

Exploring Explicit Security Actions

Jennifer Stoll and Frank Park
Georgia Institute of Technology
Atlanta, Georgia 30332
{jstoll, frank}@gatech.edu

ABSTRACT

The security burden on the user can be lessened by automating security tasks or by making them more implicit (i.e., by embedding or mapping security actions to user tasks or applications). Doing so is in keeping with the reality that for users, security is almost never their primary goal. However, there seems to be a number of security tasks which necessarily require involvement from users in ways that fall outside the scope of their current task models. In fact, we discuss instances where users must take explicit security actions for which there seems to be no conceivable way of automating or making them implicit. Consequently, these actions represent instances where security must temporarily become the primary goal for the end user. This is potentially problematic when a user study is being designed to keep security as a secondary goal. In this paper, we attempt to identify areas of explicit security so that we can better understand how they should be studied.

Keywords

End-user security, security automation, implicit security

INTRODUCTION

Security being a secondary rather than a primary goal for end-users provides ample justification for why automated security is desirable. With automation, users can focus on accomplishing their desired tasks rather than being distracted and delayed by tedious security tasks. However, not every aspect of security can be automated. Not only is the security landscape becoming increasingly complex, but the current system cannot “know” what the appropriate security action is for the user in every situation. The consequence of this is that users must inevitably deal with aspects of system security even if it remains a secondary goal. Fortunately, the “implicit security” approach as described by Grinter and Smetters [2] reduces the extent to which users must be involved in their system security. Actions which users already take based on their current task model can be used by the system to infer the security intentions and decisions. With the implicit approach, users do not need to perform security actions which are outside their model.

The benefit of the implicit security approach is that taking security actions is minimally disruptive to the user. The challenge, however, is that there must be a “mapping from application-tasks to implicit security actions” and this is difficult to ensure [2]. Unfortunately, it seems that the

intersection between the current system and existing user task models lead to instances where security actions which users must take do not follow this mapping. In fact, it seems there are a number of security actions which necessarily remain outside the scope of current user task models. We call such security actions “explicit”. For example, users must explicitly select anti-virus software to protect their whole system. Although this is a simple example, it illustrates the point that this action is not directly tied to a specific user task or application. Instead, this action is tied to the fact that the user’s system is connected to a network which enables them to do myriad non-security tasks and use a variety of applications.

We believe discussing explicit security actions is beneficial to an overall discussion of security user studies for several reasons. The first is that if security studies are designed beginning with a user’s task or application (i.e., in keeping with a more user-centered threat model), it is possible that explicit security actions can be overlooked since they do not easily or obviously follow the task-application mapping. Second, automating security or making it more implicit lessens the user’s burden. Therefore, identifying explicit security actions is the logical first step towards exploring how they can be automated or made more implicit if possible. Third, if it is found that such explicit security actions cannot be automated away or made more implicit, then a new approach may need to be explored for supporting users in such tasks.

In what follows, we begin by identifying what seem to be some examples of explicit security. We then provide some reasons why it is difficult to automate or make them implicit. It seems that explicit security tasks raise both the specific question of how explicit security actions should be studied and the general question of whether the whole of end-user security can be automated or made implicit. We believe that how these questions are answered can affect the underlying assumptions behind understanding how user studies addressing explicit security issues should be conducted. Our aim then is to provide a basis for discussing whether or not these are in fact necessarily explicit security actions and what user study approach to take in better understanding these issues.

SOURCES OF EXPLICIT SECURITY ACTIONS

There a number of reasons why explicit security actions exist outside the context of current user task models. One

commonly acknowledged reason is that user applications have been architected without considering how security can be incorporated to be more intuitive for the user [4]. However, using the implicit security approach can help with this and the work of Grinter and Smetters [2, 4] provide examples of how it can be accomplished. Therefore, in this section, we focus on other reasons (beyond that of poor application design) and provide examples of explicit security actions for users that seem to inevitably arise from them.

Constant Connectivity and Malicious Agents

Because developers often assume that security and other system related tasks are not a user's primary concern, they have automated where possible to reduce the user's burden. However, this automation requires constant connectivity to the Internet by system applications. As a result and unlike the previous decade, most users' systems now maintain a state of "constant connectivity". Many system applications now have automated updates and reporting features running in the background at startup, and often without users' knowledge. For example, Microsoft Windows XP downloads patches and updates automatically by connecting to its central server, and also sends anonymous usage statistics. It should be noted that similar automatic update features are included in popular user applications such as Adobe, Sun Java, Firefox, QuickTime, RealPlayer, and certainly on many others.

Despite the convenience that comes with this constant connectivity, there is also a cost. More specifically, the aggregate of the connectivity by both system and user applications causes security issues to emerge which in turn require the user to take additional explicit security actions. For instance, with constant connectivity, attackers have more opportunity for access to a user's system. These attackers can do any number of malicious acts that the system is unable to prevent or mitigate without the user's intervention.

For instance, attackers can exploit the Windows NetBIOS services which allows users to share files and other attached devices within the local area network (LAN). This service allows external devices to remotely access the file system of a user's device. As an operating system vulnerability, it allows unauthorized users to upload malicious files that may harm the device or open a backdoor for better control. This feature of the OS must be turned off by the user since the OS is policy-agnostic and cannot know a priori whether or not the service is needed by the system.

Exploits of OS vulnerabilities by malicious attackers are unrelated to a user's tasks or applications and therefore presumably should not have to be addressed by the user. However, perfect code to build perfect systems is far from being easily attainable, and security issues are often thrust upon the user as a result. This combined with constant connectivity enables automated but malicious background processes to work undetected and unhindered. Given that

current systems are unable to determine which are rogue processes, the user is needed to help make this determination. However, because these automated processes are hidden or running without users' knowledge, they are seldom aware of the threat to their system and what must be done to prevent or mitigate it.

User's System vs. User's Tasks/Applications

Another reason why explicit security actions exist for end-users is that there is a level at which users must interact with their system as a whole. It is true that users can still accomplish their tasks or use applications securely even if their overall system is not as secure [2]. However, there are instances where users cannot entirely ignore the security of their *whole system*. Like constant connectivity, these instances require users to consider security outside the context of their tasks and applications. Such instances arise when: 1) the security state of a user's whole system is in such jeopardy that its security consideration supersedes that of task or application considerations, or 2) the automated tools designed to protect the user's whole system require the user to take action because the system is unable to decide the necessary security action that must be taken.

An example of where the security state of the system must be considered foremost is when the user's system has been hijacked or become a zombie computer. Unknown to the user, the system is being controlled by an unknown attacker on the internet for malicious purposes such as sending out spam or being used for phishing scams. If and when the user does become aware that their computer is a zombie, it is in her best interest to take actions to rescue her system and not continue using it for her various tasks.

An example of an instance where the user must take action because her automated security tools cannot decide the necessary security action is firewall configurations. Again because a user's system is policy agnostic, it cannot decide such things as whether to block or allow a specified range of IP addresses. Although many policy setting actions are not directly relevant to a user task or application, only the user can make those decisions.

The security realities of constant connectivity, the persistence of malicious attackers, and the prevalence of OS vulnerabilities seem to impose upon the user a set of explicit security actions. To further illustrate this imposition, we provide a discussion of additional examples below where the explicit security actions have not yet been automated. Nor is it clear how to make them more implicit. Many of the examples which we further identify below are derived from what seems to be the user's new role of effectively being the system administrator [4] of their devices and home networks.

Additional Examples

1. *Safe downloads:* When downloading and executing files from an unknown source such as Internet sites, users must often decide if their intended download is

safe to execute on their system. A variety of information is available which they must aggregate and analyze to help inform their decision. However, most users are not knowledgeable enough to do this. We ask, can this security decision be automated or made implicit? On one hand, it seems possible to automate this decision by scanning the download as soon as it arrives on the user's system. Once the scan for malware is complete, the system could alert the user only if there is a problem. Unfortunately, anti-malware tools often suffer from a significant false-negative rate since there are many varieties of malware they do not cover as they are primarily signature-based. In the end, the automated scan turns out to be just one piece of information among others that a user can also consider in their determination of whether or not to execute a download on their system. It seems that the potential exists for automating the task of determining if it is safe to execute. However, doing so may require infrastructural changes at the industry and government policy level in order to effectively realize this level of automation, e.g., requiring all developers to obtain certificates before they can make open source available to the public.

On the other hand, one might consider how this explicit security task could be embedded in the user's action of opening the executable. Say for example, if the user double-clicks on the .exe icon, the system could infer that the user determined it was safe to execute the down and thereby making the secure decisions more implicit. This inference by the system however seems problematic because users often do not know whether it is safe to execute such downloads from unknown sources; and it is unclear which task model would be able to accommodate their security decision about the safety or lack thereof in opening the executable.

Again the difficulty of automating or making implicit the decision of whether a download is safe to execute raises the question of whether this task is necessarily explicit. It also raises the question of how it should be studied, i.e., does it make sense to treat it as an explicit task that will eventually go away? Or should it be treated as a task to be made implicit though that may require evolving the user task model to accommodate it? Or perhaps it should be treated as necessarily explicit and studies should be conducted to better support the user in this unavoidable security task?

2. *Verifying trust relationships with other computers or subnets:* As with the previous example, the ability to automate the task of determining which network or computer can be trusted for a connection seems difficult at best. It seems that the system needs the user to determine if the network or computer they are trying to access is indeed trustworthy or not. Although the system can rely on network profiling or anomaly detection to define heuristics for determining the

“trustworthiness” of a network or system, it is the user who must decide what the acceptable level of trustworthiness is for them. Further, it is the user who must decide if they want to trust a particular IP network always so that the trust relationships can be established automatically in the future. However, if this is the case, the system needs to be able to authenticate the networks the user is automatically being connected to with a high degree of accuracy which may not be attainable.

The question this situation raises is whether or not the user will be able to make a more accurate determination than the system about the trustworthiness of the system or subnet to which they are seeking to connect. However, unlike the system, the user will be able to weigh the relevant tradeoffs they need to consider in making their decision to connect. It would be extraordinarily challenging to weigh tradeoffs because the weights tend to vary frequently from situation to situation. Capturing the information about the user to determine what the correct tradeoffs are that the user is willing to make is extraordinarily challenging with the present system.

In considering how to make implicit the action of determining whether or not to trust an available system or network, again a user task model to use seems unavailable. Consider for example, when a user opens a browser or email client. This action is part of the existing task model of browsing for information. Perhaps this action could be interpreted as the user's intention to automatically connect to a secure network. However, in the real world, there are often many wireless access points available – some trustworthy and some rogue – but the system cannot necessarily know the difference and which it should trust on behalf of the user. Consequently, the user must tell the system which access point they trust and would like to a connection to rather leaving the system to infer it from their actions. It seems at best, the system can provide information to users to help support this task of access point selection since users may unwittingly opt to connect into a rogue network while remaining ignorant of the security risks of doing so.

3. *Setting network access policy for applications:* Policy setting for applications seems to be another area which requires users to take explicit security actions. In many cases, the system is unable to determine whether the connection to the internet was initiated by the user or by malware. For some cases, it seems possible to make implicit the security action of allowing an application to connect. A user's action of opening a gaming application should indicate to the system that the user's intention is to connect out through the relevant port.

However, malware is often incredibly adept at mimicking “good” system behavior making it difficult

for the system to be able to distinguish between user activity and malicious activity. This partly explains the high false positive and false negative rates plaguing automated security software. Consequently, users must be involved in helping the system determine if network access by applications is legitimate or not. In the current system, users are called upon to do this on a case-by-case basis and by setting general rules for applications. Unfortunately for most users, such tasks tend to be difficult and confusing.

There are other examples of explicit security which seem difficult to automate or make implicit such as system monitoring, privacy management, and defending against phishing attacks. What these seemingly necessarily explicit security tasks seem to have in common is that they involve decisions regarding trust and require weighing tradeoffs in the decision making. In contrast, when the security actions involve granting the system permission where the tradeoff considerations have already been completed, then the automation of such decisions or making them implicit seem much more feasible.

CONCLUSION

These examples serve to illustrate that there are aspects of end-user security which seem to remain squarely on the shoulders of users. It seems that these tasks may represent instances where security must necessarily become a primary goal for users. Furthermore, automating these tasks would probably come at the cost of major changes in policy and the underlying infrastructure while giving no guarantees about the accuracy of such automation in reflecting user intentions about security. On the other hand, what would be the cost of making such tasks more implicit especially when user task models seem unable to accommodate these explicit actions? Or perhaps there is another approach for understanding these seeming explicit security tasks – another perspective that might show currently hidden ways to automate or make implicit.

Based on the discussion in the previous section, it seems worthwhile to ask if seemingly explicit security tasks need to be considered differently in studies because they require actions that lie outside user task models. In fact, perhaps we should consider if there a way to evolve task models so that user intentions regarding their security can be more accurately captured. Furthermore, we could also consider the extent to which security is automated for end-users. How much can we assume that the mechanisms for ensuring user security appropriately reflect the tradeoffs users are would make? For example, some users may be willing to risk a little bit of security and privacy for a greater gain in convenience. Others, however, may prefer to do the exact opposite.

The implication for user studies is that when studying explicit security tasks, three questions should probably be

asked. The first is “Can these tasks be automated away?” If the answer is yes, then perhaps the user study needs to be tailored to make certain that in automating such tasks, the tradeoffs users would be willing to accept are accurately reflected. If the answer is no, then a second question should be asked which is “Can these tasks be made more implicit?” If the answer is yes, then perhaps the user study needs to be tailored such that user intentions regarding their security are being accurately captured. This is challenging since it is difficult to capture the tradeoffs users are willing to make by looking at user actions and task model. If the answer is no, then a third question that should be asked is “Can the user’s task model be evolved so that it can accommodate these explicit security tasks?” If yes, then perhaps the user study design should consider how this might be done and what supports the user would need to develop or evolve their task model. If no, then the user studies should perhaps focus on better understanding how to support explicit security tasks that require users to consider tradeoffs.

Asking these questions and seeking an answer will most likely lead to a better understanding of explicit security tasks. It seems that we cannot easily assume that the whole of end-user security can be automated away nor made implicit with the given current system and task models. In fact it seems highly probable that explicit security tasks will continue to persist for the end-user. However, perhaps we can use explicit security tasks as opportunities to explore new study directions such as task model evolution or how to support end-user security risk assessment under extreme uncertainty.

REFERENCES

1. Burns, Steven F. Threat Modeling: A Process To Ensure Application Security. SANS Institute. January 5 2005. Available as http://www.sans.org/reading_room/whitepapers/securecode/1646.php
2. Rebecca E. Grinter and D. K. Smetters. Three Challenges for Embedding Security into Applications. Presented at the CHI'03 workshop on HCI and Security Systems, April 6, 2003. Available as: <http://www.andrewpatrick.ca/CHI2003/HCISEC/hcisec-workshop-grinter.pdf>.
3. S. Myagmar, A. J. Lee, and W. Yurcik. Threat Modeling as a Basis for Security Requirements. In Symposium on Requirements Engineering for Information Security (SREIS), 2005.
4. Smetters, D. K. and Grinter, R. E. 2002. Moving from the design of usable security technologies to the design of useful secure applications. *New Security Paradigms Workshop* (Virginia Beach, VA, Sept. 23 - 26, 2002).