Usable-Security Evaluation

Yasser M. Hausawi[1(✉)] and William H. Allen[2]

[1] Institute of Public Administration, Jeddah, Saudi Arabia
hawsawiy@ipa.edu.sa
[2] Florida Institute of Technology, Melbourne, FL, USA
wallen@fit.edu

Abstract. Developing software products which align security and usability to make a synergistic relationship between security and usability is an engineering process that starts from the first phase of the Software Development Life-Cycle (SDLC), and continues through the rest of the phases: design, construction, and testing. However, a summative evaluation of such a process must be done after the software product is completely developed with careful attention to measuring the alignment between security and usability (i.e.: usable-security), and integrating such alignment properly within the SDLC. Therefore, this paper proposes a usable-security measuring matrix that provides a summative evaluation of the whole process of applying usable-security on software products.

Keywords: Security · Usability · Human computer interaction · HCI · HCI-SEC · Usable security · Quality attributes evaluation

1 Introduction

As a result of the increased adoption of a user-centered approach to improve human-computer interaction and increasing concerns regarding computer-related privacy and security issues, researchers and scientists have proposed many design techniques that incorporate usable-security [11,15]. Those techniques are used to build software systems that seek to balance usability and security. However, if we take a careful look at the currently available software systems, we often find that quality attributes, such as security and usability, are applied and evaluated separately [3,10,20].

Therefore, the vast majority of those systems are built and deployed without giving proper attention to evaluating the state of integrating the quality attributes, such as usable-security. It is very important to be sure that software systems are usable and secure enough [15,19], so that people can rely on such systems and interact with them as if interacting with actual human beings. People ought to be able to use and deal with systems with hands-on ease, as this helps in a better cooperation between the people and the systems [9,10]. Also, people must be able to trust that the systems are secure enough against

T. Tryfonas and I. Askoxylakis (Eds.): HAS 2015, LNCS 9190, pp. 335–346, 2015.
DOI: 10.1007/978-3-319-20376-8_30

any malicious penetration behavior that may cause any intentional undesired change or update during the human-computer interaction [2,22].

Unfortunately, without careful consideration of the balance between usability and security, a system can be designed so that those properties will actually work against each other and cannot be aligned and evaluated together as one joint concept [1,16,25]. One approach that has proven successful combines usability and security with the goal of improving the performance and quality of computer systems [8]. Therefore, we adopted the approach of bridging usability and security attributes together as one combined attribute: usable-security. One way to achieve that goal is using models and frameworks that help in assessing software usability, security, and usable-security requirements based on formative evaluation approach [10,13,20]. Assessing usable-security helps in predicting to which level software systems could be simultaneously usable and secure [13]. This work proposes a summative usable-security evaluation matrix based on the usable-security assessment models and frameworks. The matrix represents a solid component of a large scale ongoing research project for designing a Usable-Security Engineering Framework (USEF) for enhancing software development when balancing security and usability is an important matter.

Section 2 presents background on how important usability and security are to software systems and Sect. 3 introduces a proposed mathematical-based evaluation model to help determining the levels of usability and security quality attributes. Section 4 introduces a proposed matrix to evaluate usable-security of software systems. The last section is a conclusion about applying the matrix.

2 Background

Despite the success of most of the available systems in making human life even easier, they need continual evaluation and enhancement to make sure that such systems meet the usability and security requirements and standards [4,17].

2.1 Usability

One usability concern is that software systems must be effective. For example, when airline companies use automatic speech recognition systems for phone reservations, those systems can only be considered effective if they are able to convert a traveler's speech into a textual representation that can be then processed as accurately as a textual request. Another usability concern is that the systems must be efficient enough to save processing time for both users and business owners. However, there are many examples where the extended time of the sound representation process negatively impacts the speed of the automatic speech recognition systems' ability to access customers' information. Consequently, lack of effectiveness and efficiency creates unsatisfied customers and owners of such software systems [4].

2.2 Security

Security is also another important aspect to be evaluated and enhanced continually [17]. For instance, NLP approach is used in voice biometric authentication and access control management applications [5]. The voice representation techniques that are used in voice authentication systems must maintain confidentiality to prevent access to the representation techniques, which could lead to the penetration of authentication systems. Voice authentication systems can also be used against the important security goal of integrity by allowing unauthorized alterations to representation techniques that lead to a misleading or improper representation process. Human-Computer Interaction software systems must guarantee the availability of any application for legitimate users at any time according to the availability requirements specifications. For example, when biometrics systems are used for authentication, it must be available during all access permission requests to resources. Once the biometric authentication is not available, no access control will be activated.

Based on the above justifications on how important are both usability and security to any software system, Sects. 3 and 4 propose an evaluation methodology and a matrix to help in the evaluation of software systems.

3 Usability-Security Measuring and Evaluation

In order to develop a useful evaluation methodology, developers should have appropriate measuring techniques before the evaluation process begins. Access to suitable metrics gives developers the ability to predict the evaluation's outcome. However, usable-security is difficult to measure, consequently, it is difficult to evaluate as well [7,11]. Therefore, this work proposes new measuring and evaluation techniques based on OWASP [20] and SAULTA [10]. The evaluation technique serves as extended research work for the Assessment Framework for Usable-Security (AFUS) [13]. The following subsections describe methods for measuring and evaluating usability and security. In order to provide evaluation consistency, all the attributes and their properties are evaluated according to Table 2 after processing their measurements according to Table 1.

3.1 Usability Evaluation

The term usability is defined by the International Standard Organization (ISO) as the range in which a product can be operated by legitimate users to satisfactorily perform specific tasks in an effective, efficient, and specified way [26]. Some others add accuracy, memorability, and learnability as secondary usability factors [16]. This paper focuses only on the three factors of the ISO definition.

In order to evaluate the targeted usability level on any developed software system during the evaluation phase, the application state (level) of each of the usability properties (efficiency, effectiveness, and satisfaction) must be focused on, as well as providing a method to determine a measurable way to evaluate these properties on software systems. The following evaluation equations are adapted from Bevan and Macleod's concepts [6,18].

Table 1. Usability and Security Attributes and their Properties' Measuring Guidance, where the listed measuring statuses: HA, MA, SA, and NA represent High Achieved, Mostly Achieved, Some Achieved, and Not Achieved; respectively. α and β represent Usability and Security, respectively.

	Measuring Status				
	HA	MA	Achieved	SA	NA
EF1	$EF1 \geq 0.9$	$0.7 \leq EF1 < 0.9$	$0.6 \leq EF1 < 0.7$	$0.3 \leq EF1 < 0.6$	$EF1 < 0.3$
EF2	$EF2 < 1.0$	$EF2 = 1.0$	$1.0 < EF2 \leq 1.2$	$1.2 < EF2 < 1.5$	$EF2 \geq 1.5$
SA1	$SA1 \geq 0.9$	$0.7 \leq SA1 < 0.9$	$0.6 \leq SA1 < 0.7$	$0.3 \leq SA1 < 0.6$	$SA1 < 0.3$
α	$\alpha \geq 0.9$	$0.7 \leq \alpha < 0.9$	$0.6 \leq \alpha < 0.7$	$0.3 \leq \alpha < 0.6$	$\alpha < 0.3$
CO1	$CO1 \geq 0.99$	$0.95 \leq CO1 < 0.99$	$0.90 \leq CO1 < 0.95$	$0.80 \leq CO1 < 0.90$	$CO1 < 0.80$
IN1	$IN1 \geq 0.99$	$0.95 \leq IN1 < 0.99$	$0.90 \leq IN1 < 0.95$	$0.80 \leq IN1 < 0.90$	$IN1 < 0.80$
AV1	$AV1 \geq 0.99$	$0.95 \leq AV1 < 0.99$	$0.90 \leq AV1 < 0.95$	$0.80 \leq AV1 < 0.90$	$AV1 < 0.80$
β	$\beta \geq 0.9$	$0.7 \leq \beta < 0.9$	$0.6 \leq \beta < 0.7$	$0.3 \leq \beta < 0.6$	$\beta < 0.3$

Table 2. Usability and Security Attributes and their Properties' Evaluation Guidance, where the listed measuring statuses: HA, MA, SA, and NA represent High Achieved, Mostly Achieved, Some Achieved, and Not Achieved; respectively.

	Evaluation				
Measuring Status	HA	MA	Achieved	SA	NA
Evaluation Value	9	7	6	3	1

Effectiveness. Systems can only be considered as effective if their users are able to achieve their goal of operating such systems. The effectiveness property can be measured based on a goal-centered view by counting the number of successful tasks that legitimate users perform [12]. For example, a software systems is effective if it allows users to successfully create their passwords, login using their previously created passwords, or provide their biometric traits. Equation 1 can be used to evaluate the effectiveness for software systems, where n represents the total number of accepted tasks that legitimate users perform, R represents the result of each performed task's trial (either "failure" or "success"), $EF1$ represents the system effectiveness rate.

$$EF1 = \frac{1}{n} \sum_{i=1}^{n} \delta(R[i]) \tag{1}$$

$$Where \qquad \delta(\theta) \text{ is defined as: } \delta(\theta) = \begin{cases} 0 & \text{if } \theta = \text{failure} \\ 1 & \text{if } \theta = \text{success} \end{cases}$$

According to Table 1, if the result of system effectiveness rate, $EF1 \geq 0.9$, this means that the software system is highly effective. On the other hand, if the result of $EF1 < 0.3$, this means that the system is ineffective and may need further enhancement. After measuring the effectiveness, it is evaluated according to Table 2, because this provides the consistency among all of the usability properties.

Efficiency. Efficient systems must complete a specific task or process to reach a particular goal within an acceptable amount of time. The efficiency property is important because both the vendors and the users will not rely on a system that takes too long a time to perform a specific task (for instance: authentication). The measurement used to evaluate the efficiency is the amount of time that is consumed for achieving a particular goal or to complete a particular task. Equation 2 depicts the evaluation, where n represents number of trails to perform a particular task, β represents the standard average amount of time to finish such task, and T represents the amount of time to perform the task on each trial.

$$EF2 = \frac{\frac{1}{n}\sum_{i=1}^{n} T[i]}{\beta} \tag{2}$$

According to Table 1, if the result of system efficiency rate, *EF2*, is less than 1, this means that the software system is highly efficient. In contrast, if the resultant value of *EF2* is greater than or equal to 1.50, this means that the software system is inefficient because the average amount of time consumed to perform that task is too far beneath the standard average, β. After measuring the efficiency, it is evaluated according to Table 2, because this provides the consistency among all of the usability properties.

Satisfaction. For a system to be satisfactory, both the vendors and the users must be happy with the system. This is determined by the willingness of both vendors and users to rely on and reuse the system. It is important to be aware that the satisfaction is compellingly affected by the vendors' and the users' mood [16]. It is most likely to be perceived as satisfactory if it is both simultaneously effective and efficient. Evaluating the satisfaction property is a challenging task due to the difficulty of having accurate measurement tools. However, the best way to evaluate satisfaction is via questionnaires such as in SUMI [18] and the SUS [23] surveys, or interviews [16]. Therefore, satisfaction is evaluated through involving HCI-SEC principles that are related to user-centered approach [27] that focus on the ease of use, the degree of happiness, and the degree of confusion. Based on the results of the given survey, user satisfaction, *SA1*, is measured according to Table 1. After measuring user satisfaction based on the above degree of satisfaction via surveys, it is evaluated according to Table 2, because this provides the consistency among all of the usability properties.

Evaluating the above standard usability properties leads to an overall usability evaluation through summing the evaluations of the three properties (effectiveness, efficiency, and user satisfaction) as in Eq. 3, where α represents the usability:

$$\alpha = \frac{EF1 + EF2 + SA1}{3} \tag{3}$$

Equation 3 provides the α measure shown in Table 1. Consequently, same as evaluating the usability properties, overall usability of software systems is evaluated based on Table 2, because this provides the evaluation consistency among three quality attributes under investigation (usability, security, and usable-security).

3.2 Security Evaluation

The term "security" is identified in many ways. Essentially, system security is a set of methods and techniques used to prevent weaknesses from being exploited by applying three security goals: confidentiality, integrity, and availability [21]. Software systems have vulnerabilities that need to be discovered and closed, or at least protected from being imposed. Therefore, to reach an acceptable level of security, the three security properties must be applied and achieved. The following analyzes the importance of the security properties, how they are applied on the systems, and how they are measurably evaluated.

Confidentiality. Confidentiality is a goal of all secure systems. Confidentiality is defined as the ability to grant access only to authorized users. If unauthorized users gain access to computer systems, confidential information may be accessed and then used against the systems' vendors or users. To measure software systems' confidentiality, Eqs. 4, 5 and 6 are applied where n represents the total number of access trials, α represents an individual access trial (either "false access" or "true access"), TAR represents true access rate, FAR represents false access rate, and $CO1$ represents system confidentiality evaluation value.

$$CO1 = TAR \tag{4}$$

$$TAR = \frac{\sum_{i=1}^{n} \delta(\alpha[i])}{n} \tag{5}$$

$$Where \qquad \delta(\theta) \text{ is defined as: } \delta(\theta) = \begin{cases} 1 & \text{if } \theta = \text{true access} \\ 0 & \text{if } \theta = \text{false access} \end{cases}$$

and

$$FAR = 1 - TAR \tag{6}$$

According to Table 1, if the result of system confidentiality rate, $CO1$, is greater than or equal to 0.99, this means that the system is highly confidential. In contrast, if the resulted value of $CO1$ is less than or equal to 0.80, this means that the system failed to provide confidentiality because access was granted for too many unauthorized users. After measuring confidentiality, it is evaluated according to Table 2, because this provides the consistency among all of the security properties.

Integrity. Integrity means that for the authorized users, the system does not allow them to perform tasks in an improper way, and protects the data from any unauthorized alteration. As having usable-security evaluation for software systems is the goal of this paper, the integrity property must be correctly applied to make such systems secure. This property is achieved by enabling systems to create auto-backup and auto-check using proper techniques and tools like hashing, the process of comparing backup files with the same files on the system. Equation 7 depicts the integrity calculation on software systems, where n represents the total number of selected files for hashing, $IN1$ represents the integrity evaluation result.

$$IN1 = \frac{1}{n}\sum_{i=1}^{n}\delta\left(\frac{systemfile[i]}{backupfile[i]}\right) \tag{7}$$

$$Where \qquad \delta(\theta) \text{ is defined as: } \delta(\theta) = \begin{cases} 1 & \text{if } \theta = 1 \\ 0 & \text{if } \theta \neq 1 \end{cases}$$

According to Table 1, if the result of system integrity, *IN1*, is greater than or equals to 0.99, this means that the system is provides integrity, because the result of the hashing (comparing be-tween the system and the backup files) indicates that there were not any unauthorized alterations of system files. In contrast, if the resulted value of *IN1* is less than 0.80, this means that the system did not provide integrity because a critical alternation occurred. After measuring integrity, it is evaluated according to Table 2, because this provides the consistency among all of the security properties.

Availability. Availability is a security factor where the system's services and contents, or data, must be available at any time an authorized user needs to access them. It is measured based upon the number of success services or data access requests a system receives. Equation 8 shows the availability calculation, where n represents the total number of access trials, α represents an individual access trial (either "available" or "unavailable"), and *AV1* represents availability rate.

$$AV1 = \frac{1}{n}\sum_{i=1}^{n}\delta(\alpha[i]) \tag{8}$$

$$Where \qquad \delta(\theta) \text{ is defined as: } \delta(\theta) = \begin{cases} 1 & \text{if } \theta \text{ is available} \\ 0 & \text{if } \theta \text{ is unavailable} \end{cases}$$

According to Table 1, if the result of system availability evaluation, *AV1*, is greater than or equal to 0.99, this means that the software system is highly available, because it is accessible to its authorized users whenever needed. In contrast, if the resulted value of *AV1* is less than 0.80, this means that the system was not available because more than 20 % of the requests were not serviced. After measuring availability, it is evaluated according to the unified numerical evaluation values of Table 2, because this provides the consistency among all of the security properties.

Evaluating the above standard security properties leads to an overall security evaluation through summing the evaluations of the three properties (confidentiality, integrity, an availability) divided by 3. Equation 9 illustrates the calculation, where β represents the security:

$$\beta = \frac{CO1 + IN1 + AV1}{3} \tag{9}$$

The outcome of the above calculation provides one of the measuring statuses of Table 1. Consequently, same as evaluating the security properties, overall security of software systems is evaluated based on Table 2, because this provides the evaluation consistency among three quality attributes under investigation.

4 Usable-Security Evaluation Matrix

Analysis and measurement studies of usability and security properties on systems produced detailed understanding of the nature and structure of systems' usable-security. Based on the previous section, a usable-security evaluation matrix can be constructed that can be used as a guidance to achieve the overall goal of this paper, which is evaluation and enhancement for software systems to be both more usable and secure enough (see Fig. 1). The matrix shown in Fig. 1 is a proposed method for evaluating usable-security of software systems. Systems' usable-security can be evaluated by using Eqs. 3 and 9 that evaluate the systems' usability (effectiveness, efficiency, and satisfactory), and security (confidentiality, integrity, and availability). The results are then used in Eq. 10, where α represents the overall usability evaluation, β represents overall security evaluation, γ represents the matrix score, which is the usable-security evaluation.

$$\gamma = (\alpha \times \beta) - |\alpha - \beta| \tag{10}$$

The highest measuring evaluation category in the matrix is when the system attains a score of at least 0.81 as displayed in Fig. 2. This means that the system has applied and achieved the highest level of all of the six usability and security properties. Such a system is considered as usable and secure not only because it achieved the usability and security goals, but because it bridged both usability and security together with careful consideration to the HCI/user-centered approach from the begining of the development process.

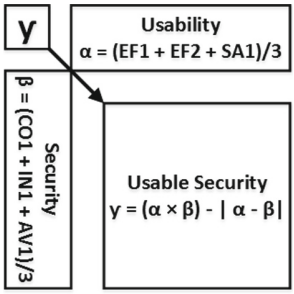

Fig. 1. The Usable-Security Evaluation Matrix: The evaluation process consists of two components: formative mathematical-based modeling that evaluates the security and usability properties, and a summative matrix that evaluates the Usable-Security based on the results of the formative modeling.

On the other hand, the lowest measuring evaluation category in the matrix is when a software system scores at most 0.01 as shown in Fig. 2, which means that the system has not achieved any beneficial values of the usability and security attributes. Such a system is considered neither usable nor secure not only because it does not achieve the usability and security goals, but because it does not bridge

γ	Highly Usable	Mostly Usable	Usable	Some Usable	Not Usable
Highly Secure	0.81	0.43	0.24	-0.33	-0.71
Mostly Secure	0.43	0.49	0.32	-0.19	-0.53
Secure	0.24	0.32	0.36	-0.12	-0.44
Some Secure	-0.33	-0.19	-0.12	0.09	-0.17
Not Secure	-0.71	-0.53	-0.44	-0.17	0.01

Fig. 2. Usability and Security Measuring Values: The measuring values and categories of the Security and usability attributes are used to evaluate Usable-Security

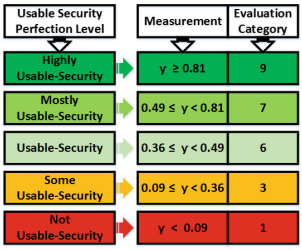

Fig. 3. Final usable-security evaluation and categorization guidance

usability and security as well. In addition, such a low score for a system indicates that its developers might have not considered the HCI/user-centered approach from the begining of the development process.

Walking through the whole evaluation process, a final usable-security evaluation is reached as one of five perfection levels: high usable-security when $\gamma \geq 0.81$, mostly usable-security when $0.49 \leq \gamma < 0.81$, usable-security when $0.36 \leq \gamma < 0.49$, some usable-security when $0.09 \leq \gamma < 0.36$, and not usable-security when $\gamma < 0.09$. Overall, a numerical categorization is given to each perfection level as follows: high usable-security is categorized as 9, mostly usable-security is categorized as 7, usable-security is categorized as 6, some usable-security is categorized as 3, and not usable-security is categorized as 1. Figure 3 presents the final usable-security evaluation and categorization guidance.

5 Case Study: Authentication Approaches

In order to show the advantage of using the usable-security evaluation matrix, this section presents the results of assessing the usability levels of an experiment that compares two authentication approaches. The Choice-Based Authentication Approach (CBAA) and the Traditional-Based Authentication Approach (TBAA) are described in detail in [14]). The experiment used 30 different scenarios for each approach and the results from the experiment are described below.

For the security values, we used one value (mostly secure: 0.7) for all the thirty scenarios of the TBAA, with expected entropy between 2^{26} and 2^{27}. As the assumed security value is considered as the security level that can be achieved when the NIST SP 800 Series is used as a foundation security policy [24]. However, security level of the CBAA has been increased because of the increased complexity for adversaries by displaying multiple authentication methods during the login process as explained in [14], where the expected entropy is between 2^{64} and 2^{65}, in addition to biometrics. Hence, we anticipated that the CBAA security should at least be better than the TBAA by 0.1. Therefore, we used a unified security value for all the thirty scenarios of the CBAA as (mostly secure: 0.8). Figure 4 displays the two sets of the scenarios.

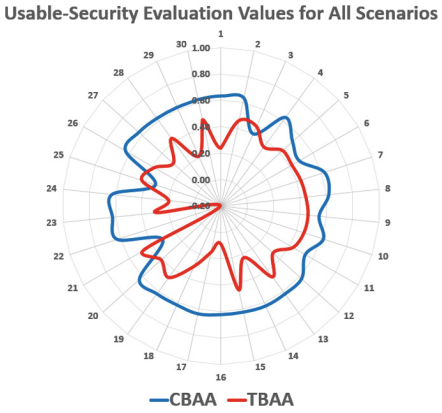

Fig. 4. Radar chart for the usable-security evaluation values. It displays usable-security coverage by the CBAA and the TBAA

Figure 4 displays a radar chart showing the trend-lines from evaluating usable-security levels among all the scenarios of both the CBAA and the TBAA. A comparison of the trend-lines for both approaches shows that the CBAA provided a better balance of usable-security than the TBAA. Moreover, it indicates that the usable-security evaluation results are more stable in the CBAA than the TBAA. In addition, the figure shows the usable-security coverage for both approaches and indicates that the CBAA occupies a wider domain than the TBAA.

Following the guidelines in Fig. 3, the results in Fig. 4 for the CBAA show that scenarios 21 and 25 achieved the "some usable-security" level (0.31 and 0.32, respectively), scenarios 3 and 6 provide the "usable-security" level of 0.40 and 0.48, respectively, while the rest of the 26 scenarios reached the "mostly usable-security" level (i.e.: the values are greater than or equal to 0.49, and less than 0.81). As a result, the overall usable-security evaluation value for the CBAA is 0.58, which is at the "mostly usable-security" level.

On the other hand, the TBAA scenarios displayed on Fig. 4 show that the scenarios 16 and 22 only achieve the "not usable-security" level (0.09 and −0.19, respectively). Moreover, 10 scenarios reached the "some usable-security" level, and only 18 scenarios provided the "usable-security" level. Consequently, the overall usable-security evaluation value for the TBAA is 0.35, which is the "some usable-security" level.

6 Conclusion

It is important that developers are able to evaluate usability, security, and usable-security software quality attributes that were specified within a system's requirements and were properly designed, correctly constructed, accurately evaluated, and appropriately deployed. However, existing methodologies for applying the above attributes do not assure the development of systems that can meet the necessary high quality standards for usability and security. The integration of such attributes becomes more important than just applying each attribute individually. The work presented in this paper goes beyond the application of attributes' integration, as it proposes an evaluation methodology to test the alignment of those attributes. Moreover, the proposed evaluation matrix is flexible enough to be used to modify the attributes' measures. It can also be extended to evaluate other quality attributes.

Acknowledgment. The authors would like to thank the Institute of Public Administration (IPA) in Saudi Arabia for their support of this work.

References

1. Adams, A., Sasse, M.A.: Users are not the enemy. Commun. ACM **42**(12), 40–46 (1999)
2. Alkussayer, A., Allen, W.H.: The ISDF framework: integrating security patterns and best practices. In: Park, J.H., Zhan, J., Lee, C., Wang, G., Kim, T., Yeo, S.-S. (eds.) ISA 2009. CCIS, vol. 36, pp. 17–28. Springer, Heidelberg (2009)
3. Alkussayer, A., Allen, W.H.: A scenario-based framework for the security evaluation of software architecture. In: 3rd IEEE International Conference on ICCSIT, vol. 5, pp. 687–695. IEEE (2010)
4. Atallah, M.J., McDonough, C.J., Raskin, V., Nirenburg, S.: Natural language processing for information assurance and security: an overview and implementations. In: Proceedings of the 2000 Workshop on New Security Paradigms, pp. 51–65. ACM (2001)
5. Benson, G., Re, S.R.: System and method for device registration and authentication, 8 June 2012, uS Patent App. 13/492,126
6. Bevan, N., Macleod, M.: Usability measurement in context. Behav. Inf. Tech. **13**(1–2), 132–145 (1994)
7. Cranor, L.F., Garfinkel, S.: Guest editors' introduction: secure or usable? IEEE Secur. Priv. **2**(5), 16–18 (2004)

8. DeWitt, A.J., Kuljis, J.: Is usable security an oxymoron? Interactions **13**(3), 41–44 (2006)
9. Ferre, X.: Integration of usability techniques into the software development process. In: International Conference on Software Engineering (Bridging the Gaps Between Software Engineering and Human-Computer Interaction), pp. 28–35 (2003)
10. Folmer, E., van Gurp, J., Bosch, J.: Scenario-based assessment of software architecture usability. In: ICSE Workshop on SE-HCI, Citeseer, pp. 61–68 (2003)
11. Garfinkel, S.: Design principles and patterns for computer systems that are simultaneously secure and usable. Ph.D. thesis, Massachusetts Institute of Technology (2005)
12. Hamilton, S., Chervany, N.L.: Evaluating information system effectiveness-part i: comparing evaluation approaches. MIS Q. **5**, 55–69 (1981)
13. Hausawi, Y.M., Allen, W.H.: An assessment framework for usable-security based on decision science. In: Tryfonas, T., Askoxylakis, I. (eds.) HAS 2014. LNCS, vol. 8533, pp. 33–44. Springer, Heidelberg (2014)
14. Hausawi, Y.M., Allen, W.H., Bahr, G.S.: Choice-based authentication: a usable-security approach. In: Stephanidis, C., Antona, M. (eds.) UAHCI 2014, Part I. LNCS, vol. 8513, pp. 114–124. Springer, Heidelberg (2014)
15. Hausawi, Y.M., Mayron, L.M.: Towards usable and secure natural language processing systems. In: Stephanidis, C. (ed.) HCII 2013, Part I. CCIS, vol. 373, pp. 109–113. Springer, Heidelberg (2013)
16. Kainda, R., Flechais, I., Roscoe, A.: Security and usability: analysis and evaluation. In: ARES 2010 International Conference on Availability, Reliability, and Security, pp. 275–282. IEEE (2010)
17. Kim, H.-C., Liu, D., Kim, H.-W.: Inherent usability problems in interactive voice response systems. In: Jacko, J.A. (ed.) Human-Computer Interaction, Part IV, HCII 2011. LNCS, vol. 6764, pp. 476–483. Springer, Heidelberg (2011)
18. Kirakowski, J., Corbett, M.: Sumi: the software usability measurement inventory. Br. J. Educ. Technol. **24**(3), 210–212 (1993)
19. Mayron, L.M., Hausawi, Y., Bahr, G.S.: Secure, usable biometric authentication systems. In: Stephanidis, C., Antona, M. (eds.) UAHCI 2013, Part I. LNCS, vol. 8009, pp. 195–204. Springer, Heidelberg (2013)
20. OWASP: risk rating methodology (2013)
21. Pfleeger, C.P., Pfleeger, S.L.: Security in Computing. Prentice Hall PTR, Upper Saddle river (2006)
22. Simpson, S.: Fundamental Practices for Secure Software Development: A Guide to the Most Effective Secure Development Practices in Use Today (2011)
23. Tullis, T., Albert, W.: Measuring the User Experience: Collecting, Analyzing, and Presenting Usability Metrics. Morgan Kaufmann, San Francisco (2013)
24. Weiß, S., Weissmann, O., Dressler, F.: A comprehensive and comparative metric for information security. In: Proceedings of IFIP International Conference on Telecommunication Systems, Modeling and Analysis (ICTSM 2005), pp. 1–10 (2005)
25. Whitten, A.: Making security usable. Ph.D. thesis, Princeton University (2004)
26. Good, M., Spine, T.M., Whiteside, J., George, P.: User-derived impact analysis as a tool for usability engineering. In: ACM SIGCHI Bulletin, vol. 17, pp. 241–246. ACM (1986)
27. Gutmann, P., Grigg, I.: Security usability. IEEE Secur. Priv. **3**(4), 56–58 (2005)