Position: The User is the Enemy

S. Vidyaraman, M. Chandrasekaran and S. Upadhyaya
Computer Science and Engineering
University at Buffalo
Buffalo, USA
Email: {vs28, mc79, shambhu}@cse.buffalo.edu

ABSTRACT
The Human Factor has long been recognized as the weakest link in computer systems security, yet, nothing technically significant has been done to address this problem in an attack agnostic manner. In this paper, we introduce the mantra of "*The User is the Enemy*" for security designers and developers alike as an underlying current towards addressing the weak human factor. We present different notions of the user and the system and argue from parallel tracks that user actions, both ignorant and non-compliant, are detrimental to the organization. We further show how the paradigm has been applied in a rather unconscious manner and contend that security mechanisms borne out of a conscious application will be more effective towards addressing this systemic problem. Our position is not meant to be a cynical attitude towards users; rather, it is meant to be the focal point of security design *attitude*, similar to the mantra "*All user input is evil*" for addressing buffer overflow attacks.

Categories and Subject Descriptors
H.1.2 [**Models and Principles**]: User/Machine Systems – *Human factors*

General Terms: Security, Human Factors

Keywords
User Centered Security, Non-Compliant users

1. INTRODUCTION
A fundamental process and the common denominator to most systems is the interaction between the system and the user. It is this very interaction that is responsible for the functioning of the system; and in most cases, regardless of the goal of the system, this very interaction, according to Schneier [13], is the greatest risk. The variable and often unpredictable factor in this interaction is the user. It is this user who has earned the label of a *Weak Human Factor*. The weak human factor does not always cause a threat in isolation; rather, the actions of users are the starting point for some attacks, and in some cases, the users themselves may launch the attacks. Weak passwords, susceptibility to social engineering attacks, failure to install security updates, etc., are but some examples of how the weak human factor manifests itself. These scenarios are prevalent in most computer systems today, and are increasingly becoming a major source of financial losses to organizations. Each of these situations introduces a weak link in what may otherwise be a technically sound system.

NSPW'07, September 18–21, 2007, North Conway, NH, USA.

As Hassel and Wiedenbeck [9] rightly mention, there is a need to inculcate a "culture of security" in users. There is a school of thought (and we agree with it) that maintains that it is not the users' fault that they perform the easiest action; rather, it is the designers fault (both security and HCI designers) to have made the most insecure operation the easiest operation. The prominent work in this regard is by Adams and Sasse [2] in their work titled "Users are not the Enemy." Although the title runs contrary to the theme in this work, it must be noted that their study was related completely to users' knowledge of security and specifically, password systems. In fact, contrary as it may sound, we agree with their primary conclusion – that users' are not aware of the threat model of the system and that "*It is important to challenge the view that users are never motivated to behave in a secure manner*" [2]. But researchers in [2], [7] and [12] all agree on a fundamental conclusion: *users are generally careless and unmotivated when it comes to system security*. This is not surprising, since the security subsystem is a class apart from other subsystems in that it has a negative requirement [11], [8] to fulfill. Thus in the normal course of work, users are not required to interact with it at a conscious level unless forced to. For example, users have to type in their username and password since it is mandatory. But since (a) installing security updates is not made mandatory and (b) they do not contribute towards the workflow, it often takes a low priority for the user to execute.

The position advocated in this paper is not meant to be a cynical one; indeed, most of the argument paths represent what many practitioners in the security field have felt, and if suitably interpreted, have even (unconsciously) implemented. We first present the core argument for our position "*The User is the Enemy*". We then state the effect of this paradigm on security design, and indicate prior approaches that we believe, exhibit traces of this approach.

2. THE USER IS THE ENEMY
The threat posed by legitimate users in an organization has appropriately been labeled as "The Enemy Within" [10] in a survey by McAfee Corporation (http://www.mcafee.com). For purposes of this paper, we divide the user broadly into two different categories:

- *Type I: A Legitimate User* – This category of users includes legitimate and authorized users of the system. These users log into the system and execute workflow processes according to their roles. According to the McAfee Survey [10], such users are varyingly labeled as "The Security Softie",

"The Gadget Geek" or "The Squatter." While they do not have any stated intentions to disrupt the system, their actions nonetheless endanger the system.

- *An Ignorant User* – "The Security Softie" or "The Squatter" fall under the category of ignorant users. These users do not have any idea of the threat model of the system and hence, may not implement the best practices suggested by the organization.
- *A Non-Compliant User* – "The Gadget Geek" is a typical non-compliant user. They may possess a limited understanding of the security threats, but are more motivated towards immediate performance gains, and as a result, circumvent security policies or refuse to follow best practices.

- *Type II: A Legitimate, but Malicious User* – Similar to Type I users, users in this category are legitimate, i.e., they possess authorized credentials to log into the system. However, their goal is to disrupt the system, either through a self inflicted cataclysmic system compromise or through slow poisoning attacks like leaking confidential information about the organization to its competitors. According to the Survey [10], such users are labeled as "The Saboteur."

We contend that both these types of users are the enemies of the system – willingly or unwillingly – and according to the classification that each user falls under, security policies have to be applied appropriately. It is trivially clear from their very definition that Type II users actively work towards compromising the system resources. It is also very clear that Type I users, in everyday parlance, are *not* the systems' enemy; indeed, they are the very users for whose protection security mechanisms have been designed. But it is this idea that we seek to modify at the very least – that the notion and treatment of Type I users must change and the notion of a 'system' as viewed by security designers is erroneous.

Notion of Users: The behavior of Type I users and the effect they have on the systems' security state requires a deeper analysis. According to Adams and Sasse [2], Type I users do not have a clear conception of the threat model of the system. In addition, they are not inclined towards the appropriate application of well established procedures, also called *best practices* in organizations. This lack of inclination (for applying *best practices*) on the users' part often stems from either ignorance or difficulty of the processes. By difficulty of processes, we mean that these processes:

(a) are not easy to implement - like choosing difficult passwords

(b) often interfere in the regular workflow - like installing security updates

Type I users usually prefer to take the easy route of operation and prefer to get their job done with least amount of work, despite the potential long-term detrimental effects of their actions. Often, they are either unaware of the effect of their actions, or prefer to ignore them. The easiest operation often involves ignoring best practices completely. Such user actions are mostly domain and context specific and are impossible to foresee. Once their symptom is exposed, they can be corrected or technical solutions can be applied to prevent the problem. But a deeper implication of this analysis emerges. The fact remains that Type I users (at least a majority of them), are more concerned towards the fulfillment of their workflow, without regard to the application of security best practices either due to ignorance or otherwise. As a result, they eventually end up compromising security by slowly eroding the security state of the system. The actual system compromise usually happens through a single cataclysmic action like the installation of a virus on the network or the exploitation of some security vulnerability in un-patched software. But the root cause of this would have been inaction (not installing a security update) or insecure action (like installation of software from a link that came through a socially engineered email) by a Type I user. As a result, the Type I user *unwittingly becomes* the Enemy of the System, and must necessarily be treated as such by the Security designer.

Notion of System: In this context, we define the notion of 'system' in a broader sense, more in terms of the business ideology where the system actually exists, rather than the often ill-considered notion of a single computer or multiple computers networked together. This notion of a 'system' has existed in an implicit manner for all security designers, but has never found proper expression due to an established design paradigm that has its roots in the HCI domain. Consider a business entity with a few thousand employees, each with a single computer, networked together for communication and collaboration through dedicated servers. In such an organization, traditional outlook has it that the 'system' is the desktop computer (or the networked computers). The users are the employees, and the purpose of the 'system' is to serve the users. Therefore, the design paradigm that has its origin in the HCI domain, demands that the 'system' be easy and simple to use, with all the intricate sub-systems being hidden from the user through a process of abstraction.

This view of the 'system', we argue, is incorrect from the business perspective or more aptly, from a viewpoint which should emphasize the organization and its goal instead of individual users. The 'system' is in fact the aggregation of the networked computers *and* their users. It is this 'system' that exists from a business standpoint – its goal being to provide services (or some profitable outcome). *From a security design viewpoint, rarely is it explicitly realized that the users too are part of the system which exists to serve a goal*.[1] This erroneous view of the system, along with the HCI design paradigm has resulted in the (inefficient) design (and implementation) of security systems and enforcement mechanisms. Since the HCI design paradigm demands that all sub-systems be hidden from the user for ease of usage, the security subsystem too has been hidden under this abstraction layer. Therefore, the Type I user is rarely aware of the existence of the security subsystem, and as Adams and Sasse [2] imply, is not aware of the threat model of the system. The interface design too has tagged along with this design paradigm. Consider, for example, the by-now-ubiquitous Microsoft Windows Security update mechanism. Desktop users usually get a 'Balloon' tip on their desktop which says "Updates are Ready." Since the design paradigm demands that the interface be easy to use, there is a close option that is conveniently located on the right edge. Although statistics are not available for the type of actions users take, it is trivial to observe that the 'close' action is the easiest one to effect in such situations. It is also not hard to imagine the average users' reaction – He would prefer to close this annoying interruption ra-

[1] The 'new' systems view has (already) been proposed by John Gall in his book SYSTEMANTICS

ther than install the updates, which more often than not demand a reboot of the system and interrupts his workflow. Thus the Type I user *unwittingly becomes* the Enemy of the System, and must necessarily be treated as such by the security designer.

Observe that by changing our notion of the system and the user, we arrive at the same conclusion, albeit from different argument paths. Also note that we do not state that the Type I user is *really* the enemy of the system as the Type II user is; we only argue that due to a complicated intertwining of design methodologies and special (negative) requirements of the security subsystem, the Type I user *unwittingly becomes* the enemy of the system, and if the overall security state of a system is to improve, this must necessarily be the *attitude* of security designers.

3. SO WHAT?

What would be the effect of such an attitude among security designers and developers? Our position is similar to the mantra of "*All User Input is Evil*" when dealing with buffer overflows: developers are required to treat all inputs (user driven or not) with care and perform appropriate checks on the inputs. Drawing the analogy further, we advocate that all *user actions* be treated with the same care as that of user inputs. Consider the following classification of user actions.

Action Type I – We define the fundamental user actions required for the workflow to be Action Type I. These fundamental actions are defined by the user's role in the environment. For example, a graphics designer will need to use some photo/video editing software. In addition, a HCI device like a tablet may need to be connected to the computer via the USB interface for rendering hand sketches. All actions pertaining to this workflow become Type I actions.

Action Type II – Ancillary actions required for the fundamental actions to work. For example, exploring the hard drive is a prerequisite for most job roles. In addition, connecting USB devices, burning images onto a CD may be in this list for a graphics designer.

Action Type III – These are actions that are not predefined like Actions types I and II. These actions are the ones that users normally execute without any restrictions, since they do not fall under the purview of 'restricted objects.' They might have the potential to disrupt the working of the system, or may be inimical to the individual. Examples of such actions include clicking on a potential phishing link in an un-trusted/unsigned email.

Our approach to the security design is as follows. Given a system and a user role, we define the Type I Actions that are deemed necessary for the job role. Towards these actions which contribute only towards the workflow, there shall be no interruption from the system. Ancillary actions (Type II actions) also receive almost zero interruption, but may occasionally be interrupted depending on the action type and the evaluated effect it may have on the system. Towards all other actions (Type III), the system shall maintain a strict monitoring status and may even deny the users permission to execute those actions. The belief here is that the Type III actions are the ones that may constitute a threat that tries to exploit the weak human factor.

The application of such classification and monitoring of actions is difficult and inadequate, to say the least. Apart from the privacy issues related to granular action monitoring, a very strict workflow definition is not only hard, but may also prove counterproductive, except for some very specialized environments. However, we believe it is a step in the right direction. Thus, the monitoring and regulation of the actions is the how we perceive the initial application of the mantra. Note that the classification is meant to be illustrative, not encompassing; we specifically do not mention about *inactions* that result in a problem. For example, not installing a security update is more of inaction rather than an action (though one could argue that closing the security prompt is an action; this however, is more of an engineering issue).

3.1 Applying the Paradigm on Users

There is usually a two pronged approach towards addressing the Type I users: ignorant users have to be *educated* about the security mechanisms operations and expectations; on the other hand, non-compliant users, we argue, have to be *persuaded* to follow the security best practices. For now, we shall skip over the question of detecting whether a user is ignorant or non-compliant. Towards the ignorant user, one of the goals of human centered security is to *aid and enable* them in making the correct decisions. The concept of a security mechanism communicating its requirements to the users or correctly exposing its operational specifics/interface to the ignorant user falls under the category of *usable security* [1]. With the attitude advocated in this paper, a usable security mechanism must provide proper feedback to an ignorant user upon either inaction or a Type III action invocation.

We shall focus more on the non-compliant user and consider the result of the attitude advocated in this paper towards addressing the threat they pose to the system. Consider the notion of a tradeoff between security and performance: so far, it has been applied to the computing system, not to the user. For example, an interactive process for the user is always prioritized and any performance hits due to security is relegated to the computing system unless unavoidable; in particular, a non-compliant user does not perceive any slowdown due to his defiance. Now consider the concept of *Safe Staging*, introduced by Whitten and Tygar [16]. If we extend the concept to a gradual QoS throttling of user (interactive or otherwise) services in the face of blatant non-cooperation, we might just be able to elicit user cooperation with security systems, thereby addressing the problem of a non-compliant user. The concept of gradual QoS reduction as a mechanism of implicit or explicit communication has been proposed in other economic scenarios [3], but has not been implemented in any meaningful manner (yet) in the realm of human centered security.

In some sense, we are arguing for a tradeoff between compliance to security best practices and user performance/nuisance *perception*, not *actual* performance degradation. This tradeoff, unlike previous approaches, *directly penalizes the non-compliant user*, much like the concept of punishment in wireless networks [6], *as opposed to leaving the non-compliant user out of the equation*. The notion of explicitly penalizing users to elicit cooperation requires further investigation, most probably requiring interdisciplinary studies.

Furthermore, the approach of monitoring and interrupting user actions/workflow for the sake of security is not entirely new. In fact, security designers have started applying this approach unconsciously, but unfortunately without any clear model. There is an undercurrent of this concept in the Microsoft Windows Operating System; there is no explicit QoS throttling, but more of an interruption to the users workflow. Consider these scenarios:

1. *Windows reboot 'nag' dialog*: On installing Microsoft Security Updates, users are prompted repeatedly to reboot the system (if required) in order to complete the update. The dialog box pops up and has an option to "Reboot Later." But the update application cannot be stopped nor is there any choice like "Do not remind me." Even if the update process (wuauclt.exe) is forcefully terminated, it is restarted automatically, unless the update service is stopped.

2. *Application Block across different Windows Zones*: Starting from Windows XP Service Pack 2 (SP2), all files downloaded by Internet Explorer are 'blocked' from executing or being opened. Uses have to explicitly 'Unblock' the files if they are to be opened without an intermediate dialog. This is applicable not only to executables, but also to Office files and other file formats.

3. *User Account Control in Windows Vista*: With the tendency of users to login in administrative mode for various reasons, integrity levels have been introduced in Vista that enables a security boundary even when applications are running with administrative credentials. The User Account Control (UAC) forces users to explicitly grant applications administrative privileges before execution.

The above scenarios actually push the security decision process to the user, but if we interpret 'QoS throttling' and 'interruption of the workflow' as an anti-user measure, we can observe the undercurrent. For a typical user, UAC assumes that administrative privileges should not be required under normal operating conditions; *it then merely interrupts the Type III actions for a typical user*. There is no notion of penalizing, nor is it feasible in a home computing environment where 'workflow' does not exist.

3.2 Synergy with Existing Approaches

The weak human factor is an age old problem, and prior approaches to the problem have met with limited success in terms of their permeation to mainstream systems. Recall that the focus of this paper is on a technical approach that involves only the software/hardware components without any externalization, i.e., educating the user or "persuading" [15] the user through incentives or fear appeals. All the prior approaches discussed in this section have a common undercurrent: they all focus on one particular manifestation of the weak human factor and propose schemes to address that particular threat vector; they do not approach the problem in a holistic manner (nor was it their intention to do so). A holistic approach is just that, an *approach*, not a methodology *per se*. Consider the work by Weirich and Sasse [15], where the authors discuss the usage of fear appeals towards persuading the user to use effective passwords. This approach considers password based systems (though it could easily be extended to other threat vectors also) and investigates the possible steps outside of the cyber domain to enforce good password practices. A more encompassing theme is presented by Brostoff and Sasse's [4] work, where the authors advocate a safety-centric system design and argue the equivalence, and perceived advantages of such a design approach. They too, like Adams and Sasse [2], state that the very term "weak human factor" somehow shifts the blame onto the users, instead of advocating better models and design techniques: given the fact that non-compliant users also exist in the system who do not explicitly fall into their error models, our position can be viewed as an extension to their safety based model.

However, the approach that is more reflective of the position advocated in this paper is found in Xia and Brustoloni's work [17], where the authors consider Man-in-the-Middle (MTM) attacks and harden browsers to protect users. From the users' view point, their technique presents an explanation of possible MTM attacks and refuses to allow further transactions unless the problem is addressed. More often-than-not, this involves the users' reading a somewhat lengthy explanation and asking administrators to resolve the situation. While the solution by itself 'solves' the problem, it is in fact the harbinger of more administrative problems, and is likely to result in a stoppage of the users' workflow if implemented in a commercial environment. Consider this coupled with the fact that (we assume) users will simply not bother to read warnings and information boxes, and would prefer to click on the nearest button that allows them to proceed with their workflow. The validity of such schemes is thus circumspect since users may not pay heed to such warnings, but what really stands out in their work is the fact that users are *forced* to stop their workflow without any option, and thus are put to discomfort, or, (very) loosely speaking, are treated as the enemy. However, in the interests of enabling workflow, they also provide a mechanism enabling the user to override warnings. In the context of file downloads, consider the approach adopted in [5], where users are requested to type out a reason for opening a potentially dangerous file, and are warned of a fine or suspension if the reason is unjustifiable. This work explicitly penalizes users for non-compliance (based on prior warnings).

These illustrations also bring out a very important point, i.e., the link between the workflow of a user and the security mechanism must be tightly coupled, and although other actions of the user are important, the security mechanism must impede them in the interests of the larger 'system' that serves an overarching goal. In this context, the suitable interpretation of the mantra would be to "interrupt the users' workflow if necessary."

3.3 Applicability

The crux of this position paper may be summarized as follows:

1. Security Mechanisms cannot afford to take the user out of the loop.

2. Potentially threatening actions (by users) tend to move the system towards a more vulnerable state; these actions must be stopped in the interests of the securing the system.

3. Users who initiate (the) potentially threatening actions despite warnings from the system must be treated as the enemy, and must be penalized.

The first two points are not 'new'; they have been felt by security practitioners and many mechanisms that involve the user in the security loop have been proposed. This position differs from previous approaches in the third point, viz., explicit penalty mechanisms for non-compliant users who refuse to act in the interests of the system. Not surprisingly, the notion of explicit penalty, stated in the absolutist form, without reference to:

- the specific interpretation
- the underlying assumptions made on the system design
- the context of application

has drawn many objections. Consider the example of Windows Updates: in particular, consider the SQL Slammer worm that made Bank of America's (nearly 10,000) ATMs inoperative. Microsoft itself was a victim to this worm, despite the fact that an update was available for the vulnerability. If the paradigm advocated in this paper were applied, users would be forced to update their machines within a specified period of time, without any option to do otherwise. By itself, this would 'solve' the problem of uninstalled updates (and vulnerabilities rising thereof). But viewed in an absolutist sense (in terms of the vendor – Microsoft – forcing updates), the specific solution of forced updates poses more problems than it solves, least of which is vendor trustworthiness. A recent survey [14], indicates that mid-sized organizations roll out as many as 1 'change[2]' per day to their systems, of which about half cause problems to the systems and result in a rollback. Thus, forced updates, under these conditions, are unadvisable. However, with reference to the SQL slammer incident, forced updates would be applied to systems under restricted conditions, subject to a timeframe for organizations to test the effect of the update on their business processes. In that sense, forced updates are appropriate in a corporate environment rather than a home environment. Likewise, every specific solution that implements this approach must be considered with respect to the context in which it is applicable, not in a universal sense. This is similar to the concept of hard staging [16], that is "unlikely to be appropriate in software intended for general consumer use."

4. CONCLUSION

Subsystems have long followed the traditional HCI principle of transparency. The application of that principle to security subsystems has resulted in an increasing 'unawareness' on the part of users towards good security measures, due to which users have 'earned' the epithet 'weak human factor'. In order to realize the position advocated in this paper, there must first be recognition from designers that security mechanisms can no longer be transparent to the user. Users must be involved in the security loop; ignorant users must be educated and non-compliant users must be 'persuaded' (or forced) into compliance for the sake of the system. The concept of educating the users and persuading them is not new; indeed, much of what has been advocated in this work is what practitioners in the security field have felt and applied, but with a caveat that enables users to override mechanisms, usually in the interests of maintaining workflow. The caveat, which is meant to be an exception, is used to circumvent the security measure for ease of use and increased performance, thereby exposing the threat vector that the mechanism was initially supposed to address. Apart from the attitude change, a better mechanism to incorporate the notions of workflow, behavior and trust into security mechanisms is required in order to realize a technical system capable of addressing the weak human factor. Thus, in the ideal world, not only can user's workflow be defined correctly, but the interaction between different 'job units' can be specified in terms of expected behavior and enforcement expectations, in terms of trust or security.

The advantage of adopting the attitude comes primarily to the security designers and implementers alike, much like the focus that was placed on buffer overflow attacks by the *mantra* "*All (user) input is evil"*. Designers will not only consider systems from a purely (functionally) enabling viewpoint, but also design them with the human factor in mind. The immediate disadvantage, however, will be in the form of a backlash from the users; those who are used to uninterrupted workflow might not like the newly imposed burden; those who are used to circumventing security measures for performance gains (like the gadget geek) will (rightly) be frustrated at being labeled 'the enemy.' In this context, based on the feedback[3] received, this position could alternatively be stated as "The User is a Vulnerability", but that does not convey the idea of a penalty mechanism for non-compliant users, as in [5]. But we hope that with an attitude of treating the user as the enemy, responses such as the UAC in Vista, forced updates, etc., can be applied more intelligently in systems.

ACKNOWLEDGEMENTS

This research was supported in part by U.S. Air Force Research Laboratory Grant No. 200821J. The authors would like to thank Kevin Kwiat of Air Force Research Labs, Mary Ellen Zurko of IBM and three anonymous reviewers for their constructive comments. The authors would also like to acknowledge and thank all the attendees of NSPW 2007 at New Hampshire, USA; their compelling objections and forceful comments have gone a long way towards improving the clarity of this work.

REFERENCES

[1] *Usable Security: http://usablesecurity.com/*.

[2] A. Adams and M. A. Sasse, *Users are not the enemy*, Commun. ACM, 42 (1999), pp. 40-46.

[3] A. Andoni and J. Staddon, *Graceful service degradation (or, how to know your payment is late)*, *Proceedings of the 6th ACM conference on Electronic commerce*, ACM Press, Vancouver, BC, Canada, 2005.

[4] S. Brostoff and M. A. Sasse, *Safe and sound: a safety-critical approach to security*, *Proceedings of the 2001 workshop on New security paradigms*, ACM Press, Cloudcroft, New Mexico, 2001.

[5] J. C. Brustoloni and R. Villamarín-Salomón, *Improving security decisions with polymorphic and audited dialogs*, *Proceedings of the 3rd symposium on Usable privacy and security*, ACM, Pittsburgh, Pennsylvania, 2007.

[6] Dave Levin, *Punishment in Selfish Wireless Networks: A Game Theoretic Analysis*, *Proceedings of Economics of Networked Systems, NetECON* Ann Arbour, Michigan, 2006.

[7] P. Dourish, R. E. Grinter, B. Dalal, J. Delgado and M. Joseph, *Security Day-to-Day: User Strategies for Managing Security as an Everyday, Practical Problem*, Institute for Software Research, University of California, Irvine, 2003.

[8] G. Fink and M. Bishop, *Property-based testing: a new approach to testing for assurance*, SIGSOFT Softw. Eng. Notes, 22 (1997), pp. 74-80.

[2] A 'change' is not functionally equivalent to a full fledged update

[3] http://www.cse.buffalo.edu/~vs28/nspw2007

[9] L. Hassel. and S. Wiedenbeck., *Human factors and information security, http://clam.rutgers.edu/~birget/grPssw/hasselSue.pdf*, 2004.

[10] McAfee Corporation, *The Enemy Within; http://www.theregister.co.uk/2005/12/15/mcafee_internal_security_survey/*, 2005.

[11] C. C. Michael and W. Radosevich., *Risk-Based and Functional Security Testing; https://buildsecurityin.us-cert.gov/portal/article/bestpractices/security_testing/overview.xml*, 2005.

[12] M. A. Sasse, *Computer Security: Anatomy of a Usability Disaster, and a Plan for Recovery*, *CHI 2003 Workshop on Human-Computer Interaction and Security Systems*, Ft. Lauderdale, FL, USA, 2003.

[13] B. Schneier, *Secrets and Lies: Digital Security in a Networked World*, John Wiley & Sons, Inc., New York, 2000.

[14] StackSafe, *IT Ops Research Report: Downtime and Other Top Concerns*, July 2007.

[15] D. Weirich and M. A. Sasse, *Pretty good persuasion: a first step towards effective password security in the real world*, *Proceedings of the 2001 workshop on New security paradigms*, ACM Press, Cloudcroft, New Mexico, 2001.

[16] A. Whitten and J. D. Tygar, *Safe staging for computer security*, *HCI and Security Systems Workshop, CHI*, Ft. Lauderdale, Florida, 2003.

[17] H. Xia and J. C. Brustoloni, *Hardening Web browsers against man-in-the-middle and eavesdropping attacks*, *Proceedings of the 14th international conference on World Wide Web*, ACM Press, Chiba, Japan, 2005.