2010 14th Panhellenic Conference on Informatics

When Security Meets Usability: A User-Centric Approach on a Crossroads Priority Problem

Christos A. Fidas
Human-Computer Interaction Group
University of Patras
Patras, Greece
e-mail: fidas@ece.upatras.gr

Artemios G. Voyiatzis
Industrial Systems Institute
Research Center "ATHENA"
Patras, Greece
e-mail: bogart@isi.gr

Nikolaos M. Avouris
Human-Computer Interaction Group
University of Patras
Patras, Greece
e-mail: avouris@upatras.gr

***Abstract*— Effective and efficient methodologies are essential for developing and maintaining information systems that are both secure and usable, especially in the case of Internet applications that require a combined effort from application, system, network, security, and human-computer interaction design areas. In this paper, we propose a user-centric approach towards achieving "usable security". As a case study we apply the proposed approach on the password management problem. Our work demonstrates that the "user-centric usable security" is a viable and promising future research direction.**

Keywords-security; usability; HCI; user-centric design

I. INTRODUCTION

The ever increasing dependence of the developed world on digital transactions for everyday life drives the need to streamline efforts on designing systems that are both secure and usable. In the era of computing, digital communication, and e-commerce, the security is still considered as orthogonal to usability in too many cases. A common statement is: "if it is secure, then it is not usable" e.g., a secure password is hard to remember and an easy-to-remember password is not secure. In this context, there is an increasing agreement lately on the need to design secure and usable computer systems. Interdisciplinary research on the emerging field of "HCI-SEC" (Human-Computer Interaction and Security) brings together researchers of the two fields towards developing "usable security".

In this paper we argue that system designers involved in "usable security" are at a crossroads. They try to reach a difficult compromise: provide a highly usable system to the users and at the same time protect the assets of the provider, even by exposing, sometimes, the users to security threats. This commonly applied approach results often in overloading the user with not task related feedback, as priority is given to protecting the provider over the users. The users perceive security as someone else's problem and due to this, make poor decisions related to security issues, considering them irrelevant to their main tasks. This approach introduces more security problems to them and the system. The loop closes as the designers introduce more security features in order to protect the system from the users. The net outcome is that designers keep developing a system with security for the provider and usability for the user. Thus, security reaches the user as an off-task issue, as a side-effect, and it is not convincingly considered as "usable security".

A possible method to overcome this still open and important issue is to apply a user-centric approach towards developing information systems with "usable security". User–centric approaches are widely used, especially in cases where user requirements are difficult to elicit and to understand. These approaches put the end-users at the center of the software development cycle and aim to achieve a common understanding among the designers and the users, related to the software under development. The objective is to reach a certain point in which the designers and the users share a common conceptual ground related to the developed system e.g. sharing a common mental model of using it. Mental models have been studied by cognitive scientists as part of efforts to understand how humans know, perceive, make decisions, and construct behavior in a variety of environments. A mental model is an internal scale-model representation of an external reality e.g., the information systems and their functionalities. It is built on-the-fly, from knowledge of prior experience, schema segments, perception, and problem-solving strategies but it also maintained and refined [15].

Therefore, our proposal is to apply a user-centric approach in designing information systems with "usable security". Our aim is to make users perceive security as an important part of their interaction process by creating and maintaining or refining appropriate mental models of interactions. Thus, both usability and security are focused on user needs; as protection of the providers' assets comes second to them.

The paper is organized as follows: In Section II, we review existing literature in the area of HCI-SEC. We

978-0-7695-4172-3/10 $26.00 © 2010 IEEE
DOI 10.1109/PCI.2010.17

develop our theoretical framework and problem formulation in Section III. We propose a user-centric approach for solving the problem in Section IV and validate our proposal in Section V by describing how it can solve the password management problem. Finally, we reach our conclusions and describe promising directions of future work in Section VI.

II. USABLE SECURITY

Since the early days of computing history, access to computing facilities was considered as a privilege that must be controlled. It was a natural approach given the aims of building such computing machines those times. Even when these facilities became more widely available, funding stakeholders and managers insisted on policies and mechanisms for controlling access to their investments [1].

Since then, security has evolved under the assumption of protecting the system resources (and later on the network too) from outsiders and misbehaving users. Over the time, computing devices and networks witnessed wide acceptance and evolved into everyday life tools for ordinary users to perform their job at hand with little or no training at all.

Since the late 1990s and Internet's exponential growth, there is an increased interest in designing systems that can be both secure and usable. Such designs must take into account the user needs [2]. However, early evaluation of secure applications showed a significant usability gap for the average user [3]. The emerging area of HCI-SEC studies the interrelated security and HCI requirements of system and application design. The Computing Research Association identified HCI-SEC as one of the "Four Grand Challenges in Trustworthy Computing" in early 2000s [4].

The literature review reveals four areas of interest at the crossroads of HCI and security: i) user authentication, ii) secure interface design, iii) usability of security products, and iv) protection of systems using CAPTCHA techniques. We review work in these areas in the following.

A. Passwords, Secrets, and other Forms of Authentication

Passwords, passphrases, and personal identification numbers (PIN) are used widely for user authentication and access control in computer systems. They are also generating the most frustration to both users and system owners. Users prefer easy-to-remember passwords that can be reused in many places [5]. They are disturbed when system policies do not accept their preferred password or oblige them to change it frequently. Users tend to forget their passwords. They require help to recall or reset it, either from other humans or by writing them down in paper or in notes next to the screen.

System owners and managers are passionate about their systems and the value they bring. Thus, they enforce policies that ensure maximum protection of their assets. Being technology savvy, it is hard for them to tolerate poor password selection and to understand users' need to write down the secret passwords as not to forget them.

The password problem is often used as an example of the opposition between security and usability: security dictates hard-to-guess passwords while usability dictates easy-to-remember passwords. We shall formulate this problem in the next section and further show that this is not an opposition between security and usability but rather between users and providers.

B. Interface Design and Security Indicators

Electronic mail protocols were not designed with security in mind; the initial goal was to allow fast communication of messages over the Internet infrastructure. The commercialization of Internet results in the need for encrypting or signing electronic messages (or both). Whitten and Tygar experiments in 1999 [2], reveal that even educated and experienced email users did not manage to send a secure email within 90 minutes of time. A repeated study by Garfinkel and Miller in 2005 [6], highlighted the continuing problems even after interface improvements.

Masone and Smith [17] note two more problems that users face on secure email applications based on digital certificates, certification authorities, and public key infrastructures, due to contradicting mental models in real and digital life: a) should the recipient doesn't have installed the CA certificate of the sender, the interface dictates not to trust the sender although trusted in real life and b) the interface dictates the user to trust any signed message regardless of who certifies the sender thus, enforce trust even for people the recipient does not trust in real life.

Electronic commerce services mix real life activities with digital transactions, mainly of monetary value. Book or music purchases through Internet and online payments of bills and taxes are typical examples of such a mix. Even with strong (financial) incentives, users tend to ignore security indicators, such as absence or invalidity of SSL certificates [7]. However, the average user cannot easily detect and understand this practice and its consequences [8]. Finally, current trend of outsourced web services results in users being redirected to third-party sites for completing payments or other sensitive operations. This further hardens user efforts to build trust on a web site and to comprehend the involved security risks [9].

A common metaphor from the physical world is the notion of letters, documents, and books, mirrored to emails, e-document, and e-books. In the physical world, all these are just paper and ink, a harmless combination. In the digital world these objects are actually active, executable code augmented with rich user interfaces and advanced software capabilities. The misconception presented to the users may result in harming users and in privacy invasion [9].

Built-in security is considered as a convenient approach, since it can hide all the complexities of security from the user. Eventually, the user has to interact with the security options of an application. Thus, at least one more technique, like tutorial or context-sensitive help, is needed [16].

C. Usability of Security Products

The usability of security products, such as firewalls and antivirus software, requires special attention from both usability and security point of view. There is an inherent difficulty on implementing security correctly in the first place; a deep understanding of the internals of a system is needed to make wise and safe choices. However, ordinary users are expected to actually tackle security issues to protect

both the system and themselves. Typical applications tend to seldom ask for a user decision on a security issue. On the other hand, security software requires continuous vigilance from the users; they are almost asked to become security experts themselves. Each decision must be correct straight from the first time or else users can compromise their privacy and system's security [16].

Johnston et al. are proposing six design criteria that once met lead to "trust" [10]: convey features; visibility of system status; learnability; aesthetic and minimalistic design; errors; and satisfaction. Trust is one of the ten usability criteria for a successful HCI according to Jacob Nielsen [11]. Herzog and Shahmerdi [16] are concluding on four usability requirements for security applications or security features of applications i) they make users reliably aware of the security task they need to perform, ii) they make users able to figure out how to successfully perform those tasks, iii) they do not allow users to make dangerous errors, and iv) they make users sufficiently comfortable with the interface as to continue using it. These requirements can be considered a condensed form of Yee's ten design guidelines for secure interface design [18]: Path of least resistance, Active authorization, Revocability, Visibility, Self-awareness, Trusted path, Expressiveness, Relevant boundaries, Identifiability, and Foresight.

D. Usability of CAPTCHA

"Completely Automated Public Turing Test to Tell Computers and Humans Apart" (CAPTCHA) refers to a software that generates and verifies challenge puzzles or tests that a computer cannot solve automatically but a human being is able to. In general, the aim of CAPTCHA software is to minimize intentional or unintentional denial-of-service attacks on heavy-use web sites and applications that are caused by massive, rapid, automated service requests. For example, CAPTCHA can be used to stop a bot registering for accounts in free hosting services or to delay login retries for breaking into a password-protected web site.

The usability of CAPTCHA receives lately attention, since it gains wide adoption and requires active user participation and cooperation. There are three types of CAPTCHA, based on the challenge they present: text, audio, and image. Yan and El Ahmad [14] provide evidence that non-Latin alphabet users can have a hard time solving CAPTCHA puzzles. Further, they find that the length of challenge strings and the colors used on the challenge (background and characters) can have impact on usability, on security, or on both of them. Hernandez-Castro and Ribagorda [12] conclude that CAPTCHA lacks "a methodological analysis and design of the schemes placed in production".

III. THE CROSSROADS PRIORITY PROBLEM

The analysis in the previous Section leads to a problem formulation layered in two levels. At the first level, we argue that the system designers are facing contradicting requirements but not on the traditional form. From one side, system stakeholders are concerned about the security of the system and are willing to sacrifice the convenience of the users in order to achieve system security. For example, considering the problem of CAPTCHA, this is not a problem of the user, but of poor system engineering; the system should be engineered in such a way that it should not allow massive registration in a short time anyway. However, the problem is passed back to the users, hampering the usability of the system by requiring them to solve puzzles.

On the other side, users are interested in the usability of the system. Current engineering practices result in adding security as an extra layer in the design and not in incorporating the security in the whole software design process. Additionally, from a usability point of view, security indeed looks like an add-on: something added later on to solve somebody else's problem, as the system is almost always usable even without the security layer. As a consequence, the security does not feel "natural" to the application.

At the second level, security mechanisms are closer to technical issues rather than user needs. This results in many misunderstandings and in confusion for the user. There are proposals for "invisible security" i.e., implementations that hide the security from the users. However, it is not possible to hide each and every security parameter from the user. The aforementioned argument is stronger for security products, like firewalls and antivirus software, which need to be adapted to user needs and preferences. In this case, the problems are: i) the security terms are not clearly communicated to the users, ii) interfaces do not explain the problem the users may face by choosing each alternative option, and iii) explanatory texts lack clear statement and are vague and ambiguous on purpose, in order to cover each and every possible (or impossible) case. If the system designers cannot estimate the risk of a security decision, it often suffices for them to use vague statements like "action may result in loss of information". It is however unfair to expect from a casual user to understand the risk involved, evaluate it, and reach a conclusion. The system must take stand and actually explain to the user the involved risks; the default action must be the one that most protects the users rather than exposes them to threats.

Garfinkel and Spafford [13] give the definition: "A computer is secure if you can depend on it and its software to behave as you expect". This definition embraces the relationship between the security properties of a system, the users and the context of use. As a consequence, designing for "usable security" turns to be a problem that needs to be customized for a specific group of users and within a certain context of use. We argue that "usable security" must not only refer to good interface design but must include the creation of system mental models that embrace security as an essential aspect of user interaction.

The crossroads priority problem can be stated as follows: "how can we establish information system design processes that lead to secure and usable user interfaces designs in the cases that security must be communicated to end-users?" We believe that user-centered design approaches can solve the problem. Our approach differentiates and complements the ideas presented so far since the security focus is not on assisting the user to protect the system but it is rather on

protecting the security of the user. We further elaborate on this view in the next sections, using the example of users selecting, using, and changing passwords.

IV. A User-Centric Approach

We propose a user-centric approach to the design of secure and usable security; the security must primarily address user needs and aims rather than system's or developer's needs. This user-centric approach urges security and usability experts to work closely from the early phase of system design, aiming at presenting security in a clear and concise manner.

User-centered design approaches focus on involving the end-users in the process, especially for identifying and validating user requirements as well as for evaluating system prototypes. They aim to investigate thoroughly users' views and how the system can support them in accomplishing specific tasks effectively, efficiently, and with a certain degree of user satisfaction.

An important aspect of this process is to model user interaction with the user interface. A good design practice aims to establish a common ground between designers and users, related to aspects of user-system interaction. A well-used and simple approach to modeling interactive systems is to describe the stages of actions users go through when performing a certain task. According to Norman [19], there are seven steps in a typical interaction cycle with a system: form the goal and the intention, specify and execute the action sequence, perceive and interpret the system state, and finally evaluate the state with respect to the goals and intentions.

First, the users form a conceptual intention from their goal (example: the user wants to access her e-banking account). Second, they try to adapt the intention to the commands provided by the system (example: explore the Web page to figure out how to realize the intention) and from these, user-perceived, commands the users carry out the action. Then, the users attempt to understand the outcome of their actions by evaluating the system response. The last three stages help the users develop their idea of the system. The whole process is repeated in cycles of action and evaluation. The users refine the model of the system they have in mind by interpreting the outcome of their actions.

An important aspect of user interface design related with security issues is whether security is a primary goal of a user or a side effect of the task. For example, if the primary goal of the users is to access their e-banking account, then security is a side effect; if the users are configuring their firewall, then security is their primary objective.

The users focus on primary than on secondary goals. Thus, user interface designers should not assume that users will put much cognitive effort in understanding security issues. Users assume that the security is not related to their task but is rather a system-oriented and automated process. As a result, from a user point of view, security is perceived more as system problem rather than as an application design issue which involves user contribution as well.

Poor practice of "usable security" ignores the fact that users develop mental models related to their interaction with the system. Once these models are created, then the users interact with the system in more automated ways, faster, and more efficiently. Changing these interactions because of security concerns, often forces the users to follow erroneous interaction sequences. These often entail more security risks than those which they wanted originally to avoid. For example, many e-banking applications force their users to change once in a while their password for security reasons. This results in a high percentage of users not being able to login the next time since their mental models continue to use the old password associated with the system of use. A natural reaction is that users write down their passwords on files or stickers as they cannot remember their current active password. Another similar example is the "keep me logged in" functionality offered by many web applications. The users remain automatically connected for a certain period of time. Then, suddenly and without any prior notice, they are disconnected and forced to connect again. This is also a poor design in terms of "usable security".

V. A User-Centric Example on Password Change

Password management is considered for a long time now as the cornerstone of usable security. There have been enormous efforts to educate users in choosing and using a secure password. However, users keep on using the same, simple password over and over again.

Best practices for security, propose users to change their password often in order to minimize the possibility that someone with enough time and resources breaks into their account. Until now, the most common system engineering approaches on the password change problem involve one of the following two options: either (i) never notify the user about the risk or (ii) suddenly force the user to choose a new password, based on some policy, like not allowing the old password or recent passwords to be reentered. We call these two options "always off" and "instant on" respectively. While institutional policies may dictate ahead of time the password change frequency (e.g. once a month), we still consider it an "instant on" case, since the system does not notify the users in advance for the upcoming action but suddenly notifies them for password change.

We can now discuss how the user-centric design approach can solve efficiently this problem. A user-centric approach incorporates the password change as a regular user task throughput the usage of the system. We consider this as an "always on" approach. This approach can be realized as follows, for an example web application.

Upon creation of an account, the application provides a tooltip on password selection and allows the users to choose any password they like. User interface elements like "tip of the day" regularly provide user insights to users on current system password policies, like how many days are left to change password, and on the advantages for the user on periodic password change and choosing strong passwords. This approach achieves three objectives: a) allows the user to directly start using the system, b) educates users over time on the advantage of frequent password change and on current system policy, and c) puts the user on a mental model that

password needs periodic refreshing. Thus, they can be mentally prepared on time for password change.

Once the users increase the amount of time spent on the application and the volume of valuable data produced, the system adapts to this behavior and gradually notifies the user that the initial password is too weak to protect their assets and should opt for a new, stronger one. This approach allows users to understand and estimate their risk, since it is based on realistic usage data. The idea here is that the more the users use the system, the more incentives they have to protect their identity and digital assets and thus, be more receptive to password change. The main observation here is that the password change does not occur after a fixed amount of time e.g., after 180 days, but rather after enough actual usage of the system. This approach has the nice side-effect that users that seldom use the system are not burden with a lot of management operations once they manage to login to the system after a great period of inactivity.

A configuration interface upon account creation will collect user preferences on how and when the user should be notified on password policy related actions. The interface must offer at least notification options within the range from "every login" to "never" in sensible time increments. When time to notify is reached, there must be alternative contact methods; except from the user interface –which might be the worst option-, users can be notified through SMS messages, instant messaging, and email messages. Users are actually accustomed to such out-of-bounds communications from real world applications. For example, consider the case of changing credit cards upon expiration. Users can accept such a method, feel that the system offers increased security, and do not interrupt their regular browsing experience.

As time goes by and the password expiration date is reached, the "password change" operation should be gradually moved in the front of the user instead of staying buried in some obscure option. To an extreme, the password change link should appear under specific circumstances as an intermediate page just after the user enters the system (and not when the user is presented with the actual application interface and calls for action).

A final step is the "password reset" operation, which is nowadays usually implemented through answering a "secret question". There are two points to improve in relation to this task. The first point is that it must be clearly communicated to the users that the security question does not protect them from people they know but from strangers. So the users must be aware that their "secret" answer is more valuable and more critical than their password. If someone close to them knows the secret answer, then their account and data can be exposed. From a system point of view this is usually not considered a real threat for the system. However, from a user point of view, this threat can be real and the involved risk much greater.

The second point is to improve the "Remind my password" cycle when a security question is not involved. Current practice is to use an alternative email for communication. From a security point of view, this can create more problems; the alternative email address might not anymore be in the possession of the user or the message might be erroneously filtered in intermediate mail servers as spam. In addition, these messages are sent without some form of encryption, so any intermediary can read and act on them. It would be preferable, from a user point of view to a) explore alternative verification means, like sending a new password through SMS and b) periodically verify users' addresses in order to keep the users notified and the addresses up-to-date. We further claim that once the users are accustomed to a password change habit for their benefit (and not enforced to), this "remind my password" option shall be used less.

Overall, a user-centric approach to password management seems more attractive and capable to solving current problems users face on selecting and using a good password. The design described above focuses on user needs and aims and not on system policies and thus, it is expected to be more acceptable by the users themselves.

VI. CONCLUSIONS

In this paper we examined the contradicting requirements that developers face when designing for "usable security". In particular, we analyzed the four main areas of research in the field i.e., password management, usability of security applications, secure interface design, and CAPTCHA protection. In this context, we identified "usable security" as a significant crossroads priority problem and argued that security must be engineered with priority for the user and not the system. Our proposal is to apply a user-centric approach for designing "usable security" and we demonstrated how it can be applied on the "password change" problem by streamlining the operation in the day-to-day use of a web application.

This work lays the foundations of a new approach on designing for "usable security", which needs to be further explored and validated. One interesting direction is to design and incorporate the password change mechanism on classical web-based applications and evaluate through user studies their usability. Another direction is to validate the applicability of the user-centric design on the other three identified areas. Further user studies can lead to refinement and improvement of our proposal on the password change problem and other typical security related scenarios.

REFERENCES

[1] S. Williams, *Free as in Freedom: Richard Stallman's Crusade for Free Software*. O'Reilly & Associates, Inc., Sebastopol, CA, 2002, pp. 52–54.

[2] A. Adams and M. A. Sasse, "Users are not the enemy: Why users compromise security mechanisms and how to take remedial measures," *Communications of the ACM* 42(12), pp. 40-46, December 1999.

[3] A. Whitten and J. D. Tygar, "Why Johnny can't encrypt: A usability evaluation of PGP 5.0,", In *Proceedings of the Eight USENIX Security Symposium (Security'99)*, Washington DC, USA, 23-26 August 1999, pp. 169-183.

[4] Computing Research Association, "Four Grand Challenged in Trustworthy Computing" final report of CRA Conference on Grand Challenged in Information Security and Assurance, Airlie House, Warrenton, Virginia, November 16-19, 2003.

Available: http://archive.cra.org/reports/trustworthy.computing.pdf

[5] J. Yan, A. Blackwell, R. Anderson, and A. Grant, "Password Memorability and Security: Empirical Results," *IEEE Security & Privacy*, 2004, pp. 25-31.

[6] S. L. Garfinkel and R. C. Miller, "Johnny 2: A User Test of Continuity Management with S/MIME and Outlook Express," *Proc. Symp. Usable Privacy and Security*, ACM Press, 2005, pp. 13-24.

[7] S. E. Schecter, 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," *IEEE Symp. on Security and Privacy*, Oakland, CA, USA, May 20-23, 2007.

[8] L. Falk, A. Prakash, and K. Borders, "Analyzing Websites for User-Visible Security Design Flaws," *Proc. Symp. Usable Privacy and Security*, ACM Press, 2008, pp.117-126.

[9] S. Bratus, C. Masone, and S. W. Smith, "Why Do Street-Smart People Do Stupid Things Online?" *IEEE Security & Privacy*, May/June 2008, pp. 71-74.

[10] J. Johnston, J. H. P. Eloff, and L. Labuschagne, "Security and human computer interfaces," *Computers & Security 22(4)*, pp. 675-684, 2003.

[11] J. Nielsen, "Ten Usability Heuristics" [Online]. Available: http://www.useit.com/papers/heuristic/heuristic_list.html

[12] C. J. Hernandez-Castro and A. Ribagorda, "Pitfalls in CAPTCHA design and implementation: The Math CAPTCHA, a case study," *Computers & Security 29(2010)*, pp. 141-157, 2010.

[13] S. Garfinkel and G. Spafford, *Practical UNIX and Internet Security*. O'Reilly & Associates, 1996.

[14] J. Yan and A. S. El Ahmad, "Usability of CAPTCHAs or usability issues in CAPTCHA design," *Proc. Symp. Usable Privacy and Security*, ACM Press, 2008, pp.44-52.

[15] S. J. Payne, "Mental Models in Human-Computer Interaction", *The Human-Computer Interaction Handbook, 2nd edition*, A. Sears and J.A. Jacko (ed.), pp. 63-76, CRC Press, 2008.

[16] A. Herzog and N. Shahmerdi, "User Help Techniques for Usable Security," *Proc. CHIMIT 2007*, ACM Press, Cambridge, MA, March 30-31, 2007.

[17] C. Masone and S. Smith, "Towards Usefully Secure Email," *IEEE Technology and Society Magazine* 26(1), pp. 25-34, Spring 2007.

[18] K.-P. Yee, "User Interaction Design for Secure Systems," *Proc. 4th Int'l Conf. Information and Communications Security*, Springer-Verlag, 2002, pp. 278-290.

[19] D. A. Norman, *The Design of Everyday Things*, Basic Books, New York, 2002.