A Method for Developing Abuse Cases and Its Evaluation

Imano Williams[1], Xiaohong Yuan[1*], Jeffrey Todd McDonald[2], Mohd Anwar[1]

[1] Department of Computer Science, North Carolina A&T State University, 1601 East Market St., Greensboro, North Carolina, USA.
[2] Department of Computer Science, University of South Alabama, 3150 Jaguar Drive, Mobile, Alabama, USA.

* Corresponding author. Tel.: 1(336)2853700; email: xhyuan@ncat.edu
Manuscript submitted January 11, 2016; accepted March 11, 2016.
doi: 10.17706/jsw.11.5.520-527

Abstract: To develop secure software, software engineers need to have the mindset of attackers. Developing abuse cases can help software engineers to think more like attackers. This paper describes a method for developing abuse cases based on threat modeling, attack patterns, and Common Weakness Enumeration. The method also includes ranking the abuse cases according to their risks. This method intends to help non-experts create abuse cases following a specific process, and leveraging the knowledge bases of threat modeling, attack patterns, and Common Weakness Enumeration. The proposed method was evaluated through two evaluation studies conducted in two secure software engineering courses at two different universities. Evaluation studies show that the proposed method was easier to follow by non-experts in generating abuse cases than brainstorming, and could reduce the time needed for creating abuse cases. Other findings from the evaluation studies are also discussed in the paper.

Key words: Abuse cases, threat modeling, attack patterns, common weakness enumeration, secure software development.

1. Introduction

It has been widely recognized that to secure cyberspace, it is critical to develop secure software. To develop secure software that are robust to defend exploits and attacks, security needs to be built into the Software Development Life Cycle (SDLC) [1]-[3]. Software engineers need to have the mindset of attackers, and understand threats that might affect the product [4], [5]. One of the software security best practices that can help software developers to think like an attacker is to create abuse cases. An abuse case is a use case from an attacker's perspective with the intent to harm the system, an actor of the system, or a stakeholder [6], [7]. Creating abuse cases allows developers to elicit security requirements, decide and document how the software should react to illegitimate use, and develop countermeasures and mitigations of the abuse cases [8], [9].

It has been suggested that that abuse cases be created through informed brainstorming [10]. Such a method would require developers have high expertise and experience in security in order to produce meaningful and useful abuse cases. McGraw [8] suggested that abuse cases be developed based on a set of requirements and standard use cases, and a list of attack patterns. Our previous work proposed a specific process for developing abuse cases based on threat modeling and attack patterns [11]. Such a method intends to allow software developers who do not have high expertise and experience in security to be able to develop meaningful abuse cases. Though an example was used to illustrate the use of this method, the

effectiveness of this method was not evaluated. This paper describes a method for developing abuse cases that extends the proposed method in [11], and the evaluation of the extended method. This method extends the previous method in the following ways: (a) This method includes ranking the risks of the abuse cases; (2) It incorporates information from Common Weakness Enumeration (CWE) [12] for mitigating the risks of the abuse cases.

The rest of the paper is organized as follows. Section 2 describes the proposed method for developing abuse cases based on threat modeling and attack patterns. Section 3 discusses our evaluation methods and results. Section 4 concludes the paper.

2. The Method for Developing Abuse Cases

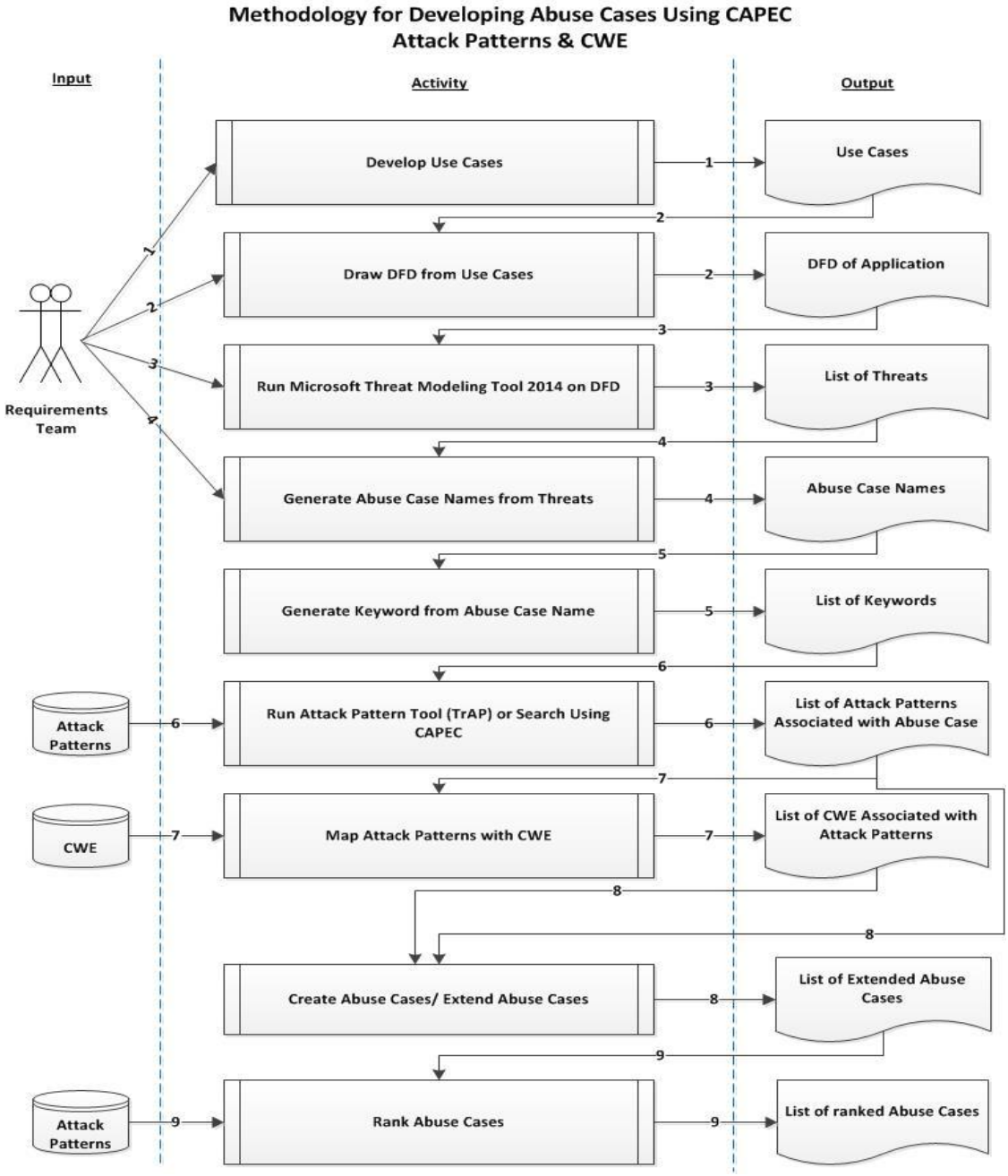

Fig. 1. The method for developing abuse cases based on Microsoft threat modeling, attack patterns, & CWE.

We propose a method for developing abuse cases based on Microsoft's threat modeling, attack patterns and Common Weakness Enumeration (CWE) as show in Fig. 1. [13].

The steps in Fig. 1. are explained below:

1) Develop use cases for the system based on system requirements/design documents or based on interacting with the system.
2) Draw Data Flow Diagrams (DFD) based on the use cases in Microsoft Threat Modelling Tool 2014 [14], which is a free tool developed by Microsoft to assist in threat modeling process [15], [16].
3) RUN Microsoft Threat Modeling Tool 2014 to identify potential threats for each element in the DFD, i.e., threats related to the entities, processes, data stores and interactions.
4) Identify abuse cases from the list of the threats identified in step (3). Use the threat names as abuse case names, or come up with abuse case names that best fit the threat and the use case.
5) For each of the abuse cases, generate a keyword(s) to search for attack patterns from Common Attack Patterns Enumeration and Classification (CAPEC) [17] that are related to the abuse case.
6) Use the Tool for Retrieving relevant Attack Patterns (TrAP)[18], [19] to search for CAPEC attack patterns relevant to the abuse using the keywords identified in step (5).
7) For each retrieved CAPEC attack pattern, go to the section of "Related Weakness" to find the related CWE-ID. Click on the CWE-ID link to read the description and the mitigation for the weakness.
8) Use the information from the attack patterns and CWE to fill out the abuse case information. The abuse case information is given in an Abuse Case Description Template which includes the following information: associated use case, abuse case name, attack pattern name and ID, keyword that was used to search for the attack pattern, abuse case objective, precondition/prerequisite, resources, attacker skills level, abusive attack flow, abusive post condition, system post condition, solution and mitigation, and the ranking of the abuse case.
9) Rank the abuse cases according to their risks.

In this method, the new elements that were introduced into the previous work [11] include steps 7 and 9.

In this method, the abuse cases generated are ranked according to their risks. The DREAD approach [15] proposed by Microsoft is used for ranking the abuse cases. Using DREAD, risk value can be calculated using the following formula:

DREAD Formula: Risk_DREAD = (DAMAGE + REPRODUCIBILITY + EXPLOITABILITY + AFFECTED USERS + DISCOVERABILITY) / 5

The variables in the formula takes values that are numeric scales from 0 to 10, where 0 indicate low, and 10 indicates high or very high. Information from the attack pattern relevant to the abuse case can be used to determine the risk rating using DREAD. For example, "Typical Severity" field of the attack pattern can be used to determine Damage Potential, "Typical Likelihood of Exploit" field of the attack pattern can be used to determine Reproducibility, "Attack Skills or Knowledge Required" field can be used to determine "Exploitability", and "Indicators-Warnings of Attack", "Probing Techniques" or "Attack Execution Flow Indicator" can be used to determine Discoverability. The value of Affected Users will be determined by the developer's understanding of the threats.

3. Evaluation of the Method for Developing Abuse Cases

Two evaluation studies were conducted on the proposed method for developing abuse cases. The first study was a pilot study conducted in a "Secure Software Engineering" (COMP727) class at North Carolina A&T State University in the Spring 2015 semester. Based on the results of the pilot study, the second evaluation study was designed and conducted in a "Secure Software Engineering" (CSC 440) class at University of South Alabama in the Fall 2015 semester. The goal of the evaluation studies is to find out how

effective is the proposed method for developing abuse case compared with brainstorming method for non-experts. The evaluation methods and results of the two evaluation studies are described below.

3.1. The First Evaluation Study of the Proposed Method

"Secure Software Engineering" (COMP727) is an online graduate level class. It teaches students how to incorporate security throughout the software development lifecycle (SDLC). The textbook for the course is "Software Security: Building Security In" [8]. There were 21 students enrolled in the class. Our hypotheses are: (1) the proposed method is easier to use by non-experts than brainstorming method; (2) the proposed method requires less time for creating abuse cases than brainstorming method.

The procedure of the first evaluation study is as follows:

1) Students were given Part A of a project which asked students to create abuse cases for a mock online shopping web application called Tunestore using brain storming approach, and then rank the abuse cases. No instructions were provided to the students on how to rank the abuse cases. The students have access to the mock online shopping web application and can interact with the application. Students worked on this project in groups of three. The students were required to work on the project individually first, and then discuss with his/her group members to create a final group document. The groups were required to log their discussion on the discussion forum on the online learning system, Blackboard.
2) After Part A of the project was submitted, students were given Part B of the project. Part B requires the students follow the proposed method (Fig. 1.) to develop abuse cases for Tunestore and rank the abuse cases. Students also worked in the same groups of three as in Part A. They were also required to work on the project individually first, and then discuss with his/her group members to create a final group document. In addition, they were asked questions such as issues they encountered with the steps in the proposed method, improvement that could be made to the proposed method, advantage and disadvantages of the proposed method compared with brainstorming, time spent on the assignment, etc.

Results. Students' responses to the questions in part B show that, more than 70% of the students thought the proposed method for creating abuse case was more straight forward (easy to follow) than brainstorming, gave more information, and produced more accurate results than brainstorming. Most students spent about half of the time in part B than in part A. These are consistent with our hypothesis that the proposed method is easier to use by non-experts than brainstorming, and that non-experts could produce abuse cases with less time than brainstorming. Issues encountered by the students include: (1) Some students had difficulties with generating the DFD for threat modeling; (2) Some students had difficulties with coming up with the keywords to be used for search for CAPEC attack patterns. Some students commented that the main effort needed was to understand the steps of the proposed methods. Once the method was understood, it was easy to follow the steps of the proposed method.

3.2. The Second Evaluation Study of the Proposed Method

3.2.1. Evaluation study overview

Evaluation study 2 was conducted in the "Secure Software Engineering" (CSC 440) class at University of South Alabama in the Fall 2015 semester. This course teaches students how to incorporate security throughout the software development lifecycle. The textbook for this course was "Software Security: Building Security In" [8]. There were 28 students enrolled in the course.

Based on our experience with the first evaluation study, we made the following changes to the design of the second evaluation study:

1) Instead of providing a mock web application to the students, two Software Requirement

Specifications (SRS) documents were provided to the students. This simulates the software requirements phase in which abuse cases need to be developed.

2) Instead of asking the students to develop the DFD for the software to be developed, the students were provided with a DFD. The intention was to eliminate the confounding variable of varied quality and levels of details of the DFDs generated by the students.
3) In the first study, both Part A and Part B asked students to generate abuse cases for the same system; students' work in Part A may very likely affect their work in Part B. To eliminate the order effects, in the second study, the participants were divided into two groups (Group 1 and Group 2) and were given two assignments. In Assignment one, students in Group 1 worked on Project 1 while students in Group 2 worked on Project 2. In Assignment two, students in Group 1 worked on Project 2, while students in Group 2 worked on Project 1.
4) Instead of asking students to work in groups, students were asked to work individually in the second evaluation study to find out each student's learning and effort in abuse case development.
5) Minor changes were made to the assignment description instructions to make it clearer to the students based on student feedback from the first evaluation study.

In addition to the hypotheses tested in the first evaluation study, the following hypotheses were added:

Hypothesis 1: Using the proposed method non-experts will be able to generate better quality abuse cases than brainstorming;

Hypothesis 2: Using the proposed method non-experts will rank abuse cases with better consistency than brainstorming;

Hypothesis 3: Using the proposed method the students will generate more consistent abuse cases in many different individual attempts than brainstorming;

Hypothesis 4: Using the proposed method non-experts will be able to generate more abuse cases compared with brainstorming.

The procedure of the second evaluation study is as follows:

1) The students were given Assignment 1, in which they were asked to develop abuse cases using brainstorming and rank the abuse cases. No instructions were given to the students on how to rank the abuse cases. Students were divided into two groups. Students in Group 1 were given the software requirements specification document (SRS) and DFD of an online shopping application. Students in Group 2 were given the SRS and DFD of an online banking application. Each student worked individually on the assignment.
2) The students were given Assignment 2 after Assignment 1 was submitted. Assignment 2 asked students to follow the proposed method to develop abuse cases and rank the abuse cases. Group 1 and Group 2 switched projects, i.e., Group 1 worked on the online banking application, and Group 2 worked on the online shopping application.

3.2.2. Evaluation results

Of the 28 students enrolled in the class, 26 submitted their assignments. From student survey responses, 79% of the students thought the proposed method was clear and easy to follow. This confirms the result from the first evaluation study. The average estimated time for developing the abuse cases in Assignment 1 was 2.13 hours and the average estimated time for developing the abuse cases in Assignment 2 was 1.65 hours. This indicates using the proposed method to develop abuse cases may have taken less time compared to the brainstorming method, which concurs with the observation from the first evaluation study.

The following discusses the results related to the additional hypotheses of the second evaluation study:

Hypothesis 1: Using the proposed method non-experts will be able to generate better quality abuse cases than brainstorming. We compared the total number of abuse cases created for each web application in both

Assignment 1 and Assignment 2. It was found that the average number of abuse cases was slightly higher in Assignment 2 than in Assignment 1. It was also found that students' Assignment 2 work included abuse cases that are harder to perform and involve more sophisticated attacks than those in Assignment 1. Hypothesis 1 was supported by these observations.

Hypothesis 2: Using the proposed method non-experts will rank abuse cases with better consistency than regular brainstorming. We calculated the standard deviation of rankings for similar abuse cases for each application in each assignment. The average standard deviation of the rankings of similar abuse cases is 2.11 for the online banking application in Assignment 1. The average standard deviation of the rankings of similar abuse cases is 2.12 for the online banking application in Assignment 2. The average standard deviation of the rankings of similar abuse cases is 1.21 for the online shopping application in Assignment 1. The average standard deviation of the rankings of similar abuse cases is 1.48 for the online shopping application in Assignment 2. Therefore Hypothesis 2 was not supported.

Hypothesis 3: Using the proposed method the students will generate more consistent abuse cases in many different individual attempts than brainstorming. It was observed that there were more similar abuse cases in Assignment for a given use case in Assignment 1 than in Assignment 2. Therefore this hypothesis was not supported.

Hypothesis 4: Using the proposed method non-experts will be able to generate more abuse cases compared with brainstorming. The number of abuse cases created in Assignment 2 is more than those created in Assignment 1. In Assignment 1, on average each student generates 5.75 abuse cases for the online shopping application, and 6 abuse cases for the online banking application. In Assignment 2, on average each student generates 6.44 abuse cases for the online shopping application and 6.5 abuse cases for the online banking application. Moreover, there were more types of abuse cases created in Assignment 2 than in Assignment 1. These observations supported Hypothesis 4.

4. Conclusion

This paper describes a method for developing abuse cases based on Microsoft's threat modeling, attack patterns, and CWE. This method leverages the knowledge base of Microsoft threat modeling, CAPEC attack patterns, and CWE with the goal of enabling software engineers, especially those without high expertise in computer security to develop meaningful and useful abuse cases, and develop secure software. Two evaluation studies were conducted on the proposed method in the context of two Secure Software Engineering courses at two different universities. Both evaluation studies show that the proposed method is easier to follow than brainstorming method, and could help software developers to improve efficiency in developing abuse cases. The results of the second evaluation study also show that using the proposed method non-experts could generate better quality abuse cases and more abuses cases compared with brainstorming. The proposed method was also able to help non-experts to think more like an attacker. However, two of the hypotheses – "Using the proposed method non-experts will rank abuse cases with better consistency than regular brainstorming" and "Using the proposed method the students will generate more consistent abuse cases in many different individual attempts than brainstorming" were not supported by the second evaluation study.

Some of the limitations that may have affected the evaluation results of the second evaluation study include:

1) The number of abuse cases found by the students was limited. The participants were instructed to find abuse cases for all the use cases. However, only some of the students followed the instruction. This might have affected the evaluation of hypotheses 2 and 3.
2) The assignments were given at a time when students were busy preparing for mid-term exam, which

might have limited the amount of time and efforts the students allocated for the assignments, which might have affected the results of student work.

Our future plan is to further evaluate the proposed method with software professionals to understand how effective the proposed method helps software professional to develop secure software in real world development environment. Future work will also include designing and implementing a tool that automates part of the process to make it easier for software engineers to use this method in their software development.

Acknowledgment

This work is partially supported by National Science Foundation under the grant DGE-1318695. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] Howard, M., & Lipner, S. (2006, June). T*he Security Development Lifecycle: SDL: A Process for Developing Demonstrably More Secure Software*. Microsoft Press.

[2] Build Security In: Setting a higher standard for software assurance, Retrieved February 20, 2015, from https://buildsecurityin.us-cert.gov/bsi/home.html

[3] CERT software assurance. Software engineering institute. Retrieved February 20, 2015 from http://www.cert.org/work/software_assurance.html

[4] Viega, J. & McGraw, G. (2002). *Building Secure Software: How to Avoid Security Problems the Right Way.* Boston: Addison-Wesley.

[5] Ardi, S., Byers, D., Meland, P. H., Tøndel, I. A., & Shahmehri, N. (2007). *How Can the Developer Benefit From Security Modeling?* presented at the Second International Conference on Availability, Reliability and Security.

[6] Alexander, I. (2003). Misuse cases: Use cases with hostile intent. *IEEE Software*, *20(1),* 58-66.

[7] McDermott, J. & Fox, C. (1999). Using abuse case models for security requirements analysis. *Proceedings of 15th Annual Computer Security Applications Conference* (pp.55-64).

[8] McGraw, G. (2006). *Software Security: Build Security In*, Addison-Wesley Professionals.

[9] Tndel, I. A., Jensen, J., & Rstad, L. (2010, February). Combining misuse cases with attack trees and security activity models. *Proceedings of the International Conference on Availability, Reliability, and Security (pp. 438-445).*

[10] Hope, P., McGraw, G., & Anton, A. I. (2004). Misuse and abuse cases: Getting past the positive. *Security and Privacy, 2(3)*, 90–92.

[11] Yuan, X, Borkor, E. N., & Yu, H. (2015). Developing abuse cases based on threat modeling and attack patterns. *Journal of Software*, *10(4)*.

[12] Common weakness enumeration. Retrieved, from https://cwe.mitre.org/.

[13] Williams, I. (2015). Evaluating a method to develop and rank abuse case based on threat modeling, attack patterns and common weakness enumeration. Thesis. North Carolina A&T State University.

[14] Microsoft threat modeling tool 2014. Retrieved on February 20, 2015, from http://www.microsoft.com/en-us/download/details.aspx?id=42518

[15] Chatper 3 threat modeling. Retrieved on February 20, 2015, from http://msdn.microsoft.com/en-us/library/ff648644.aspx

[16] Uncover security design flaws using the STRIDE approach. Retrieved on February 20, 2015, from http://msdn.microsoft.com/en-us/magazine/cc163519.aspx#S3

[17] MITRE. (2012). Common attack pattern enumeration and classification. Retrieved on February 20, 2015, from http://capec.mitre.org/index.html
[18] Yuan, X., Nuakoh, E. B., Beal, J. S. & Yu, H. (2014) Retrieving relevant CAPEC attack patterns for secure software development. *Proceedings of the 9th Cyber and Information Security Research Conference*.
[19] The tool for retrieving relevant attack patterns (TrAP). Retrieved from http://cstrap.ncat.edu/attackpatterns/logintotrap.html

Imano Williams was born in St. Catherine, Jamaica. Imano received his B.Sc. degree in computer science and electronics (double major) from the University of The West Indies, Jamaica in 2012. He then worked as a software quality assurance engineer before pursuing his MS degree in computer science at North Carolina Agricultural and Technical State University, Greensboro, North Carolina, United States of America in 2014. He received his MS in computer science in 2015. His current research interests include software security, usable security.

Xiaohong Yuan was born in China. She received her Ph.D in computer science from Florida Atlantic University, Boca Raton, Florida, USA in 2000. She is currently a professor in the Department of Computer Science, and the director of center for cyber defense at North Carolina Agricultural and Technical State University, Greensboro, North Carolina, USA. Her research interests include software security, health informatics security and privacy, mobile security, and information assurance education.

J. Todd McDonald is a professor of computer science in the School of Computing at the University of South Alabama and received his PhD in computer science from Florida State University, Tallahassee, FL, 2006. He received his master of science degree in computer engineering from the Air Force Institute of Technology in 2000 and his bachelor of science degree in computer science from the U. S. Air Force Academy in 1990. He served as an assistant professor and a chair of computer science and computer engineering in the Department of Electrical and Computer Engineering at the Air Force Institute of Technology from 2006 to 2010. In 2011, he joined the University of South Alabama. He has published over 30 papers and journals related to secure programming, software and hardware-based protection, cyber defense, and agent-based security. Dr. McDonald has shared in $1M of prior grant funding from the Air Force Office of Scientific Research and Air Force Research Laboratory to investigate software protection, cyber defense, and FPGA security and has received over $3M in funding from the National Science Foundation to develop cybersecurity education and research infrastructure. He retired from the U.S. Air Force as a lieutenant colonel after serving over 21 years as a communications-information/cyberspace operations officer specializing in cyber systems defense, research, and education. He is a senior member of the IEEE and member of ACM, Eta Kappa Nu, and Upsilon Pi Epsilon.

Mohd Anwar is an assistant professor of computer science at North Carolina A&T State University. He directs the Secure and Usable Social Media & Networks Lab. Dr. Anwar has experience and interest in designing usable secure, privacy-preserving, and trusted infrastructures, systems, and applications in different online contexts, specifically in Online Social Networks (OSN), e-Learning, and e-Health domains. His publications are in areas of access control, identity and trust modeling in cyberspace, privacy-enhancing technologies, human factors in security, cloud security, mHealth, and big data and smart health.