Learning of Personalized Security Settings

Mehrbod Sharifi
Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
mehrbod@cs.cmu.edu
www.cs.cmu.edu/~msharifi

Eugene Fink
Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
e.fink@cs.cmu.edu
www.cs.cmu.edu/~eugene

Jaime G. Carbonell
Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
jgc@cs.cmu.edu
www.cs.cmu.edu/~jgc

***Abstract*—While many cybersecurity tools are available to computer users, their default configurations often do not match needs of specific users. Since most modern users are not computer experts, they are often unable to customize these tools, thus getting either insufficient or excessive security. To address this problem, we are developing an automated assistant that learns security needs of the user and helps customize available tools.**

***Keywords*—Cybersecurity, machine learning.**

I. Introduction

The growth of the Internet has led to development of numerous anti-virus and anti-malware tools and enhancement of security in many applications. The increased number and complexity of security decisions introduces a new challenge for the user, who needs to understand the related options and customize security tools. This challenge is often overwhelming for users who are not computer experts. As a result, they may select inappropriate settings and make wrong choices in response to dialog boxes, thus becoming vulnerable to malware attacks. The related research shows that inappropriate use of these tools is one of the main causes of computer vulnerabilities [1].

Security tools, web browsers, and other security-conscious applications display various prompts to ensure that the user is aware of potential threats, but these prompts are often difficult to understand. In Figure 1, we give an example of Internet Explorer settings that are incomprehensible for most novice users. Note that, while we use Internet Explorer in all examples, other web browsers cause similar issues, and we make no claims about relative advantages of specific browsers.

Many users do not even try to understand security prompts and instead get into the habit of giving the same answer to all prompts, such as always clicking *yes*. For example, some users routinely bypass security certificate warnings [1], which can result in insufficient security when transmitting sensitive data, say, in the case of online banking.

A common solution for reducing the number of prompts is the "one-size-fits-all" approach. Specifically, software includes default settings and sometimes a limited customization on the initial run or when first using certain features. This approach is often ineffective because appropriate security decisions may depend on specific user needs.

To address this problem, we propose an approach based on learning user needs and assisting the user with security decisions, which adapts to the changing environment and evolving user needs by observing the user and asking targeted questions. For example, if the user consistently bypasses security certificate warnings, the system assesses the related risks and determines whether it should bypass such warnings automatically, without prompting the user, or provide a customized explanation of the risks and get the user feedback.

Figure 1: Example of confusing settings in Internet Explorer.

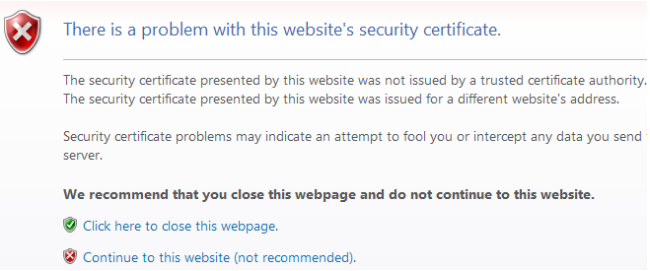

Figure 2: Certificate issue warning.

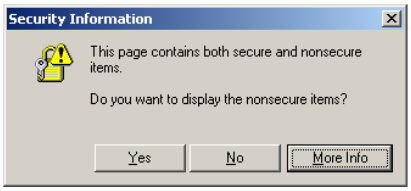

Figure 3: Mixed content warning.

The described work has been supported by the Department of Homeland Security through Command, Control and Interoperability Center for Advanced Data Analysis (CCICADA).

II. MOTIVATING SCENARIOS

We begin with use cases that illustrate two security concerns. The first case is related to data exchange, whereas the second involves execution of software from the Internet.

A. Data Exchange

The data exchange between the user's computer and a website is typically based on the HTTP protocol, which is vulnerable to interception and manipulation by hackers. When exchanging sensitive data, websites use HTTPS, which includes encryption protocols such as Secure Socket Layer (SSL). To verify the website identity, the browser checks that the website's SSL certificate contains the correct domain name, has not expired, and is issued through a trusted certificate authority. If the certificate does not satisfy these conditions, the browser issues a warning (Figure 2).

Since encryption may be computationally expensive, websites often switch to HTTPS only when necessary, and the user should be aware of these switches. Browsers have the option to notify the user of switching between HTTP and HTTPS, but these notifications are annoying and usually disabled. A more complicated issue arises when a webpage "looks" to be using HTTPS but its components are using HTTP. Some browsers notify the user about such situations and ask whether to display the non-secure parts (Figure 3). If the user clicks *Yes*, the browser loads all contents; if she chooses *No*, the system blocks the non-secure parts. Most user however do not understand this prompt and are unsure how to respond.

To address this problem, we provide one more button, called *Unsure*, in addition to *Yes* and *No*. If the user clicks *Unsure*, a separate program tries to:

- Give a nontechnical clarification.
- Ask simpler relevant questions.
- Make a decision on behalf of the user.

This assistant program has access to information about:

- *User*, such as how much she knows about SSL certificates.
- *Website*, such as whether it belongs to a trusted company.
- *Current task*, such as whether the user is sending an email or transferring money.

It can also ask targeted questions, such as "are you going to provide confidential information, e.g. your password?"

If the user always gives the same response to the questions in Figures 2 and 3, without understanding their significance, then the use of an automated assistant strengthens the security and improves the user experience.

B. Software Downloads

The original purpose of browsers was to show HTML pages but they have evolved to present more active contents, such Java applets, Flash, and ActiveX, which are handled by separate client-side applications. Besides, the user can extend browsers with toolbars and plug-ins, which are independent programs, and she can also use browsers to install stand-alone programs.

These capabilities have introduced security risks, and operating systems and browsers have evolved to reduce those risks. For example, the Windows system checks where an executable has come from and enforces certain restrictions, such as a required digital signature with a publisher name. These measures however do not account for specific context. For instance, browsers use the same procedure for installing Adobe Flash, trusted by everyone, and for a potentially malicious plug-in obtained from an obscure source.

The security assistant accounts for more information about software sources. For example, if the user has *purchased* a specific software, it is likely that she has more trust in it as compared with a freeware. The assistant can automatically detect a software purchase or directly ask the user about it, and provide more permissions to the purchased software.

III. SECURITY ASSISTANT

The assistant maintains the following data models:

- *User-knowledge model:* The level of the user expertise in different technical areas.
- *User model:* Preferences and usage patterns, such as whether the user is making backups, whether she is keeping sensitive data, and whether she is willing to answer questions.
- *Task model:* What the user is currently doing and related security needs.
- *Third-party model:* The entity that owns a website, its type, number of customers, reputation, and so on.
- *Security-setting model:* The settings for various applications.

The system learns these models from the following inputs:

- Automatically collected data and observed user behavior; e.g. whether she regularly uses Microsoft Word.
- The user's answers to targeted questions; e.g. "do you plan to use Microsoft Word?"

The system invokes the assistant in the following situations:

- *Manual invocation:* The user clicks *Unsure*, which is added to all prompts, or requests a change of security settings.
- *Auto invocation:* A sequence of the user's actions suggests that the assistant may help; e.g. the user has consistently ignored certificate issues.

The assistant can perform the following actions:

- Ask targeted questions about the user's needs, adapted to the user's technical knowledge.
- Make a specific security decision or adjust settings.
- Explain the options in more understandable terms.

For example, it can make the following decisions in the use cases of Section II:

- *Information Exchange:*
  - Show or hide the non-secure parts of a webpage.
  - Prevent submission of data to a website.
  - Allow or block navigation to a website.

- *Software Downloads:*
  - Allow or block a run of an executable.
  - Allow or block a plug-in installation.

IV. Related Work

Artificial intelligence researchers have extensively studied the challenge of building agents that learn from their environment through passive observations and active queries [15]. For example, a travel assistant may elicit user preferences [13]. We have applied similar techniques to eliciting preferences in the context of scheduling [2, 3, 6, 7]. Researchers have also studied user modeling in recommender systems [10, 14].

On the other hand, the work on automated security assistants has been very limited. Burns *et al*. developed an assistant for system administrators that configured networks to adapt to changing environment [5]. He *et al*. described security agents that communicated with each other to reconcile cryptography protocols [8]. Jendricke and Gerd tom Markotten built a system for monitoring network traffic and detecting the cases of sending sensitive data over insecure connections; it kept profiles of users and websites but used no learning [11].

Horvitz et al. [9] designed an automated assistant for Microsoft Office suite, which represented sequences of user actions as a graphical model, inferred the user's goals, and determined when the user needed assistance.

We have developed a similar approach but applied it to security, which requires different observations and questions. Specifically, we are investigating the problems of representing information about the user, websites, tasks, and security needs; learning through observations and targeted questions; and making decisions that minimize security risks.

V. Approach

The models described in Section III are sets of properties, and the assistant learns values of these properties. The user model and third-party model are flat attribute sets, whereas the user-knowledge model (Figure 4) and task model (Figure 5) are tree-structured hierarchies. The values in the user-knowledge model are degrees of the user's familiarity with the related topics, which range from 0 (no knowledge) to 10 (expert). The task-model values are security requirements, which range from 0 (no security) to 10 (high security). These hierarchies enable the assistant to guess missing values. If it does not have a value for some node, it uses the value of the ancestor node, computed from the known values in the ancestor's subtree.

The assistant converts its observations and the user's answers to model values. It adapts questions to match the estimate of the user's knowledge level, and its confidence in the user's answers depends on the same knowledge estimate.

It also learns relations between entities, such as the relation between the "NYTime.com" website and the "reading news" task. Furthermore, it adds and reorganizes nodes based on observations. For example, it may add nodes for new browsers and word processors to the user-knowledge model.

The assistant explicitly accounts for uncertainty in the learned model properties, represented by probability distributions. For example, the uncertain properties of the user model are as shown in Figure 7, where $U = \{U_1, U_2, \ldots, U_n\}$ is the set of all properties, each U_i takes values $\{u_{i1}, u_{i2}, \ldots, u_{im}\}$, and $\boldsymbol{\pi}_i$ is the parameter vector of the probability distribution over these values.

In Figure 8, we show the assistant's graphical model. All variables are from the previously introduced models, except for *Q*, which is the user's responses to questions, and *H*, which is her past responses to security prompts. When the system calls the assistant, it selects appropriate high-level actions (Figure 9). First, the assistant checks whether it has enough data for making a decision. For example, if it has never observed the use of some software, it may not have enough data and may thus need to ask related questions. If it has enough data, it calculates the posterior distribution for the values in the security-setting model (Figure 6). If the distribution is sufficiently skewed toward one value, the assistant responds to the security prompt or adjusts the settings; else, it provides a clarification to the user.

VI. Concluding Remarks

We have presented the initial work on a security assistant that learns from observing the user and asking targeted questions. While we have developed the described techniques in the context of security, they may be applicable to other tasks that require configuring complex applications. The future work may include sharing some non-private data across users, which may help novices to benefit from decisions of more experienced users. For example, we may use the number of users who trust a website as a measure of the website's trustworthiness. We also need to address the privacy concerns involved in the data collection, which is essential for practical applications.

Acknowledgments

We are grateful to Helen Mukomel for her detailed comments.

References

[1] OWASP: Open Web Application Security Project. OWASP top 10: The ten most critical web application security risks, 2010.

[2] U. Bardak, E. Fink, and J. G. Carbonell. Scheduling with uncertain resources: Representation and utility function. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, pages 1486–1492, 2006.

[3] U. Bardak, E. Fink, C. R. Martens, and J. G. Carbonell. Scheduling with uncertain resources: Elicitation of additional data. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, pages 1493–1498, 2006.

[4] U. Bayer, P. M. Comparetti, C. Hlauschek, C. Kruegel, and E. Kirda. Scalable, behavior-based malware clustering. In *Proceedings of the Sixteenth Annual Network and Distributed System Security Symposium*, 2009.

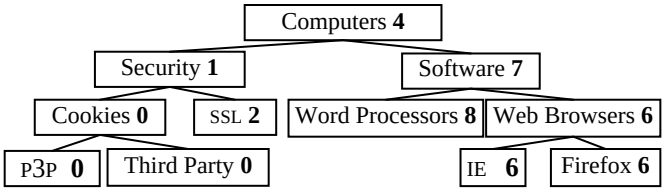

Figure 4: User-knowledge model. The numbers represent the levels of the user's familiarity with the topics.

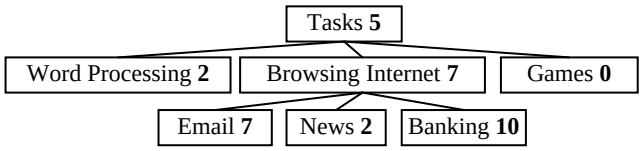

Figure 5: Task model. The numbers represent the security requirements of the tasks.

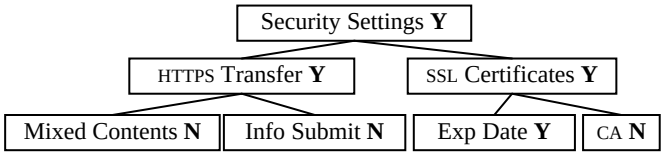

Figure 6: Security-setting model. Y (yes) or N (no) indicates whether the related security function is enabled.

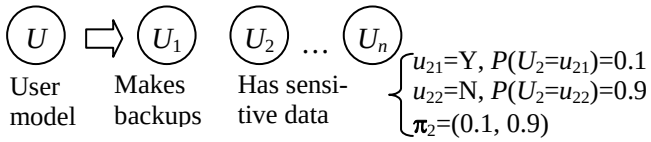

Figure 7: Random variables in the user model.

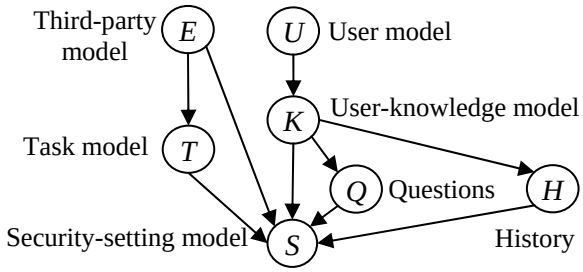

Figure 8: Graphical model for determining security settings. Each circle is a set of variables as shown in Figure 7.

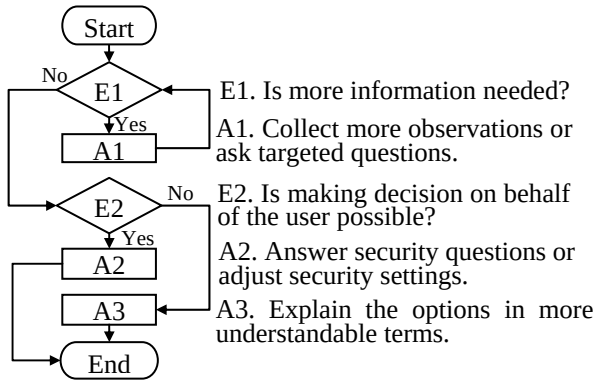

Figure 9: Actions of the security assistant.

[5] J. Burns, A. Cheng, P. Gurung, S. Rajagopalan, P. Rao, D. Rosenbluth, A. V. Surendran, and D. M. Martin Jr. Automatic management of network security policy. In *Proceedings of the DARPA Information Survivability Conference and Exposition*, 2001.

[6] A. Carpentier, M. Sharifi, E. Fink, and J. G. Carbonell. Scheduling with uncertain resources: Learning to ask the right questions. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, pages 2543–2547, 2008.

[7] S. Gardiner, E. Fink, and J. G. Carbonell. Scheduling with uncertain resources: Learning to make reasonable assumptions. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, pages 2554–2559, 2008.

[8] Q. He, K. Sycara, and T. Finin. Personal security agent: KQML-based PKI. In *Proceedings of the Second International Conference on Autonomous Agents*, pages 377–384, 1998.

[9] E. Horvitz, J. Breese, D. Heckerman, D. Hovel, and K. Rommelse. The Lumiere project: Bayesian user modeling for inferring the goals and needs of software users. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pages 256–265, 1998.

[10] T. Iwata, K. Saito, and T. Yamada. Modeling user behavior in recommender systems based on maximum entropy. In *Proceedings of the Sixteenth International Conference on World Wide Web*, pages 1281–1282, 2007.

[11] U. Jendricke and D. Gerd tom Markotten. Usability meets security—the Identity-Manager as your personal security assistant for the Internet. In *Proceedings of the Sixteenth Annual Computer Security Applications Conference*, pages 344–353, 2000.

[12] T. Lee and J. J. Mody. Behavioral classification. In *Proceedings of the EICAR Conference*, 2006.

[13] G. Linden, S. Hanks, and N. Lesh. Interactive assessment of user preference models: The automated travel assistant. In *Proceedings of the Sixth International Conference on User Modeling*, pages 67–78, 1997.

[14] E. Manavoglu, D. Pavlov, and C. L. Giles. Probabilistic user behavior models. In *Proceedings of the IEEE International Conference on Data Mining*, pages 203–210, 2003.

[15] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2003.