Helping Engineers Design NEAT Security Warnings

Rob Reeder, Ellen Cram Kowalczyk, and Adam Shostack
Microsoft
Redmond, WA, USA
{roreeder, ellencr, adam.shostack}@microsoft.com

ABSTRACT

Software engineers who design large systems have a multitude of concerns to address before shipping their software. Usability and security are merely two of these concerns, and usable security is a small slice of those. Thus, software engineers can only be expected to spend a small fraction of their time on usable security concerns. Our team, the Usable Security team in Microsoft Trustworthy Computing, acts as a central resource for product teams. We have been working to help them use the latest knowledge from the usable security community to design security warnings. Because these engineers have so many demands on their time, we have had to condense our guidance into a short, easily consumed form. In fact, we have condensed it to four letters: NEAT. A good security warning should be Necessary, Explained, Actionable, and Tested. With these four letters and the training materials we have built around them, engineers are able to comprehend and use the latest usable security results.

INTRODUCTION

Computer users face a barrage of decisions about who and what to "trust," and when to be concerned about their security. These security decisions arise when users initiate activities like visiting a website, installing an executable from the Web, or using an application that needs to get through the firewall. Security decisions are surfaced to users by a platform – for example, an operating system or a Web browser. Designers of platforms design the experience users go through when making trust decisions, and this user experience can lead users to make better or worse decisions, depending on how they are designed.

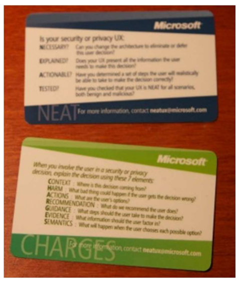
Our team at Microsoft, the Usable Security team, was formed to help engineers within Microsoft design better user experiences for making security decisions. This paper is a part of the story of how we have done that and how we help engineers today. We believe the approach is likely usable at other organizations, or by researchers analyzing the usable security of systems.

DEVELOPING GUIDANCE FOR ENGINEERS

Our team's first task was to gather the usable security knowledge that we would encourage Microsoft engineers to follow. We gathered a group of internal Microsoft experts in both security and usability to help determine what that knowledge should be. Initially, the group surveyed the need for usable security advice by inviting product teams with plans for security-related features to present those features to the group and receive expert feedback on the user experiences in those plans. Through these sessions, the group learned what usable security questions the teams needed answers to. Key questions included:

- When is it appropriate to interrupt users with a warning dialog to ask security questions?
- When presenting a security question to a user with a dialog, how should the dialog user interface be designed?

After several of these sessions, the group began an effort to gather the knowledge to share with teams. To gather this knowledge, the group drew upon internal and external usable security research as well as insights gained from the presentations by product teams. Since usable security is still a nascent field, this process was not easy; there are many competing ideas and many gaps in knowledge that make it difficult to gather a definitive set of knowledge to share with engineers. Existing literature was seen as too remote from the day-to-day needs of engineers.



Microsoft

Is your security or privacy UX:

NECESSARY? Can you change the architecture to eliminate or defer this user decision?

EXPLAINED? Does your UX present all the information the user needs to make this decision?

ACTIONABLE? Have you determined a set of steps the user will realistically be able to take to make the decision correctly?

TESTED? Have you checked that your UX is NEAT for all scenarios, both benign and malicious?

NEAT For more information, contact neatux@microsoft.com

Microsoft

When you involve the user in a security or privacy decision, explain the decision using these 7 elements:

CONTEXT : Where is this decision coming from?

HARM : What bad thing could happen if the user gets the decision wrong?

ACTIONS : What are the user's options?

RECOMMENDATION : What do we recommend the user does?

GUIDANCE : What steps should the user take to make the decision?

EVIDENCE : What information should the user factor in?

SEMANTICS : What will happen when the user chooses each possible option?

CHARGES For more information, contact neatux@microsoft.com

Figure 1. Wallet-sized cards summarizing our first-version usable security guidance.

Ultimately, the group produced a paper that captured a consensus view of the most important aspects of knowledge about designing usable security warnings to share as guidance with engineers. The paper consisted of 24 pages, with 68 items of advice arranged into a hierarchy 3 levels deep. Having produced the paper, we showed it to a few engineers to see what they thought. We quickly saw we had a significant problem: Microsoft engineers do not have time in their day to read 24 pages and 68 bullet points about usable security. The list of concerns for a Microsoft engineer is long; it includes functionality, performance, reliability, localization, accessibility, backward compatibility, and maintainability, just to name a few. Security and usability are both on this list, to be sure, but usable security is only a tiny slice of usability (most of a product's user experience has nothing to do with security) and a tiny slice of security (security includes both the development of security-related features and product-wide activities like threat modeling and penetration testing). Time for usable security is thus very limited.

So, our team took on a second task to simplify our usable security guidance. As we confronted this second task, we also sought to satisfy a second goal: raising awareness of the importance of usable security. Since the field is still nascent, not all engineers have been exposed to it. We saw an opportunity as we simplified our guidance to both make it easier and faster for engineers to consume and also to make it more memorable by inventing a convenient mnemonic. The mnemonic we came up with is a nifty acronym: NEAT.

NEAT: WHAT SECURITY WARNINGS SHOULD BE

As we reviewed our 24 pages of guidance with its 68 bullet points, a few stood out as particularly important to help answer the key questions product teams had about how to design good security warnings. We took these key points and condensed them into NEAT. The core message of NEAT is that a security warning should be:

- **Necessary**: A warning should only interrupt a user if it is absolutely necessary to involve the user. Sometimes, a system can automatically take a safe course of action without interrupting the user. Sometimes, a security decision can be deferred to a later point in time.
- **Explained**: If it is actually necessary to interrupt the user with a security warning, the warning should explain the decision the user needs to make and provide the user with all the information necessary to enable them to make a good decision. Since the Explained part of NEAT is perhaps the most important, we devised another acronym, CHARGE US (see below), to help engineers remember what information to provide in a security warning.
- **Actionable**: A security warning should only be presented to the user if there is a set of steps the user could realistically take to make the right decision in all scenarios, both benign (where there is no attack present) and malicious (where an attack is present).
- **Tested**: Security warnings should be tested by all means available, including visual inspection by many eyes and formal usability testing.

For the Explained part of NEAT, we include the acronym CHARGE US, to represent eight of the key elements of a well-explained security warning:

- **Context**: An explanation of the source of a decision – the application that raised it and the item (file, website, etc.) the user is being asked to trust.
- **Harm**: An explanation of the potential consequences of getting the decision wrong.
- **Actions**: A list of options the user has.
- **Recommendation**: A recommendation from the system about what to do; usually this means recommending the user choose the safer option.
- **Guidance**: A series of steps the user can take to make a good decision, and a clear statement of the knowledge the user has that might help make the best decision (e.g., sometimes knowing what the user is trying to accomplish can help the system make a better decision).
- **Evidence**: Any information the user should factor into their decision; e.g., if this is a decision about whether to run a program, the program's publisher is an important piece of evidence.
- **Unique** knowledge user has: Warnings often occur because the engineer expects the user to have some specific contextual information that the system does not. That information should be explicitly identified and communicated to the user either implicitly or explicitly, e.g., is this network you're connecting to at home, at work, or at an airport?
- **Semantics**: A clear statement of what will happen for each option the user may choose.

To promote our NEAT guidance, we have developed training materials to help engineers remember NEAT and dig deeper into the details of our guidance if they need to. We have produced handy wallet-sized cards with the NEAT and CHARGES (since updated to CHARGE US) acronyms on each side of the card along with text to explain them (see Figure 1). We have developed a one-hour talk we deliver to product teams and an extensive slide deck with detailed examples that engineers can use on their own. We have a checklist that engineers can use to ensure they have followed all of the aspects of NEAT, and we have shared a bug bar with teams to help them prioritize usable-security-related work items.

CONCLUSION

Our NEAT guidance is now in use by product teams at Microsoft. We often teach NEAT to interested engineers. It can be taught in about an hour, and we find that engineers remember the acronym, or at least remember that there is an acronym. In any case, our guidance and training have raised awareness of usable security at Microsoft. The NEAT guidance is a scalable way for us to share our expertise in usable security with product teams, as it gives them an easy way to remember and apply knowledge from usable security research.

A key lesson we've learned in our experience with NEAT is that to integrate usable security (or any discipline) into the software development lifecycle, it is important to make it as easy as possible for busy engineers to follow the advice we give them. There is great value in translating the results from research experiments into actionable takeaways for engineers. NEAT, along with its associated materials, has been a great first step in helping engineers follow the tenets of usable security.