Security in Practice: Security-Usability Chasm

Atul Prakash

Computer Science and Engineering Division,
University of Michigan, Ann Arbor, MI 48109, USA

Abstract. Computer systems security area has received increased attention from both academics and in industry. However, recent work indicates that substantial security gaps emerge when systems are deployed, even with the use of state-of-the-art security protocols. Our findings suggest that wide-spread security problems exist even when protocols such as SSL and SSH are deployed because systems today do not give security warnings properly or make it trivial for users to bypass them. Even when these protocols are deployed correctly, systems often leave themselves vulnerable to social-engineering attacks as an artifact of their design. In one of our studies, we examined the web sites of 706 financial institutions and found over 90% of them to have made poor design choices when it comes to security, even though all deployed SSL for communicating passwords and doing transactions. In another study, we examined the usage of SSH within our own department and found that most users would be susceptible to a man-in-the-middle attack. Based on our studies, we postulate that some of the most interesting challenges for security researchers and practitioners lie at the intersection of security theory, their application to practice, and user behavior. We point out some of those challenges and hope that the research community can help address them.

1 Introduction

Online activities become an integral part of our day to day life. For example, in the financial world, many people have given up conventional check writing in favor of online banking. Brokerage sites are available for people to buy and trade stock online. Financial institutions have a substantial stake in securing their customer accounts. In order to protect customer accounts, financial institutions deploy latest security technology, such as SSL/TLS [1] (originally developed by Netscape), and employ security specialists to design their systems. Organizations are also increasing deploying security protocols, such as SSH [2], to prove remote secure access to their servers, in place of insecure protocols such as telnet.

A question that arises is whether the current practices for deployment of security solutions are adequate. We have done some preliminary studies that suggest most systems today continue to be vulnerable to network and spoofing attacks, even when they deploy secure communication protocols. This is true for both systems in academia as well as in the commercial world. Most of the problems that we identified are at the intersection of usability and security, but some appear to be due to insufficient appreciation of security risks beyond eavesdropping of passwords or sensitive network communication. We therefore submit that research needs to take place at the intersection of security principles and its practice at the end-user level to address security gaps that arise at the end-user level. In my talk, I plan to present some of our findings and hope that the research community as well as the commercial world will explore solutions to addressing those gaps.

We carried out two studies. The first study [3] examined the usage of SSH within our department and the second study [4] analyzed the design of web sites of financial institutions in the United States from a security perspective. We will not go into the details of the studies in this paper, but just highlight some of the security gaps that were identified by these studies. Besides our studies, other studies [5, 6] also identified gaps in security in real-world systems due to inadequate interface design. Our studies highlight gaps even for sophisticated users who understand security.

In the next section, we discuss the results from two of our studies. In Section 3, we present some challenges in IT security in developing countries such as India that will be increasingly pushing information technology use to people in rural areas. In Section 4, we discuss some of the lessons learned and some open challenges. Finally, we conclude the paper.

2 Results from Studies

We briefly describe preliminary results from two on-going studies of vulnerabilities at the user level. The first study examined the vulnerability of users to man-in-the-middle attacks on SSH on our department's machines. The users in this study can be considered to be fairly sophisticated users since they mostly consist of computer science graduate students and some faculty. The second study analyzed over 700 web sites of banks for a range of vulnerabilities that resulted from poor web-site design decisions from a security perspective, as opposed to vulnerabilities that result from poor coding practices (which lead to vulnerability to attacks such as SQL injection or cross-side scripting) or incorrect configuration of SSL.

2.1 Vulnerability Assessment of an SSH Deployment

SSH is considered to be a well-designed protocol from a security perspective and is widely deployed. When used properly, it is designed to provide protection against both passive and active network attacks. One limitation of SSH is that a client must have the server's public key *a priori*; SSH does not provide secure support for key distribution. However, implementations of SSH do provide error messages and, if so configured, terminate the connection if the destination host is unable to present proof of possession of the server's private key.

We monitored SSH logs to analyze user behavior when our system administrators changed the SSH host key on a popular server within our department. The server's public key had remained static for over two years and thus expected to be installed at most user's machines. Over 70 users attempted to login over the server after the key change during the monitored period. We found that less than 10% of the users asked the administrators if there was a key change and none verified the actual key. Rest of the users decided to log in, ignoring the warnings from SSH. This was true for even users whose connections were terminated by SSH (this happens in SSH in "strict checking" mode). Almost all of these users logged in later – presumably after deleting the offending server's host key from their configuration file, typically *˜/.ssh/known_hosts* on non-Windows hosts, so that it could be replaced automatically on the next SSH login.

We also analyzed the SSH logs of users who had opened accounts in the last three months on the server and inquired as to if any of them had asked our administrators for the server's host key. None had. All users accepted the initial key that was provided by the server.

An interesting question is why did users in our study make poor security choices when using SSH? One could potentially say that they were ignorant of risks, but we do not believe that to be the case. Most of the users in our study were graduate students and faculty in Computer Science and were sophisticated enough to get around SSH warnings by removing the offending keys from *˜/.ssh/known_hosts* file. Furthermore, SSH gives pretty explicit warnings about the risks of establishing the connection. We do not have data to answer this question conclusively, but we did talk to a few of the users, including some we knew were aware of the risks (because they have done research related to security).

Based on our conversations, we came up with a few possible factors. Some users may simply be using the *path of least resistance*. They perceived the SSH error to be an obstacle to connecting to the server. The simplest solution to get around the problem was to delete the server's public key rather than contact an administrator to verify the key. This suggests that the simplest path to getting around a security error for a user should be carefully secured, rather than being treated as a rare anomaly that is not particularly interesting from security perspective. SSH does not do that well. SSH warnings do suggest to contact the administrator, but they do not say how or what to do if the administrator is not available (e.g., late at night). Thus, the easiest path for a user is an alternative one – delete the offending key and accept the new key.

Unfortunately, other systems, such as web sites, do not fare much better either in handling security errors. Many sites will send temporary passwords via email, for example, when users forget their password. The assumption is that the path from the server to user's email account is secure (or someone else's problem). But we security researchers know that is not necessarily the case. The last hop to the user is perhaps the most vulnerable link because a user could be connected to an insecure wireless network and accessing his email account. Most web-based email systems today (e.g., gmail, yahoo) authenticate users over SSL, but deliver email in the clear. A variety of other solutions for handling forgotten passwords are available, some better than this. But there is no clear standard or guidelines from the security community for the best way to address this problem, even though it occurs so frequently in practice, which is an astonishing gap.

Another factor could be that users get frequent security warnings from SSH when logging to different hosts around the campus that they tune out the warnings and consider them as essentially a nuisance from SSH. Lack of consistency in user experience to security-related events can really hurt system security. Unfortunately, our study shows that it does not help for administrators of one machine to provide a consistent experience. In our study, SSH host keys did not change on the server for over two years. But users do get SSH warnings from other machines, including login server pools, and when logging in from new machines. We believe that this may be diluting the impact of a security warning even from a machine that gives warnings rarely.

In our SSH user study, we found that most users bypassed SSH warnings even though the host key was changed after a period of two years. As we mentioned earlier, one problem is that standard SSH warnings tell you that there could be an active attack going on, but not how to get around the problem and accomplish the work. Not connection to the server, as the warnings suggest that the users do, is not really an option for most users because they need to get their work done. Instead, if the warnings provide clear instructions for verifying the new key (perhaps by providing a secure URL to a well-known, standardized site), more users may have followed those instructions. Of course, it is possible that the instructions themselves could be tampered with by an attacker, leading to an interesting challenge. In SSH, the key distribution problem is simply left as an assumption for security. Unfortunately, no standard ways have evolved for guiding administrators on how to best address it.

Though our study is limited to only one department and one type of population, it strongly suggests that SSH deployments fails to provide protection against man-in-the-middle attacks at the user-level, even though the protocol itself may be designed to prevent man-in-the-middle attacks when correctly used. We note that if a user uses password-based authentication in SSH, they are susceptible to a man-in-the-middle attack. It would be safer for a user to use public-key based authentication in SSH (using passphrases). In that case, an attacker could spoof a server, but not login as the user to the real server. In our study, we found that most users did not use passphrases though that is an available option. Perhaps, passwords are more convenient and provide a path of less resistance than passphrases.

It is possible that the users made, what they considered to be, a rational decision. They perceived the risk of an actual man-in-the-middle attack to be low and the possibility of a configuration change at the server higher. But that is disconcerting because it means that users will discount security risks and will use unsafe methods, if given a choice.

SSH designers made the choice to trusts the users to make the right security decisions. Most SSH clients, when seeing a bad server's key, very clearly tell a user not to connect and to verify the server's fingerprint with an administrator because they may be subject to a serious attack that will totally compromise the session's security. But it also tells them that they can get around the problem by deleting the cached server's key. Our study indicates that most users will choose the latter option, raising the question whether even sophisticated users should be trusted by implementers of security systems to make security decisions.

2.2 Vulnerability Assessment of Financial Websites

In [4], we analyzed over 700 web sites of financial institutions with presence in the United States. The main aspect we were looking for is how these institutions structured the online interaction with their customers from the perspective of security vulnerabilities. The good news is that almost sites use SSL for authenticating their web server to users and do not send passwords in the clear. The bad news is that most of them still make poor design decisions at the macro level that leave their users more vulnerable to spoofing, phishing, and social engineering attacks.

Below are two representative examples of poor design decisions that many of the financial web sites suffered from (several other vulnerabilities can be found in [4]):

1. Providing a secure login window, but on an insecure page
2. Insecure pages for customer service and instructions for password reset

Why do these decisions get made and why are they poor decisions from a security perspective? Let us consider each of the two examples in turn. Here is a plausible chain of events between an hypothetical bank's user-interface design group and its security group that lead to the first decision:

1. Providing a login window on the home page is convenient for users, as per the user-interface team. It saves users one click in going to a separate login page and we know that the number of clicks matter, when it comes to ease-of-use.
2. The security group rationalizes that the userid and password is being submitted securely over SSL. Therefore, the credentials are safe against spoofing and network attacks.

Thus, it is plausible that the software group at the bank rationalizes that the design decision of providing the login window on an insecure home page is reasonable because the Javascript code that handles the login window submits the userid and password over an SSL-protected session. SSL guarantees security (assuming that SSL is secure), therefore the login process is secure.

Unfortunately, the reasoning is faulty. From a user's perspective, they do not know if the login window form is genuine or not, since it is on an unauthenticated page. Thus, the login credentials must be entered based on blind faith that the login page is not spoofed and not tampered with, and that the financial institution has taken precautions to submit the information securely.

I recall reporting this problem to my brokerage company (which I shall leave unnamed) in United States several years ago when they used to provide a login/password window on an insecure home page (though the credentials were submitted over SSL). The response was that if I was concerned, I should just click the login button without entering the user id and password. That will take me to an SSL-protected page, where I could enter the correct user ID and password. I would argue that it it unreasonable to expect most users to be aware of such a workaround and most users will not use it anyway since it is not the path of least resistance. To that company's credit, at least they had a solution for me and eventually they modified their login procedures so that they are no longer vulnerable to that problem. Many other financial institutions, however, continue to take that risk.

Now, let's examine the second problem of providing contact information on an insecure page. One of the institutions that holds retirement accounts for faculty at my university, TIAA/CREF, was guilty of that. Why is this a poor security decision? Let us assume that part of the threat model is a man-in-the-middle attack or a site spoofing attack. That is one of the reasons why the use of SSL for protecting communication related to authentication and transactions makes sense. Suppose that an attacker redirects the user to a spoofed "Contact Us" page that contains bad phone numbers for TIAA-CREF that are actually manned by the attacker. If a user were to use those phone numbers for account queries, it is conceivable that user would be asked to provide credentials, such as social security number and birthdate, to verify their identity and would simply end up giving those to the attacker.

I initially suspected that the problem was primarily due to TIAA/CREF not being aware of the risk. They could easily have provided the information on an SSL-protected page. Not using SSL for all pages does potentially reduce the load on the web server, but I doubt that protecting the "Contact Us" page with SSL, or even the whole web site, would be a major expense for a large company that handles billions of customer dollars. I did report the problem to them but, at the time of this writing, they continue to provide contact information on a non-protected page. Hopefully, they are working on it. Hit the "Contact Us" link at their site to see if they have since fixed the problem.

A few financial web sites have addressed this problem. For example, Fidelity brokerage uses SSL to protect their entire site. However, a vast majority of sites remain vulnerable today.

3 Broader Problem of Authenticating Users

So far, we only touched on the problem of verifying identity in the context of an authentication protocol. But, there is a more general problem that has broader implications. In the United States, social security numbers serve as universal, unique identifiers for individuals and form the basis of establishing credentials for other services such as bank accounts and credit cards. But, as is well-known, the use of a single identifier also increases the risk of identity theft. In developing countries, such as India, most citizens do not have a unique ID assigned at present. Tax IDs and voter cards are being increasingly assigned to individuals, but they are not yet universal. So, there is a window of opportunity to recommend better systems if the security community can come up with them.

Another interesting problem in a country like India is that of identifying people over the phone. A large percentage of people do not speak English, the country is multi-lingual, and as a result the mapping to a bit string that represents a name in an authentication database is ambiguous. Other attributes, such as home address, are also ambiguous because many addresses, particularly in rural areas, are simply a collection of attributes that describe the place sufficiently for the local mail delivery person; a classical example given in India is an address specified by just the recipient's name and a nearby local landmark - presumably the postal carrier would know the person and how to deliver the letter). This raises the question how to do verification. For example, if a user gives an approximately matching name and address (rather than an exact match), should that be accepted for locating a record? Would it reduce the "number of bits" in the verification process? I conjecture that it does, since it is akin to doing an approximate match on passwords, rather than an exact match. But analyzing the extent of its impact on authentication systems requires further research.

4 Lessons Learned

Fortunately, the situation is not entire hopeless. Our studies suggest a few design principles that may help in securing systems as well as providing good usability, though their validation requires further work:

End-user visibility of security guarantees : System designers must not only secure the network communication channel, but also ensure that the end-user application is able to make that security visible to the end-user before the user uses that channel. Ideally, this visibility of security attributes to the end-user should be tamper-proof for the desired threat model and be presented in a way that the user is likely to understand and notice. Unfortunately, providing good user presentation of security guarantees is easier said than done. The work by Cranor et al [7] shows that it can be a significant challenge in designing interfaces that present P3P privacy policies of web sites to users so that users are likely to notice them and be able to use them for making informed security decisions. The work in [5] highlights similar problems in web browsers for presenting SSL security context. Providing a tamper-proof way of displaying security guarantees is also a challenge. For example, Javascript could be used to control the contents of the displayed URL at the bottom of a page when a link is highlighted. Many users rely on that to see if a link should be trusted prior to clicking. Browsers are doing a better job today of preventing scripts from modifying the displayed security context, but it is still an arms race.

Secure the context : This is a well-known principle that not only communication must be secured but also its context [8, 9]. For example, SSL 2.0 was vulnerable to cipher rollback attack because it did not adequately protect the key negotiation steps. SSH fails to do that for host keys. Unfortunately, most administrators do not adequately tell users how to verify the host keys in a convenient way, leaving users vulnerable to security attacks. Providing a login window (even that is submitted over SSL) on an insecure page is also an example of this problem; it would be easy for the attacker to replace the page contents so that the information is submitted insecurely. For a bank to provide contact phone numbers on an insecure page is also an example of this problem.

Secure the path of least resistance when errors occur : Security errors may be rare (most logins are successful), but the implications can be significant if those errors are not correctly handled. A careful analysis of users' likely responses to errors and the security implications of that must be done and standards evolved for handling them safely. That analysis should assume that most users will choose the simplest path to getting around the problem rather than the most secure path.

Security warnings should be rare events: In our SSH study, it is conceivable that users get SSH warnings about bad or missing host keys so frequently that they have come to accept it as "normal", rather than a possible security risk. Organizationally, system administrators should make it a high priority to not change host keys except when a key is suspected to be compromised. For example, if a server is upgraded (e.g., new computer replacing an old one), it would be better to transfer the old machine's key rather than generating new keys. It would be even better to consider standard extensions to SSH to support signed host keys (using long-term certificates of organization's authorities). That may help make the warnings rarer when machines are upgraded and new host keys are installed.

End-user verification of security-related information: If a protocol gives security warnings with information such as fingerprints, the user must be told how to verify them. SSH fails in that regard. It does print out the fingerprint of the host's key. However, most users are unlikely to be aware how to verify that fingerprint.

5 Conclusions

We presented some of the findings from two of our studies that suggest existence of wide-spread security problems at the end-user level in systems that attempt to deploy security protocols to secure user interactions. Our findings suggest that significant security gaps are introduced at the application level, even in systems where security is the primary concern and users can be considered to be sophisticated about security. This points to a significant challenge for the security community. The community must not simply ignore this issue, leaving it to application writers or user-interface designers to address. We must develop security-oriented guidelines for application developers as well as automatic security modeling and evaluation tools to assist the developers in assessing the security of their systems. The guidelines and tools should factor in results from existing studies on user behavior.

Acknowledgments

The results from studies cited in this paper would not have been possible without the help of graduate students, Brett Clippingdale and Laura Falk. Brett was instrumental in collection and analysis of results on SSH usage. Laura Falk was instrumental in writing scripts to help analyze security at bank sites. I also thank the organizers of ICISS 2007 for the opportunity to present this work.

References

1. Rescorla, E.: SSL and TLS: Designing and Building Secure Systems. Addison-Wesley (2000)
2. Ylonen, T., Lonvick, C.: The secure shell (SSH) protocol architecture (Jan. 2006) RFC 4251, IETF draft.
3. Clippingale, B., Prakash, A.: Usability vulnerabilities in SSH: When good users go bad. unpublished manuscript. Contact author(s) for a copy (September 2007)
4. Prakash, A., Falk, L.: Web security analysis of financial institutions. Technical Report CSE-TR-534-07, Department of EECS, University of Michigan (2007)
5. Schechter, S., Dhamija, R., Ozment, A., Fischer, I.: 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)
6. Whitten, A., Tygar, J.: Why Johnny can't encrypt: A usability evaluation of PGP 5.0. In: Proc. of the 8th Usenix Security Symposium. (1999)
7. Cranor, L., Guduru, P., Arjula, M.: User interfaces for privacy agents. ACM Transactions on Computer Human Interaction **12**(2) (2006) 135–178
8. Wagner, D., Schneier, B.: Analysis of the SSL 3.0 protocol. In: Proc. of The 2nd Usenix Workshop on Electronic Commerce. (Nov. 1996) Revised April, 2007.
9. McDaniel, P.: On context in authorization policy. In: Proc. of the 8th ACM Symposium on Access Control Models and Technologies (SACMAT). (June 2003) 80–89