommendations and contact information.

4. *Inadequate policies for user ids and passwords:* It is important to maintain consistent and strong policies on passwords and user ids. We found some sites allow customers to use short passwords or they require e-mail addresses for user names.

5. *E-Mailing security sensitive information insecurely:* E-mailing any sensitive information is dangerous. We found that some sites offered to send statements and passwords through e-mail but not very many people have secure e-mail.

We focused on these five types of flaws in particular because they were the prevalent on an initial subset of sites that we examined manually. In Section 3, we discuss our methodology for selecting these particular design flaws in greater detail.

During the course of our study, we found that 30% of the sites surveyed break the chain of trust, 47% present a login page on an insecure page, 55% present contact and other sensitive information on insecure pages, and 31% allow e-mail addresses as user names. Overall, only 24% of the sites were completely free of these design flaws, indicating that some of the flaws we identified are not widely understood, even among institutions where security is critical.

The widespread existence of secure usability design flaws on financial websites suggests that the experts at these institutions do not test for them. Therefore, this paper also explores ways of automatically detecting these design flaws by searching for certain strings on these websites. We developed a tool for automatically detecting flaws that would also be applicable for discovering design flaws that are not covered in this study, such as using personally identifiable information for authentication.

One of the most interesting design flaws we discovered is the presentation of FAQs and contact information on insecure pages. In the past, FAQs and contact information were usually sent through the mail to the customer. It is not generally recognized that this information should be protected. However, when this information is presented online, the user becomes vulnerable to social-engineering and offline attacks as a result of the information being displayed on an insecure page.

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 gives a brief synopsis of the security design flaws that we looked for and their implications. Section 4 discusses our methodology for analyzing the flaws and Section 5 discusses our results. Section 6 concludes.

2. RELATED WORK

It is well-known that security vulnerabilities may result if users are unable to understand security-relevant information presented by applications. The work by Cranor et al. [2] shows that it can be a significant challenge to design interfaces that present P3P website privacy policies to users in a straightforward manner. Schechter et al. [14] show that most users are unlikely to correctly interpret SSL security context presented by a browser as part of a decision whether to authenticate to a website. Our work is similar in that some of the flaws that we consider impair a user's ability to make correct security decisions. However, our work differs in that the cause is not poor or confusing client-side interfaces. Instead, the flaws originate in poor design or policy choices at the server that prevent or make it difficult for users to make correct choices from the perspective of securing their transactions.

Provos et al. [13] provide an analysis of web-based malware. They conducted a twelve-month study and found several attack strategies that turned regular web pages into malware sites. They identify four different aspects of content control that are responsible for causing browser exploitation. They are advertising, third-party widgets, user-contributed content and web server security. Through analysis and examples they show how each can be used to exploit web browsers. Their analysis is for specific malware and vulnerabilities exploited at the client side. In our work, we do not focus on implementation-related vulnerabilities, but design or policy-level flaws.

Network scanners, such as Nessus [11], and application-level website scanners, such as AppScan [17], can be used to analyze for many configuration and implementation bugs, such as use of unpatched services and vulnerability to cross-side scripting or SQL-injection attacks. As far as we are aware, the design flaws that we examine are currently not identified by these scanners. Later on in the paper, we outline our methodology for automatically identifying candidates for a subset of our design flaws.

In our analysis, one of the design flaws we examined was whether sites have policies for strong passwords and avoid user ids based on e-mail addresses and social security numbers. We did that analysis because using easy-to-guess user ids or weak passwords is normally considered to be poor security practice. However, there is some recent evidence to the contrary. Florencio et al. [6] conducted an analysis of online passwords. They state that strong passwords do nothing to protect online users from password attacks such as phishing and key logging, and simply put considerable burden on the user. They found that relatively weak passwords are sufficient to make brute-force attacks unrealistic as long as the "three strikes" rule is in place. The "three strikes" rule says that a user has three tries to login before they are locked out. At that point, they have to wait a certain period before they are allowed to try logging in again or they have to contact customer service. They discovered that increasing the strength of the user id rather than the password is better when increased credential space is necessary. However, this study was not specific to financial institutions and it is possible that their recommendations of allowing weak online passwords are not applicable to that domain. Earlier studies, such as [12], have argued that the three-strike rule may not be sufficient. If the user ids become known, parallel dictionary attacks could occur against a large number of accounts by trying common passwords for all the accounts. Even with lockout, the attack could be repeated after a few days when many of the accounts are unlocked by legitimate users.

Fu et al. [8] point out several common mistakes in providing client authentication services on the web. They particularly examine the design of authenticators in the web cookies that are provided to clients and find that poor design of authenticators could permit an adversary to forge authenticators for an unknown user (called *existential forgery*) or a selected user (called *selective forgery*). Our research is complementary to this work.

Website authentication mechanisms are an important tool in protecting the user from various attacks. Some of these mechanisms are not fully understood and at times completely ignored by the user. Schechter et al. [15] evaluate website authentication measures used to protect the user from phishing, man-in-the-middle, and other forgery attacks. They found that almost all users will enter their passwords even when https and site-authentication images are absent. Our study differs in that we look for server-side design flaws that preclude secure usage even by an expert who correctly interprets all security indicators.

Dhamija et al. [4] studied phishing attack strategies. They provide empirical evidence about which strategies are successful in deceiving the user. Their study first analyzed a set of captured

phishing attacks from which they devised hypotheses about why certain strategies work. They found that 23% of participants did not look at browser-based cues such as the status bar, security indicators, and address bars, which lead to incorrect choices 40% of the time. They also found that visual deception can fool even an educated user. Our study complements their work by analyzing websites that suffer from design flaws and thus make users more vulnerable to online fraud.

The appearance of a usable graphical interface and marketing claims of easy usability can be very deceiving. Whitten et al. [18] evaluated whether or not PGP 5.0 can be used successfully by novices in order to send and receive electronic mail securely. They chose PGP 5.0 because the user interface appeared to be reasonably well-designed. When their participants were given 90 minutes to sign and encrypt a message using PGP 5.0, the majority were unable to do so. They found a number of user interface design flaws which seemed to limit the success of the participants. Our work differs in that we focus on server-side security design flaws rather than client-side design flaws.

Security toolbars can be valuable in warning the user about fraudulent websites. However, they are only useful when the user views them as helpful and takes full advantage of them. Wu et al. [19] conducted two user studies, one of which focused on web security toolbars. They found that the security indicators present in web browsers were ineffective at preventing phishing attacks. One of the users who was fooled by a phishing attack cited the fact that his own bank redirects him to a different domain without warning (This is the "Break in the chain of trust" design flaw in our study) and therefore he was not suspicious of a phishing site with the same behavior. Our study is different because it characterizes server-side design flaws rather than evaluating client-side solutions for fixing these vulnerabilities.

In order for a user to make an informed decision, certain aspects of security situations need to be made visible to the user. Rogerio de Paula et al. [3] report their experiences in designing, developing, and testing technical infrastructures in order to see how security is manifested during these phases. The discovered that the implementations and integration of various components of today's technical infrastructure is awkward and hard to use. Our work is similar in nature. We analyzed sites for visible design flaws that would lead users to make poor decisions regarding security.

Although people are aware of phishing attacks and the risks involved, they might not completely understand the strategies in identifying phishing emails. Downs et al. [5] conducted a study that involved interviewing 20 non-expert computer users to understand the decisions they make when encountering suspicious email. The authors found that awareness of the risks is not linked to perceived vulnerability or to useful strategies in identifying a suspicious email. Our work complements this study.

3. SECURE USABILITY DESIGN FLAWS

Initially, we browsed twenty financial websites and examined their security policies and practices. This helped us identify and focus on the most interesting and common user-visible design flaws. As we examined these sites from a secure usability aspect, we found certain design features of the web pages that made it very difficult for someone to use the site securely. As we evaluated these sites, we focused our attention on design flaws that we could detect using automated tools. We chose not to consider flaws that: (1) required account access or account creation, (2) were implementation flaws that were opaque to the user, or (3) would have been difficult to evaluate automatically, such as those that would require interpretation of arbitrary natural-language security questions, for example, to evaluate password reset policies. After narrowing down the set of flaws to those that fit the above criteria, we focused on flaws that were prevalent on sites from our manual inspection. We selected the following five flaws for our study:

1. *Break in the chain of trust*
2. *Presenting secure login options on insecure pages*
3. *Contact Information/Security Advice on Insecure Pages*
4. *Inadequate policies for user ids and passwords*
5. *E-Mailing security sensitive information insecurely*

Below, we elaborate on the design flaws with specific examples from the bank sites that we examined and discuss the flaws in more detail, with specific examples.

3.1 Break in the Chain of Trust

In general, a good design principle is that a website's pages must give users enough security context so that they can distinguish potentially malicious content from trustworthy content. The use of SSL-protected pages is a simple example of such a context. The implication for a user is that if a website provides an SSL-protected page, the user can trust the contents of that page as long as the user trusts that website.

In our study, we explored a more subtle example of this problem, which we refer to as the *break-in-chain-of-trust* problem. Generally, if a careful user visits a secure website, he or she will look for the institution's name in the URL, prefixed by https. We define the secure website as the *root of trust* for the user. Several financial institution websites start out correctly, but for some transactions, the customer is redirected to a site that has a different domain name than the financial institution's site that was originally visited. The signed certificate also bears a different company name. At this point, the chain of trust is potentially broken because now it is up to the user to determine if the new site is really affiliated with the financial institution (i.e., the financial institution trusts the new site) or it happens to be a window that popped up as a result of some other event, or even an attack.

We found several instances in which no information is provided on the original site stating that the user would be redirected to a third-party site and that the third-party site can be trusted. (Naturally, this information would need to be protected as well and placed on a secure page.) This results in a careful customer/user having insufficient information to decide if the third-party website can be trusted.

We often see warnings at websites when they redirect you to a third-party site that state the third-party site is *not* under the referring site's control and that the users should use that site at their own risk. But, it is less common to see messages explicitly saying that a website is taking you to a trusted third-party site. We see that as a gap because usually, there is no easy way for the user to determine whether they should trust a third-party site.

The highest risk in the above scenario occurs when the user is taken from an *insecure* bank's web page to a third-party website that provides financial services. One example of this is TCF Bank, *http://www.tcfbank.com* as shown in Figure 1. A customer wishing to go to online banking clicks on the link and is transferred to the following URL: *https://secure.mvnt4.com /tcf/ OnlineBanking/ index. jsp*. To simulate a careful user, we viewed the signed certificate of *http://secure.mvnt4.com*, but TCF Bank was not the owner of this certificate. The owner was Metavante Corporation. There was no indication how this corporation was connected with TCF

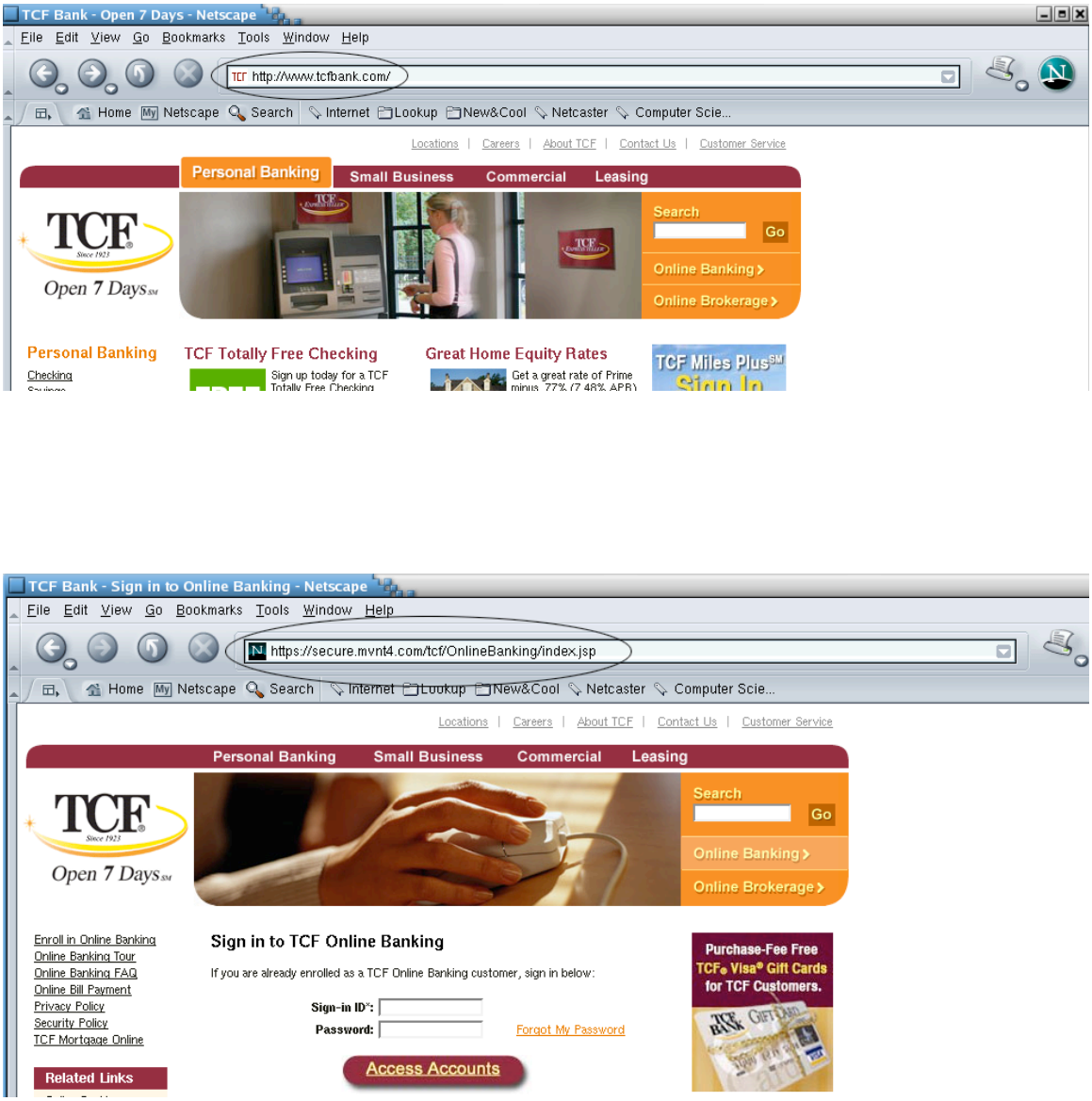

Figure 1: Improper redirection to another site

Bank and furthermore, an insecure page forwarded the user to this third-party site. [1]

A slightly safer, but not the safest, way to handle such transitions to third-party sites would be with a two-step process:

1. The user visits the bank's home page and clicks on a service link.

2. The user is taken to an SSL-protected page with the same domain name as the home page. The user can verify the domain name and that it is SSL-protected, and can conclude that the page contents can be trusted.

3. A link on that secure page forwards the user to a secure third-party site. The implication is that the user should trust the third-party site because the link to it was provided by a trusted page.

However, even in the above case, trusting the third-party website may not be a straightforward decision for a careful user. Browsing should be seamless for the user without such decisions. When presented with a difficult or confusing decision, users are likely to avoid the decision and go with the default action or let the site guide them, which leads to a bad security decision. For example, at the University of Michigan credit union's website, where one of the authors has an account, users authenticate properly and are taken to a secure account page. However, one of the options on the accounts page is a link to sign-up for Bill Pay services. If an account holder decides to sign-up for Bill Pay, a new window pops up that belongs to a third-party vendor. Trusting this pop-up window in some cases would be appropriate, but in the case of this credit union, this window asks the user to enter fairly intrusive information, such as mother's maiden name, social security number, account number, and birth date. No message is given on the original account window indicating that this pop-up from third-party website will occur and that the credit union vouches for this third-party site to safeguard the user's data. We would argue that the credit union could have handled this design better, by either providing better disclosures or by not requiring the user to enter that information. This way, a careful user would be less likely to have security concerns about the third-party site.

YURLs [20] have been proposed for website authentication. They provide a means for users to build trust in website authentication. A YURL helps implement a decentralized authentication model that uses the hash of a website's public key as the URL authority. Each user maintains a list of trusted sites, identified by the hashes of the site's public keys. One could conceive of a trust model similar to that used in PGP, where financial institutions certify URLs as being trustworthy and they get added to users' lists of trusted sites. However, at present, current web servers and browsers do not implement YURLs. We expect that there would be significant challenges in implementing a trust model for the web and incorporating it in the browsers. With the current infrastructure, the best way for websites to maintain a secure root of trust would be for them to provide adequate notifications before taking you to third-party sites and to always make such transitions from secure pages.

[1]The institutions that we use as examples in this paper (e.g., TCF bank) should only be considered representative examples to help make the discussion more concrete. For the purpose of this paper, we primarily picked on institutions that have branches in our home town or we have experience with as customers. As noted in the Results section, the problems we found are widespread. The authors have alerted the institutions that we specifically identify in the paper about the problems.

3.2 Presenting Secure Login Options on Insecure Pages

Login pages and options displayed on insecure pages leave users vulnerable to man-in-the-middle attacks. They have no way of knowing if their usernames and passwords are being sent to a hacker site. This makes it impossible for a user to make the correct decision. An example of this type of flaw would be a website that provides a secure Javascript login window on a non SSL-protected page. A real example of this was found at LaSalle Bank's website *http://www.lasallebank.com* (shown in Figure 2). The login area is at the top left side of a non-SSL page. At LaSalle bank, as at other banks, the embedded Javascript code for the login window does submit the information via SSL. However, the user has no obvious way of knowing that prior to submitting the information. Since the overall page is not SSL-protected, the entire page could be spoofed, including the login box, with a man-in-the-middle or DNS hijacking attack. We also noticed that the login area shows a picture of a lock and the phrase "Secured with SSL" technology. However, this is only a market strategy that actually makes things worse by giving users a false sense of security. Other examples of this style of vulnerability include password-reset forms or forms for opening new accounts that are embedded in insecure pages.

In these situations, the information may be submitted securely but the user is not provided with any assurance of that being the case. The browser could potentially analyze the HTML or Javascript code to see if the information would be submitted securely and the results displayed in a user-verifiable way. However, doing that analysis in a provable way is likely to be non-trivial. As shown by [19], the majority of users do not pay attention to security toolbars. It may be the case that users might disregard form submission destinations as well. If the information could be presented as a pop up window and block access to the page, this might encourage and help the user behave in a secure manner. Simple checks, such as examining for the presence of "https" in the Javascript code behind the Submit button are not sufficient. A code analyzer would also have to determine that the information collected in the form cannot be somehow saved by the Javascript and sent to another site.

We have reason to believe that this particular problem is recognized by some financial institutions. Several years ago, Vanguard, a brokerage company, used to provide the login window on their home page (which was only accessible as an http page). One of the authors contacted them at that time raising it as a potential security concern. The response was that if a customer was concerned, the customer could hit the Submit button without entering a valid user id and password, and that would take the customer to an SSL-protected login page. Since then, however, Vanguard modified their login process, moving the login window to an SSL-protected page. We have noticed a similar trend at several other financial institutions. However, many still continue to provide an unsafe login page.

In principle, even if an SSL-protected page provides a login window, there is no guarantee that the logic for the login window's Submit button is properly implemented to send the information securely. However, from a customer's perspective, there is an implied understanding with a financial institution that an SSL-protected page provided by that institution has trustworthy contents, including handling of contents submitted to that page. In this paper, we assume that this implied understanding holds.

3.3 Contact Information/Security Advice on Insecure Pages

A well-known principle in security protocol design is that not only the data channel must be secured, but also the context that is

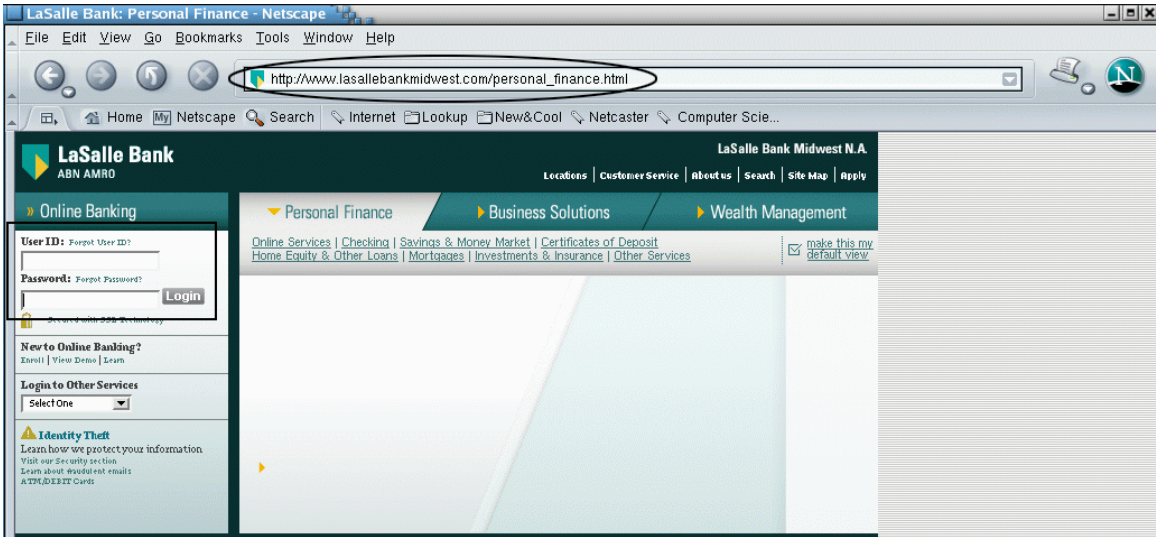

Figure 2: Login information on an insecure page

used to generate the session keys for the channel [16]. For example, SSL 2.0 was vulnerable to cipher rollback attack because it did not adequately protect the key negotiation steps. A similar point is made in [10] regarding the importance of protecting security-context for a broader class of security applications.

Unfortunately, we found widespread violation of this principle at many financial websites. The specific design flaw we looked for is whether financial websites provide their contact information or security advice from an insecure page. Contact information or security advice can be considered *security-relevant context* because users rely on that information being correct for security-sensitive operations. Consider the case where the customer service contact information for resetting passwords is provided on an insecure page. To compromise the system, an attacker only needs to spoof or modify the page, replacing the customer service phone numbers with bogus numbers. These bogus phone numbers could be set up to collect information from customers when they call for help. For example, if a user calls to reset their password, it is standard practice in U.S. for customer service agent to ask the user for their social security number, birth date, and possibly mother's maiden name. Most users will gladly volunteer that information, assuming that their bank is only exercising diligence in verifying their identity before resetting their password. Such an offline attack with bogus phone numbers is not hypothetical. One of the authors of this paper recently (on November 6th, 2007) received the following e-mail:

> *Dear Credit Union customer,*
>
> *We regret to inform you that we have received numerous fraudulent e-mails which ask for personal account information. The e-mails contained links to fraudulent pages that looked legit.*
>
> *Please remember that we will never ask for personal account information via e-mail or web pages.*
>
> *Because of this we are launching a new security system to make Credit Union accounts more secure and safe. To take advantage of our new consumer Identity Theft Protection Program we had to deactivate access to your card account.*
>
> *To activate your card please call (877) 410-6468*
>
> *Activation is free of charge and will take place as soon as you finish the activation process.*

We did not call the phone number given. However, it appears to have no purpose other than to collect sensitive information from credit union members.

A related example of this style of design flaw is shown in Figure 3. In order to update the name and address, this institution gives you an address and tells you to write to them (including information such as account number and social security number) but the page is not SSL-protected. An attack on the system would be to replace the page so it contains a spoofed mailing address, which is controlled by the attacker (e.g., a temporary P.O. Box).

3.4 Inadequate Policies for User IDs and Passwords

The specific design flaws in this category that we looked for were the following:

- The use of social security numbers and e-mail addresses for user ids. Although such credentials are easy to remember for the user, they are unfortunately also easy to guess or collect.
- Not having any stated policy on allowed passwords or permitting clearly weak passwords. This makes it easier for accounts to be vulnerable to dictionary attacks.

E-mail addresses are easily collected from the Internet, which spammers use to build their address database. Suppose that an attacker has some information about the individual and his/her bank, which uses e-mail addresses as user ids. It would be straightforward for the attacker to look up the user's e-mail address(es). This gives the attacker the customer's user id, leaving only the password as the barrier for gaining access. Similarly, if an attacker has obtained enough information to have a victim's social security number and the bank uses social security numbers as user ids, the same attack could be launched. Florencio et al. [6] show that attackers can easily bulk guess the space of social security numbers when they are used as user ids. The space is relatively small because

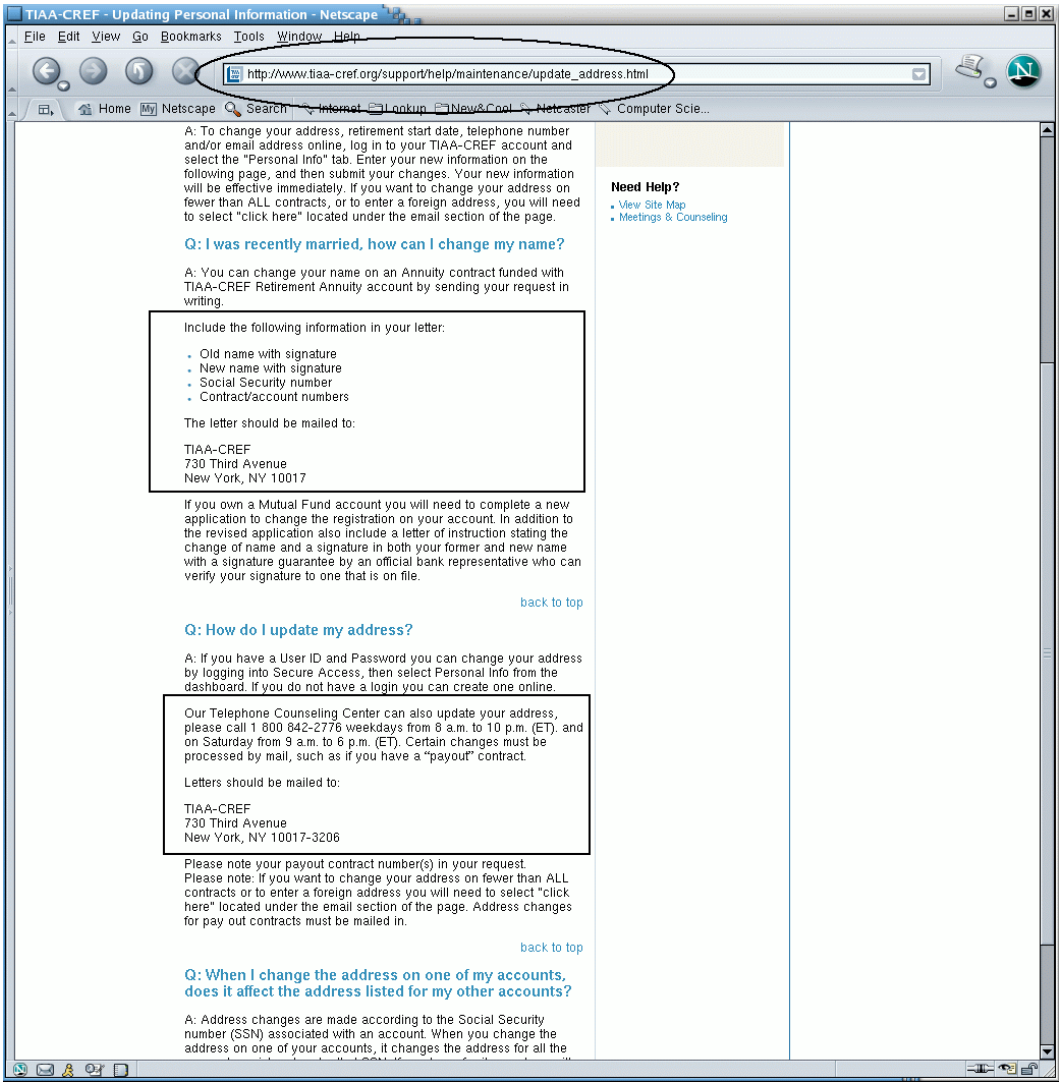

Figure 3: Security-sensitive contact information on an insecure page

there are only nine digits in a social security number and the digits range from 0 to 9.

An example of this problem is the LaSalle Bank website, *www.lasallebank.com*, and TIAA CREF, *www.tiaa-cref.com*. Both sites default to your social security number as the user id. Most financial institutions, including these two, use the social security number as the initial user id, but give the user the option to change it to another value. However, some sites are more explicit about the importance of changing the user id. For example, Fidelity Investments (*www.fidelity.com*) recommends that users change their user id to a value other than their security number.

Just allowing alternative user ids is not sufficient for security. If the website allows the use of e-mail address and/or social security number (along with a password) to reset the user id, then the user id is not providing any extra layer of protection. An attacker could simply try to do a dictionary attack on the reset procedure and reset the user id to a desired value. In our evaluation, we did not check for the existence of this alternative pathway for the use of weak ids. We only checked if the primary login method permitted the use of weak user ids or weak passwords.

3.5 E-Mailing Security-Sensitive Information Insecurely

The specific design flaw we looked for was whether the financial sites offered to send security-sensitive statements or passwords via e-mail. The problem with this procedure is that e-mail data path is generally not secure. If passwords or account statements are e-mailed through an insecure mail server, an attacker could be viewing unencrypted traffic on the network and obtain the sensitive information.

An example of this flaw can be found on the TIAA-CREF website. The specific wording is as follows:

> *Can I have printed statements as well as an electronic copy sent to me?*
>
> *You can elect e-delivery of your statements via the email tab in Secure Access. In addition, when your statement is available there is also an option within Documents to receive a hard copy by selecting the "send by mail" tab. This will generate a hardcopy of your report.*

They offer to send your statements via e-mail but the user is not told whether this will simply be a notification about availability of a statement, a link to the statement, or the actual statement. If it is simply a notification, it would not be a problem. If it is a link, the user potentially becomes more vulnerable to phishing attacks. If it is an actual statement, then the statement is subject to eavesdropping.

4. METHODOLOGY

We used an automated tool to analyze 214 financial institution websites (the list we used can be found at [1]) for our chosen list of design flaws and then confirmed the flaws manually. We used *wget* to recursively download the financial institution websites during November and December of 2006. We chose to download the sites so that we had uninterrupted access and had a consistent, static view of each website. The websites may have fixed the design flaws mentioned in this paper after our initial download.

Once we downloaded each website, we uses scripts to recursively traverse and analyze the HTML pages for certain patterns and identify the security design flaws. Below, we describe the pattern-matching and algorithm used to detect the design flaw.

4.1 Break in the Chain of Trust

For each web site, we recorded the domain and searched each page for URLs that did not match the domain. We looked for two cases: 1) insecure pages making a transition to a secure page and 2) a secure page making a transition to a secure page. For the first case, we considered this transition to be a design flaw. Under no circumstance should an insecure page make a transition to a security-sensitive website hosted on another domain, regardless of whether the destination site uses SSL.

For the second case, we considered this transition to be a design flaw if not properly introduced. If the secure site properly introduced the new site, then we considered the transaction to be safe. A properly introduced site provided a brief notation about the new site and why the user was being transferred. Most generally, this came in the form of pop ups. Automating this was difficult so we checked for proper introduction to third sites manually.

4.2 Presenting Secure Login Options on Insecure Pages

We searched each web page for the string "login". If the string was found, we searched the same page for the strings "username" or "user id" or "password". If the string “login” and “username” or “user id” or “password” were found on the same page, we then verified whether the page was displayed using the http protocol. If this was the case, we assumed this site contained the design flaw.

4.3 Contact Information/Security Advice on Insecure Pages

We searched each web page for the string "contact", "information", or "FAQ". If those strings where found, we checked whether the page was protected with SSL. If not, then we considered it to contain the design flaw.

4.4 Inadequate Policies for User IDs and Passwords

For the case of allowing easy-to-guess user ids, we parsed the web page files searching for the presence of one of the strings: “social security number”, “e-mail”, or “address”. If such a page also contained the strings “login” and “user id”, it was assumed to violate the property. We manually confirmed the results, filtering out any false matches.

For inadequate password strength policies, we parsed the web page files searching for the string “password” (excluding the Login pages). If the string "password" was found, we then searched for the presence of one of the following strings: “recommendation”, “strong”, or “setting”. If any of those strings were found, we made a conservative assumption that the website had a policy on setting strong passwords.

Our count could be optimistic; some sites may require strong passwords without stating an explicit policy. We had no obvious means of verifying this without generating an account on the website. Our count could also be conservative for sites that have poor policies resulting in weak passwords. Thus, our results for this design flaw should only be taken as a rough estimate of the extent of this particular problem.

4.5 E-Mailing Security-Sensitive Information Insecurely

In this case, we used a different set of strings than those for finding Contact Information.

We parsed the web page files, searching for the presence of either of the two strings “statements” or “password” as well as the

Table 1: Summary of Security-relevant Design Flaws at Financial Institutions

Specific design flaw	% of sites affected	Principle violated
Break in the chain of trust	30%	Inadequate security context for informed decisions
Presenting secure login options on insecure pages	47%	Embedding sensitive forms on insecure web pages
Contact information/security advice on insecure pages	55%	Not securing security-relevant context
Inadequate policies for user ids and passwords	28%	Hard-to-guess credentials
E-mailing security sensitive information insecurely	31%	Confidentiality

presence of the two strings "sending" and "e-mail". In order to reduce the number of false positives, we assigned values based on proximity. The closer the two sets of words, the higher the value or probability. A page needed to have an 85% probability in order to be included in our set. If a page satisfied this property, the website was considered to have this design flaw. We manually confirmed the presence of these statements for the sites that exceeded the 85% threshold.

5. RESULTS

With automated tools, such as the one used in our study, false positives are possible. To the extent feasible, we manually examined the results to eliminate false positives from the reported data. Our break-in-chain-of-trust data had a significant number false positives. Our automated tool reported about 30% of the websites to potentially use third-party sites in an unsafe way, but only 17% were found to do so without giving some sort of notification to the user about that transition.

Table 1 summarizes the list of specific design flaws that we looked for and the percentage of sites that were affected by those flaws. As is evident from the data, several of the design flaws are widespread. In particular, many financial sites were found to provide login boxes on insecure home pages. Less than half of the financial institutions bother to secure their customer service contact, password reset, and FAQ pages. On the positive side, most sites made an effort to provide good policies for user ids and passwords. We found 28% were lacking in that regard, with a larger fraction permitting easy-to-guess user ids, such as e-mail addresses or social security numbers. The use of third-party sites or a domain that is different from that of the home page was fairly common. Approximately 30% of financial institutions used multiple servers or third-parties to provide some security-sensitive services without informing the user about the transition to a trusted third-party site.

Figure 4 shows the number of design flaws existing on the financial websites. We found that it was common to see sites with multiple design flaws. 76% of the sites had at least one design flaw and 68% had two or more design flaws. 10% of the sites had all five design flaws.

We now consider some potential sources of errors in our data. It is likely that *wget* failed to completely retrieve all the pages of a website because of its inability to handle Javascript. *wget* also would not have been able to download pages that are served after a user authenticates, since *wget* could not authenticate as a customer at these institutions. With a partial graph, however, all of our results, except for the data on password policies, would be conservative estimates. For password recommendations, it is possible that *wget* failed to get some pages that may have contained policies on passwords, or that password policies were displayed only if the user selects a weak password. All other design flaws that we looked at have a *monotonicity* property; adding in additional nodes and links can only increase the number of flaws, not decrease them. For example, if a graph contains nodes that display security-sensitive content on a non-SSL page, adding in more nodes and edges will not make this flaw go away.

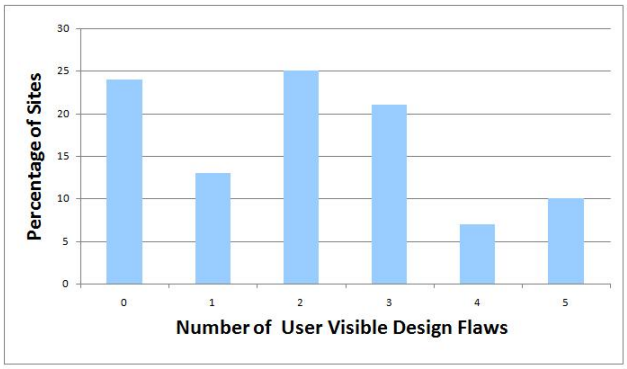

Figure 4: For the five design flaws discussed in this paper, this plot shows the percentage of 214 financial websites that had the given number of security design flaws.

Another potential source of error is that our choice of patterns may not have discovered all the problem pages. This would make our results conservative. Finally, there is a possibility of human error in our manual inspection process to eliminate false positives, though we hope the risk of that was minimal.

6. LESSONS LEARNED AND CONCLUSIONS

Our survey shows that most financial websites today are taking traditional steps for securing their websites, such as the use of strong credentials. Only a small fraction were found to not provide some sort of password policy or to rely on easy-to-harvest user ids, such as social security numbers or e-mail addresses. However, our work shows that most financial websites are not adequately protected against secure usability design flaws. These flaws can prevent even the most knowledgeable user from making proper security decisions. We found that 76% of sites have at least one design flaw. The pervasiveness of these flaws indicates that they are not well-understood by web security experts. Many sites continue to provide login windows on non-SSL pages. This is a problem because users have no way of knowing what will happen with their login credentials from examination of the browser's display. Many sites also provide security-relevant non-confidential data (such as Contact Information) on non-SSL pages. If those pages could be spoofed, users could be vulnerable to offline attacks that totally compromise long-term secrets, such as social security numbers, birthdays, account numbers, etc.

Our work also shows that the current set of web security analysis and design techniques still leave significant security gaps. In our discussion of methodology and results, we describe our approach for automatically detecting website secure usability design flaws. We recommend that web developers employ these tech-

niques when performing web security evaluations to prevent future websites from having the vulnerabilities we identify in this paper.

In the future, we plan on evaluating additional design flaws, such as password reset policies, that will require more sophisticated policy analysis. In some cases, automating interpretation of natural-language security questions may be necessary.

7. REFERENCES

[1] Banking study: list of financial institutions. http://www.eecs.umich.edu/∼laura/webusability/websites.html.

[2] L. Cranor, P. Guduru, and M. Arjula. User interfaces for privacy agents. *ACM Transactions on Computer Human Interaction*, 12(2):135–178, 2006.

[3] R. de Paula and et. al. Two experiences designing for effective security. In *SOUPS '05: Proceedings of the second symposium on Usable privacy and security*, New York, NY, USA, 2005. ACM.

[4] R. Dhamija, J. Tygar, and M. Hearst. Why phishing works. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2006.

[5] J. S. Downs, M. B. Holbrook, and L. F. Cranor. Decision strategies and susceptibility to phishing. In *SOUPS '06: Proceedings of the second symposium on Usable privacy and security*, New York, NY, USA, 2006. ACM.

[6] D. Florencio and B. C. Cormac Herley. Do strong web passwords accomplish anything? In *Proceedings of the USENIX Workshop on Hot Topics in Security (HotSec)*, 2007.

[7] L. Freed. State of customer satisfaction with online banking, forsee results/forbes.com, April 2007.

[8] K. Fu, E. Sit, K. Smith, and N. Feamster. Dos and don'ts of client authentication on the web. In *Proceedings of the 10th USENIX Security Symposium*, Washington, D.C., August 2001. An extended version is available as MIT-LCS-TR-818 (Best Student Paper Award).

[9] Banking on the www - banks of the usa. http://www.quazell.com/bank/bank_usa.html.

[10] P. McDaniel. On context in authorization policy. In *Proc. of the 8th ACM Symposium on Access Control Models and Technologies (SACMAT)*, pages 80–89, June 2003.

[11] Nessus Vulnerability Scanner. http://www.nessus.org.

[12] B. Pinkas and T. Sanders. Securing passwords against dictionary attacks. In *ACM CCS*, 2002.

[13] N. Provos, D. McNamee, P. Mavrommatis, K. Wang, and N. Modadugu. The ghost in the browser analysis of web-based malware. In *Proceedings of the USENIX Workshop on Hot topics in Understand Botnets (HotBots)*, 2007.

[14] S. Schechter, R. Dhamija, A. Ozment, and I. Fischer. The emperor's new security indicators: An evaluation of website authentication and the effect of role playing on usability studies. In *IEEE Symposium on Security and Privacy*, 2007.

[15] S. E. Schechter, R. Dhamija, A. Ozment, and I. Fischer. The emperor's new security indicators. In *SP '07: Proceedings of the 2007 IEEE Symposium on Security and Privacy*, pages 51–65, Washington, DC, USA, 2007. IEEE Computer Society.

[16] D. Wagner and B. Schneier. Analysis of the SSL 3.0 protocol. In *Proc. of The 2nd Usenix Workshop on Electronic Commerce*, Nov. 1996. Revised April, 2007.

[17] WatchFire's AppScan Product.

[18] A. Whitten and J. D. Tygar. Why Johnny can't encrypt: A usability evaluation of PGP 5.0. In *8th USENIX Security Symposium*, 1999.

[19] M. Wu, R. C. Miller, and S. L. Garfinkel. Do security toolbars actually prevent phishing attacks? In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 601–610, New York, NY, USA, 2006. ACM.

[20] Why Use YURLs?, 2003. http://www.waterken.com/dev/YURL/Why/.