Eight Lightweight Usable Security Principles for Developers

Peter Leo Gorski | INFODAS GmbH
Luigi Lo Iacono | H-BRS University of Applied Sciences
Matthew Smith | University of Bonn, Fraunhofer FKIE

We propose eight usable security principles that provide software developers with a lightweight framework to help them integrate security in a user-friendly way. These principles should help developers who must weigh usability and security tradeoffs to facilitate adoption.

Cybersecurity is a basic precondition for our digital society and should work for everyone, from children to grandparents, from citizens to policymakers, from industry to the military and so on. Security must also work in a wide range of domains, from casual personal use to critical infrastructures, each with its own set of requirements. There are many technical security measures that can be put in place to fulfill these requirements. However, there are major challenges in designing and integrating such measures into systems so that they are used correctly, or at all. When security failures inevitably occur, human error is often identified as the cause. For some security experts and developers, this has shaped the idea that users are the weakest link. There is also the related misconception that security and usability are incompatible qualities of a digital system. The growing discipline of "usable security" addresses this myth[1] by developing scientific insights into how security and usability can be reconciled and how existing conflicts of requirements can be balanced from the perspective of the overarching goal of "effective cybersecurity."

Since its emergence some 25 years ago, the field of usable security has offered many important insights into how security features can be aligned with users' needs, abilities, and expectations. However, the transition of these findings into practice has been less well explored. Key players in this regard are software developers as they need to integrate security mechanisms into products in a way that meets security requirements but also fit users' capabilities. However, as security is rarely the main purpose or selling point of a product, developers have to balance the effort for themselves and their users. In some cases, (like plain email) very little security was implemented. In others (like PGP), the implementation was so complex that adoption was minimal. Both situations are not ideal.

The body of knowledge in the field of usable security, which could help address this issue, is often quite specific. This makes it a less-than-ideal source of guidance for many software developers with limited resources, who cannot, for instance, set up and run complex security user studies. To see a broader adoption of usable

Digital Object Identifier 10.1109/MSEC.2022.3205484
Date of current version: 7 October 2022

security mechanisms, the challenge is to integrate existing knowledge from research into software development in a lightweight manner that is easily accessible to developers.

An effective way of sharing insights from research with developers is aggregating the experience of domain experts into guiding principles. These broad rules of thumb have proven helpful, e.g., in the form of security principles[2] and usability principles.[3,4]

There are also first collections of usable security principles: for example, Green and Smith present 10 principles for the development of usable cryptographic libraries;[5] Acar et al. do not directly present principles, but they identified key lessons learned from usable security for end users that can serve as principles for developers;[6] and Gorski et al. conducted a literature review of usable security principles from which they constructed a set of 23 principles.[7] However, these existing resources are either very specialized or very comprehensive, requiring a fair amount of work and specialized knowledge to implement.

In this article, we propose a collection of eight usable security principles that provide developers with a lightweight and practical framework for thinking about how to integrate security in an end-user-friendly way. As such, the principles are not geared toward critical or high-security domains, where security concerns can trump usability and adoption of security mechanisms can be mandated. Instead, these principles are aimed at helping developers who want good security but must weigh the tradeoffs and facilitate adoption.

In this, we were inspired by Garfinkel's second principle of "Good Security Now,"[8] i.e., we want these principles to drive adoption of the usable security mindset for as many developers as possible now. To this end, we developed our principles based on those gathered in the work of Gorski et al.[7] However, we wanted our principles to be more general and lightweight so as to be easily memorable for developers. Nonetheless, where possible, we refer to more specialized and comprehensive usable security principles to facilitate more in-depth research. Methodologically, we each independently created a list of 10 principles we would recommend to software developers. We then compared and discussed our three lists on the basis of literature and application examples and condensed them into the eight principles we propose here.

Usable Security Principles

The presented eight principles (see Table 1) are meant to help developers design more usable security mechanisms for end users. Not all principles will be applicable to all situations, and there are certainly application contexts where security requirements can demand more restrictions in usability. However, even in situations where security might trump usability, we believe it is useful to critically assess whether any of the principles can be applied as usability does not have to be at odds with security. If done right, good usability can and will increase security.

Principles

1. Bake It In

If possible, integrate security mechanisms in a way that the users do not need to interact with them (cf. "Path of Least Resistance"[7]). The correlation between decision making and the effort to act is a broad subject of study in the field of neuroscience.[9] Similar to the fact that every additional click needed to purchase something reduces the number of sales, every step required by a security mechanism reduces the number of users willing to use it. It also often introduces points of failure where users can make mistakes or forget to activate security entirely.

Good examples of this principle are the implementations of end-to-end encryption enabled by default in messaging systems such as iMessage or WhatsApp. These systems indicate that messages and calls are encrypted at the beginning of every conversation, but users do not have to do anything to enable this (see Figure 1). Compared to PGP and S/MIME this zero-interaction encryption led to billions of users

Table 1. An overview of the proposed usable security principles.

Number	Usable security principle: A short description
1	Bake it in; try to integrate security so that your users don't have to interact with or put effort into it.
2	Don't maximize security at the cost of usability; the best security is of no use if people do not use it.
3	Offer more security to those who want it; enable power users without burdening everyone else.
4	Protect the needs of the many with the expertise of the few; enabling experts to detect attacks might be able to deter attacks in general.
5	Make the language simpler than you think is necessary; many words and concepts that are well known to you are not well understood by your end users.
6	Use personal examples; it makes otherwise abstract concepts much more tangible.
7	Be mindful when delegating decisions to your user; if it's too hard for you to automate, it's probably too hard to decide for many of your users.
8	Gather users' mental models and build your system to address their misconception; talk to your users about their understanding of a system or concept, you'll be surprised.

having their messages protected, while the manual effort and expertise needed to use PGP or S/MIME has led to the longest-running series of usability failures,[11] leaving email security vastly unused.

The challenges for developers are to balance user motivation and security goals, reduce the physical and mental effort required to apply a security feature, and implement security defaults without making them invisible (cf. "Visibility"[7]).

2. Don't Maximize Security at the Cost of Usability

Ideally, security and usability do not have to be at odds.[1] However, it is not uncommon for higher levels of security to increase the burden on the end user, and even small burdens can lead to big problems (cf. "Convenience"[7]). Following the aphorism "perfect is the enemy of good," if there is a tradeoff between usability and security, consider carefully how many users you might lose or how many might make mistakes on the road to perfect security (cf. "Good Security Now"[7] and "More Is Not Always Better"[6]).

The messaging/email example from principle 1 serves here as well. Many current messengers use a centralized key management infrastructure that requires trust in the companies running them to not manipulate the keys. From a security perspective, this is not as good as the manual/web-of-trust model used in PGP. However, the central key management scheme simplified the end-to-end encryption so significantly that everybody using these messaging apps can use it. Thus, adoption is much higher compared to the manual key management of PGP, for example.

A negative example of this principle is overly complicated password policies (see Figure 2). Here, the system tries to force better security, which leads to many evasion strategies and errors.

3. Offer More Security to Those Who Want It

Principles 1 and 2 argue for zero- or low-effort security to be the default setting for the majority of users, even at the cost of some (theoretical) security. However, if it

Figure 1. Bake it in: end-to-end encryption baked into the WhatsApp messenger.[10]

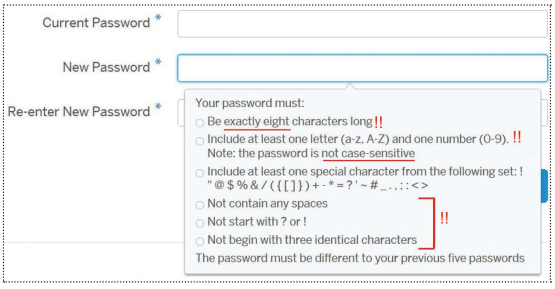

Figure 2. Don't maximize security at the cost of usability: the negative example of a complex and ineffective password policy.[12]

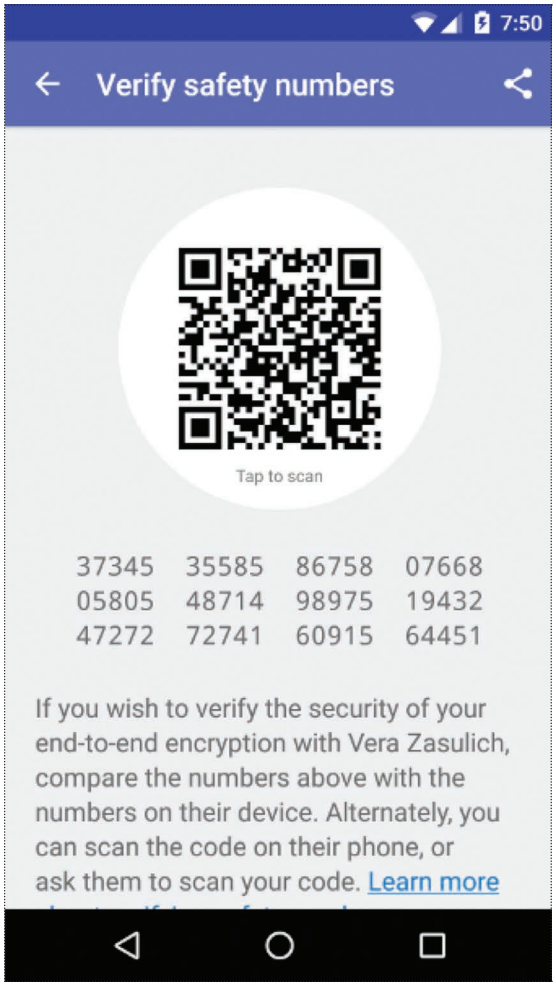

Figure 3. Offer more security to those who want it: a security measure offered by Signal to ensure the authenticity of communication partners, which can be accessed and used only by tapping the user profile icon. (Source: "Safety number updates", 17 Nov 2016, https://www.signal.org/blog/safety-number-updates/.)

can be integrated without violating principles 1 and 2, adding additional options and safeguards for motivated power users is a good idea (cf. "Appropriate Boundaries" and "Expressiveness"[7]).

Messenger apps like Signal allow users to manually verify public keys to ensure the authenticity of communication partners. This increases security for those willing to invest time and effort in key management and code audits. To view and verify a safety number, the user must tap on the icon of their user profile to access and use this additional security measure (see Figure 3).

4. Protect the Needs of the Many With the Expertise of the few

To mitigate potential security downsides of principles 1 and 2, it is worth considering whether a security mechanism can be designed in such a way that the vigilance of a small group of motivated experts can guard the majority of regular users. The fact that power users can check key fingerprints in Signal means that mass surveillance would be noticed, and thus the capability of the few protects the system as a whole. The ability to review source code also falls into this category (cf. "Expertise"[7]). Figure 4 shows Signal's source code repository, offering the possibility for experts to conduct code reviews.

Certificate Transparency (CT) (https://certificate.transparency.dev/) is another good example of this principle. In a nutshell, a current security weakness in the Certificate Authority (CA) server certificate system is that compromised or rogue CAs can create certificates for any domain name. Thus, the CA system is a weakest-link system and enables man-in-the-middle (MITM) attacks that go undetected by the majority of users. Improving the security of the CA system without adding undue burden on the end users has proved challenging. The CT approach adds a layer of protection for all users without burdening the regular user at all. With CT, certificates are publicly logged and a small group of experts can monitor these logs to detect malicious certificates. Although this does not directly protect against MITM attacks, it disincentivizes rogue CAs and detects compromised CAs, which improves the overall security of the system with no additional knowledge or work required by the regular users.

5. Make the Language Simpler Than You Think Is Necessary

"Speak the User's Language" is a well-known usability principle and is particularly relevant for the usability of security as well (cf. "Understandability"[7]). Years of experience can make us blind to the fact that many words that seem simple to us, like *authenticate, authorize, private key, public key, certificates, encryption, signature,* or *policy* might be unknown, only vaguely understood, or carry nontechnical meanings to many users. Thus, implementing the general "Keep It Simple" principle is highly recommendable in a usable security context. Try to make things as simple as you can and then make another pass to make it even simpler. It is also a good idea to let people outside your regular work or social circle read the texts and repeat, in their own words, what they think they mean. This ties in with principle 8. The difficulty is preserving accuracy at the same time. This principle can also be combined with principle 3 by offering alternative texts or additional information for experts.

Security dialogues in messaging apps have undergone some evolution in terms of the language used to communicate complex security matters. We can also use Figure 3 as an example here, as it describes an authentication ceremony by comparing (the fingerprints of) public keys without using the usual technical terms in the description, such as authentication and public key.

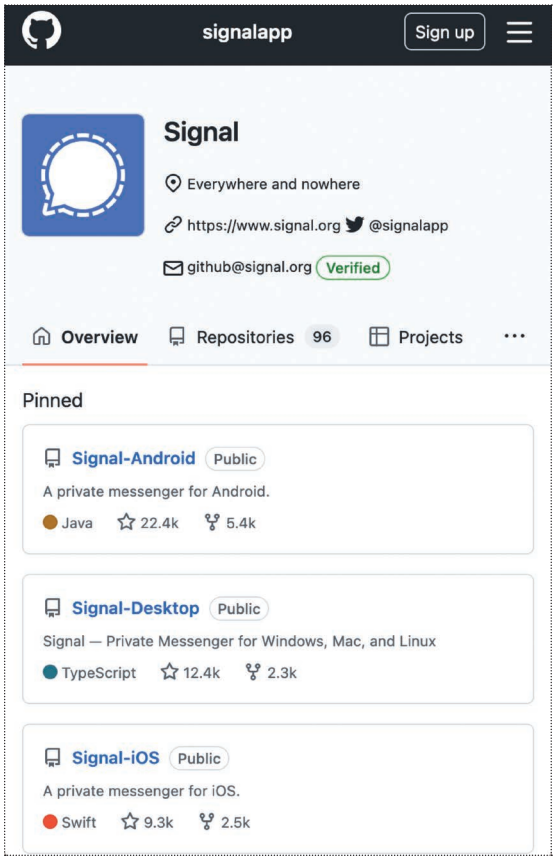

Figure 4. Protect the needs of the many with the expertise of the few: Signal transparently publishes source code so that experts can review it. (Source: https://github.com/signalapp.)

Another example is the icons used in web browsers to inform about a secure connection to a web server. Garron and Palmer from the Chrome security team explain: "We have to strike a balance: representing the security state of a webpage as accurately as possible, while making sure users are not overwhelmed with too many possible states and details. We've come to understand that our yellow 'caution triangle' badge can be confusing when compared to the HTTP page icon, and we believe that it is better not to emphasize the difference in security between these two states to most users."[13]

Chrome 67 added the term *Secure* to the website security icon in an attempt to make it simpler and more understandable (see Figure 5). However, it could still be misinterpreted by users, like being secure against Internet fraud, which is not correct. Since Chrome 69, only a gray padlock icon has been implemented, and developers are thinking about removing the icon altogether.

6. Use Personal Examples

Abstract security and privacy concepts are often hard to grasp. Principle 5 already takes this into account and recommends making the language easy to understand. To help regular users understand and assess their own actions and the potential consequences, show them, if possible, personalized examples of what settings mean or what the consequences of an action are (cf. "Clarity" and "Understandability"[7] and Harbach et al.[14]).

Facebook's "View As" feature, which allows users to easily see what information they are making available to the public, is a good example of this principle (see Figure 6). The concrete nature of an example is often easier to understand than abstract rules, and using the actual personal data makes it more salient.

7. Be Mindful When Delegating Decisions to Your User

When designing interaction with the users, consider whether the user is likely to have the expertise and

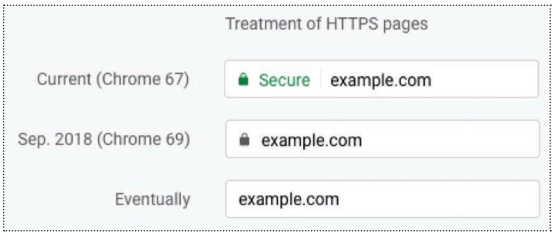

Figure 5. Make the language simpler than you think is necessary: Chrome completely discarded the term *Secure* from the website security icon.[13]

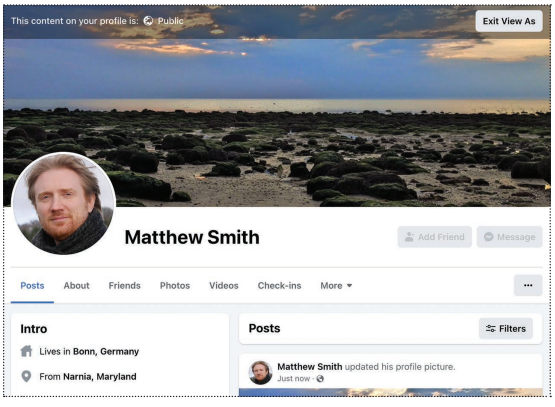

Figure 6. Use personal examples: Facebook allows users to preview the effect of their privacy settings with a "View As" feature.

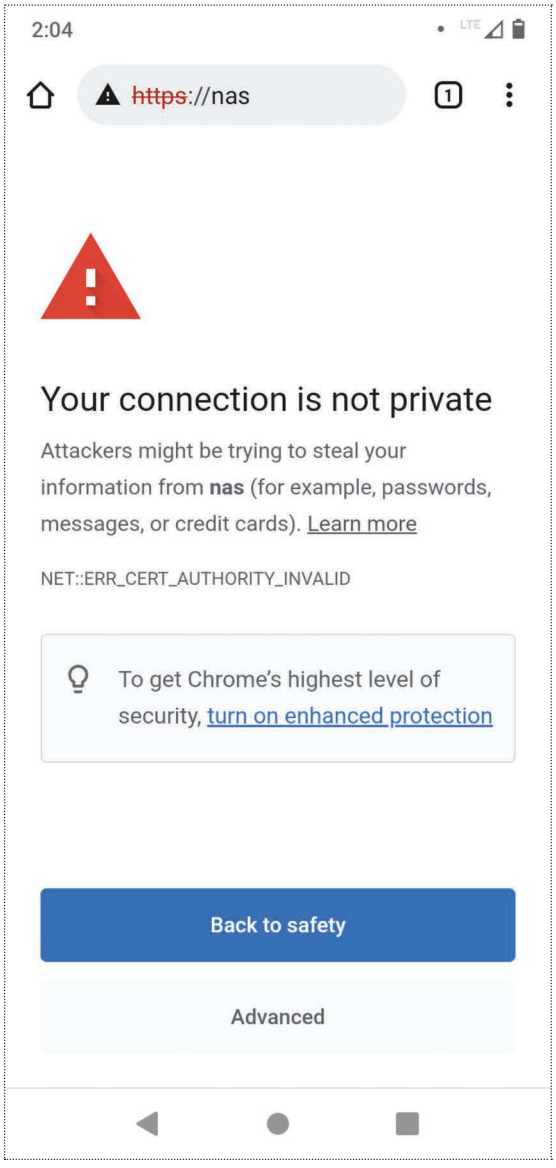

Figure 7. Be mindful when delegating decisions to your user: a TLS warning in Chrome caused by a self-signed certificate.

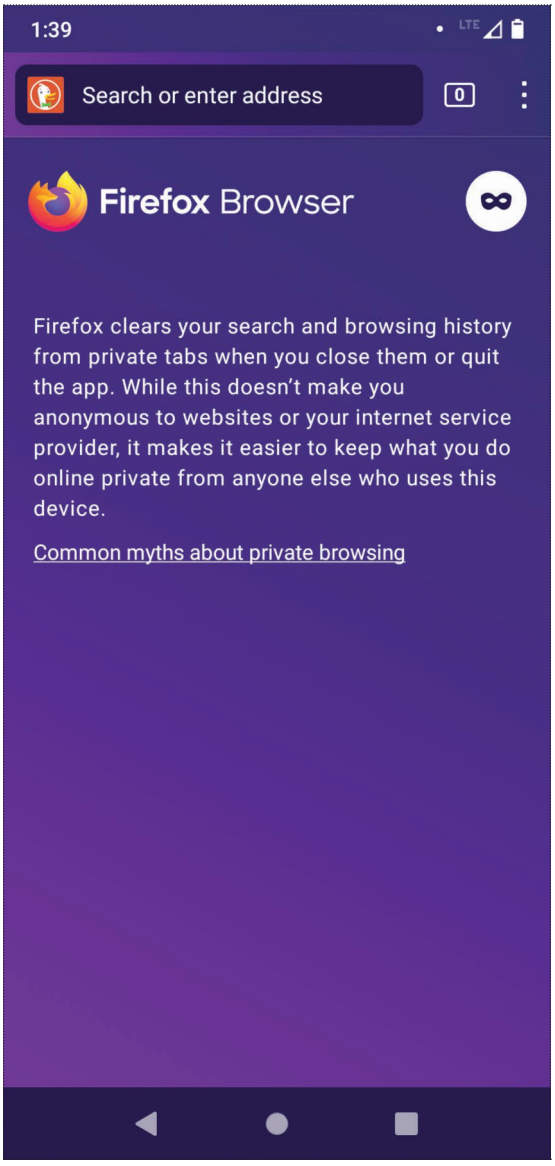

Figure 8. Gather users' mental models and build your system to address their misconceptions: Firefox explains the features of private browsing to eliminate common user misconceptions.

motivation to make a good decision. Transport Layer Security warnings are a negative example. Well-executed MITM attacks utilize a valid certificate and do not display a warning. For most users, warnings shown in response to less well-resourced attacks (e.g., using a self-signed certificate) are indistinguishable from warnings due to misconfigured servers (see Figure 7). Consequently, giving users an override is dangerous. If you, the expert, cannot codify a set of rules to automatically resolve the interaction, it is an indication that the decision might be too complex for the end user. Removing the user override moves the burden from the end user to those administrators who misconfigure their servers. This step would also fulfill principle 4. At this point, it is important to highlight that administrators need (domain specific) usable security measures too.

8. Gather Users' Mental Models and Build Your System to Address Their Misconceptions

Even if we follow principles 1–7, it is still possible that users will misunderstand security issues and/or make mistakes when interacting with security mechanisms (cf. "Least Surprise"[7]). Ideally, full-blown user studies can be conducted to observe mistakes and build mental models specific to the security mechanism and the user group. However, if this is beyond the resources available, reading about existing mental models in related fields or having informal conversations with users can already provide valuable insights into what can go wrong. It can be a good strategy to directly address the misunderstandings you encounter.

The message explaining the limitations of private browsing (i.e., that it does not make the user anonymous) is a good example of addressing common misconceptions.[15] Figure 8 illustrates the ways in which the Firefox browser tries to explain what private browsing actually means.

We proposed this set of eight usable security principles for developers who want to integrate or improve security mechanisms in a user friendly manner. They are not meant to be exhaustive but rather to be simple and lightweight so that many developers can adopt them. They should raise awareness that good usability can improve existing security mechanisms and introduce them where they are currently lacking. They are also meant to be a starting point for more comprehensive or specific resources[7] where we collect these and other usable security principles. To this end, we maintain the usablesecurityprinciples.dev website. ■

Acknowledgment

We thank Jan Tolsdorf and Stephan Wiefling as well as the anonymous reviewers for their guidance and insightful comments. This research was partially funded by European Research Council Grant 678341: Frontiers of Usable Security.

References

1. M. A. Sasse, M. Smith, C. Herley, H. Lipford, and K. Vaniea, "Debunking security-usability tradeoff myths," *IEEE Security Privacy*, vol. 14, no. 5, pp. 33–39, Sep./Oct. 2016, doi: 10.1109/MSP.2016.110.
2. J. H. Saltzer and M. D. Schroeder, "The protection of in formation in computer systems," *Proc. IEEE*, vol. 63,

no. 9, pp. 1278–1308, Sep. 1975, doi: 10.1109/PROC.1975.9939.

3. J. Nielsen, "Heuristic evaluation," in *Usability Inspection Methods*, J. Nielsen and R. L. Mack, Eds., New York, NY, USA: Wiley, 1994.
4. Ergonomics of Human-system Interaction — Part 110: Interaction Principles, ISO 9241-110, International Organization for Standardization, Geneva, Switzerland, 2020.
5. M. Green and M. Smith, "Developers are not the enemy!: The Need for Usable Security APIs," *IEEE Security Privacy*, vol. 14, no. 5, pp. 40–46, Sep./Oct. 2016, doi: 10.1109/MSP.2016.111.
6. Y. Acar, S. Fahl, and M. L. Mazurek, "You are not your developer, either: A research agenda for usable security and privacy research beyond end users," in *Proc. IEEE Cybersecurity Develop. (SecDev)*, 2016, pp. 3–8, doi: 10.1109/SecDev.2016.013.
7. P. L. Gorski, E. von Zezschwitz, L. Lo Iacono, and M. Smith, "On providing systematized access to consolidated principles, guidelines and patterns for usable security research and development," *J. Cybersecurity*, vol. 5, no. 1, pp. 1–19, Dec. 2019, doi: 10.1093/cybsec/tyz014.
8. S. Garfinkel, "Design Principles and Patterns for Computer Systems That Are Simultaneously Secure and Usable," Ph.D. thesis, Massachusetts Inst. of Technol., Cambridge, USA, 2005.
9. N. Hagura, P. Haggard, and J. Diedrichsen, "Perceptual decisions are biased by the cost to act," *eLife*, vol. 6, p. e18422, Feb. 2017, doi: 10.7554/eLife.18422.
10. "About end-to-end encryption." WhatsApp Help Center. [Online]. Available: https://faq.whatsapp.com/general/security-and-privacy/end-to-end-encryption/
11. A. Whitten and J. D. Tygar, "Why Johnny can't encrypt: A usability evaluation of PGP 5.0," in *Proc. 8th USENIX Secur. Symp. (USENIX Security)*, 1999, pp. 169–183.
12. "Ted sales tweet." Twitter. [Online]. Available: https://twitter.com/tedworthy_/status/751365313149726720
13. "Evolving Chrome's security indicators." Chromium. [Online]. Available: https://blog.chromium.org/2018/05/evolving-chromes-security-indicators.html
14. M. Harbach, M. Hettig, S. Weber, and M. Smith, "Using personal examples to improve risk communication for security & privacy decisions," in *Proc. SIGCHI Conf. Human Factors Comput. Syst. (CHI)*, 2014, pp. 2647–2656, doi: 10.1145/2556288.2556978.
15. H. Habib et al., "Away from prying eyes: Analyzing usage and understanding of private browsing," in *Proc. 14th Symp. Usable Privacy Secur. (SOUPS)*, 2018, pp. 159–175.

Peter Leo Gorski leads a research group at INFODAS GmbH, Köln 50765, Germany. His research interests include security-enhancing technologies for critical infrastructures and software development processes. Gorski received a Ph.D. in computer science from TU Berlin, Germany. Contact him at p.gorski@infodas.de.

Luigi Lo Iacono leads the Data and Application Security Group at H-BRS University of Applied Sciences, Sankt Augustin 53757, Germany. His research interests include security- and privacy-enhancing technologies for distributed software systems with a particular focus on their usability. Lo Iacono received a Ph.D. in computer science from University of Siegen, Germany. Contact him at luigi.lo_iacono@h-brs.de.

Matthew Smith leads the Behavioural Security Research Group at the University of Bonn, Bonn 53113, Germany. His research interests lie at the intersection of technical IT security and privacy and behavioral research with a special focus on security workers such as developers and administrators. Smith received a Ph.D. in computer science from the University of Marburg, Germany. Contact him at smith@cs.uni-bonn.de.