Crowd Sourcing the Creation of Personae Non Gratae for Requirements-Phase Threat Modeling

Nancy Mead[1], Forrest Shull[1], Janine Spears[2], Stefan Heibl[3], Sam Weber[4], and Jane Cleland-Huang[5]
Software Engineering Institute[1], Cleveland State University[2], DePaul University[3],
New York University[4], University of Notre Dame[5]
nrm@sei.cmu.edu, fjshull@sei.cmu.edu, j.l.spears@cshohio.edu, Stefan.Hiebl3@gmail.com,
samweber@nyu.edu, JaneClelandHuang@nd.edu

***Abstract*—Security threats should be identified in the early phases of a project so that design solutions can be explored and mitigating requirements specified. In this paper, we present a crowd-sourcing approach for creating Personae non Gratae (PnGs), which model attack goals and techniques of unwanted, potentially malicious users. We present a proof of concept study that takes a diverse collection of potentially redundant PnGs and merges them into a single set. Our approach combines machine learning techniques and visualization. It is illustrated and evaluated using a collection of PnGs collected from undergraduate students for a drone-based rescue scenario. Lessons learned from the proof of concept study are discussed and lay the foundations for future work.**

I. Introduction

Modern software systems are constantly exposed to attacks from adversaries which, if successful, could prevent a system from functioning as intended or could result in exposure of confidential information. Accounts of credit card theft and other types of security breaches concerning a broad range of cyber physical systems, transportation systems, self-driving cars, etc., appear almost daily in the news. Building any public-facing system clearly demands a systematic approach for analyzing security needs and documenting mitigating requirements. The challenge of security is that adversaries are intelligently trying to defeat the intent of system stakeholders—not necessarily maliciously. For example, many extremely serious security violations have been caused by hard-working employees who put highly sensitive data on USB drives in order to work from home. While security can be analyzed at the networking and code level to prevent buffer overflows, SQL injection attacks, and so on, there is value in creating a mindset of defensive thinking early in the requirements engineering process. Thinking defensively means that for every new requirement or feature, we need to think about how it could be abused or defeated by adversaries.

An illustrative example of requirements process failure is a rash of Automated Teller Machine (ATM) fraud incidents that occurred in the 1980s, as described by Ross Anderson [1]. ATM designers were aware of the necessity of preventing fraud and implemented security features accordingly. Unfortunately, Anderson argues, they failed to correctly anticipate their adversaries' abilities and means of attack. The system designers assumed that criminals would attempt to break the cryptography underlying the ATM system, and so diligently designed the system to withstand such attacks. However, non-cryptographic attacks did not receive such consideration. The result was a system design that provided little defense against some fairly straightforward adversary actions. For example, at one bank, tellers could change a customer's mailing address to their own, request a replacement ATM card and PIN number to be sent to the customer's new address, and then change the mailing address back, all without any activity being recorded in the bank's logs.

In most systems, security requirements, if they exist at all, tend to be rather repetitive, specifying relatively obvious features such as "Only authorized users shall access personal healthcare information." Occasionally requirements are included that stem from regulatory controls, such as "An audit log must be maintained of every access to the patient's healthcare information." As the ATM fraud example shows, merely specifying security features is insufficient—one needs to anticipate ways in which a system can be misused by adversaries. It is critically important to perform systematic, rigorous, and customized threat analyses [9]. Once the likely threats have been identified, the associated attack methods can be documented, for example, as misuse cases [15]. With this information in hand, mitigation strategies aimed at thwarting the attack can be defined and documented as use cases that lead to appropriate written security requirements. This type of approach is standard in many existing security requirements engineering processes, such as SQUARE [12]. As a consequence, early specification of security requirements positively impacts fundamental architectural decisions that enable security concerns to be addressed from the ground up, rather than added as late-in-the-day patches in an attempt to remediate security vulnerabilities.

This paper introduces the use of Personae non Gratae (PnGs) as a means to focus attention on early-phase threat modeling. Personas have long been used for User Interaction (UX) design as a means of integrating user goals and perspectives into the design process [4]. Personas

2332-6441/17 $31.00 © 2017 IEEE
DOI 10.1109/RE.2017.63

represent the intended archetypal users of the system. In contrast, PnGs represent archetypal users who interact with the system in unwanted ways, and as a result undermine the system's security. In a recent study [20], we compared the use of PnGs against two other threat modeling methods, STRIDE and Security Cards, and found that threat models built using PnGs exhibited a higher degree of consistency than other techniques. Nonetheless, no individual threat model included all identified threats. In this paper, we therefore explore the idea of crowdsourcing the task of threat identification. Our approach uses information retrieval techniques to analyze the identified threats, collate them, and provide the results to a human analyst to assist with construction of a unified threat model.

In the remainder of this paper, Section II provides an overview of existing threat modeling techniques. Section III describes PnGs and their contribution to threat modeling. Sections IV and V present study data collection and analysis methods. The paper ends with a discussion of the results to date, threats to validity, and conclusions.

II. Threat Modeling

There are many definitions of threat modeling. Based on our discussions with practicing threat modelers, we adopt the definition that "a threat modeling method (TMM) is an approach for creating an abstraction of a software system, aimed at identifying attackers' abilities and goals, and using that abstraction to generate and catalog possible threats that the system must mitigate." [19]

There are a number of threat modeling methods. Perhaps the most well-known and widely used TMM is *STRIDE*. It was invented in 1999 by Kohnfelder & Garg [10], implemented at Microsoft, and widely adopted. A typical STRIDE implementation includes modeling the system with Data Flow Diagrams (DFDs), mapping the DFDs to threat categories, determining the threats via threat trees, and documenting the threats and steps for their prevention.

Trike was developed in 2005 to improve on perceived deficiencies of STRIDE [18]. It is designed for security auditing from a risk management perspective and models threats from a defensive viewpoint (i.e., in contrast to an attacker's viewpoint).

The *Process for Attack Simulation and Threat Analysis (PASTA)* was developed around 2012 by Morana and UcedaVelez [23]; a companion book was published in 2015 [13]. PASTA is a seven-stage process that yields the impact of threats to an application and business.

Security Cards were developed at the University of Washington by Tamara Denning, Batya Friedman, and Tadayoshi Kohno [6], [5]. The stated purpose of Security Cards is to facilitate the exploration of potential security threats for a particular system, and more broadly, to help develop a security mindset. The audience envisioned for Security Cards includes educators, students, researchers, and practitioners. The use of Security Cards helps to answer the following questions: If your system is compromised, what human assets could be impacted? Who might attack your system and why? What resources might the adversary have? How might the adversary attack your system?

A *Misuse case* [21] is a TMM that extends the popular use case diagram to include security threats and countermeasures. A use case diagram, capturing core system functionality, is transformed into a misuse case diagram by adding potential mis-actors (i.e., threat agents), undesirable mis-use cases (i.e., threats) that could potentially disrupt or otherwise harm the desired system functionality, and use cases that counter identified threats. As such, misuse case diagrams depict the interactions between bad actors, threats, countermeasures, and the original use cases.

III. Personae Non Gratae (PnGs)

In this paper, we use PNGs to support threat modeling [3]. PnGs are inspired by the notion of *personas* in UX design and adopt several ideas inherent to Misuse cases.

A. Personas

A persona provides a realistic and engaging representation of a specific user group. It is typically depicted through a representative image and a personal description that portrays something about the pysche, background, emotions and attitudes, and personal traits of the fictitious person [16], [14]. The task of creating a persona usually involves surveying and interviewing users, identifying optimal ways for slicing users into categories, collecting data to demonstrate that the proposed slices create distinguishable user groups, discovering patterns within the user groups, constructing personas for each group, and then creating scenarios describing how the persona might interact with the system under development. A project will typically have about 5-8 personas.

While personas are typically used for purposes of UX design, there are examples in which they have been used as part of the requirements elicitation and design process. For example, within the privacy literature, Spears and Erete [22] defined a persona to elucidate a segment of end users who are unconcerned about online behavioral tracking. In another example, Dotan et al. evaluated the use of personas to communicate users' goals and preferences to project members as part of a two-day design workshop for the APOSDLE project [7]. Similarly, Robertson et al. also discussed the use of personas for gathering requirements when actual stakeholders are not available [17]. In these examples, the focus was on eliciting a general set of end user goals and system requirements. Cleland-Huang et al. introduced the notion of using personas to analyze architectural goals [2] and PnGs for security threats [3].

B. *Threat Modeling with PnGs*

PnGs represent archetypal users who behave in unwanted, possibly nefarious ways. However, like ordinary personas, PnGs have specific goals that they wish to achieve, and specific actions that they may take to achieve their goals. Modeling PnGs can therefore help us to think about the ways in which a system might be vulnerable to abuse and use this information to specify appropriate mitigating requirements.

The PnG approach makes threat modeling more tractable by asking users to focus on attackers, their motivations, and abilities. Once this step is completed, users are asked to brainstorm ideas about targets and likely attack mechanisms that the attackers would deploy. The theory behind this approach is that if engineers can understand what capabilities an attacker may have, and what types of mechanisms they may use to compromise a system, the engineers will gain a better understanding of targets or weaknesses within their own systems and the degree to which they can be compromised. Some critics of this approach argue that a PnG can often take users down the wrong path. For example, for a system related to national security, users might reason that the system may be the target of a sophisticated attack from another nation state. This conclusion, however, overlooks the fact that a nation state might compromise a system first through a much simpler entry point and then ratchet up operations from there.

We provide examples of two PnGs in Figure 1. These PnGs were constructed manually by two of the authors for training purposes and target the domain of Electro-Cardio Converters. Each PnG includes an image of the persona, his or her name, a description, the assumed role (e.g., Mechanical Engineer), and a moniker (e.g., Bitter and revengeful). Furthermore, it includes a set of relevant goals and skills, and a set of misuse cases which describe specific ways in which the PnG intends to attack the system [15]. From this, we can construct a threat model that includes the actor (i.e., the PnG) and the attack mechanism and target specified in the misuse cases. Attack intent is provided in the general description.

IV. Utilizing the "Crowd" to Create PnGs

To collect a set of PnGs for our crowd-sourcing study, we asked students in two introductory information security courses, one undergraduate and the other graduate, to work in teams of three to four people to construct PnGs for one of two systems [20]. In this paper, we focus on one of these systems which utilized UAVs to perform a rescue mission. The drones could carry payloads such as emergency supplies and were capable of autonomous operation if communication with the base station was lost. Each scenario was described in a two-page document that represented very early ideas for each of the projects. Goals, major constraints and high-level designs were provided, but not the lower-level implementation decisions.

Marvin
Mechanical Engineer
Bitter and revengeful

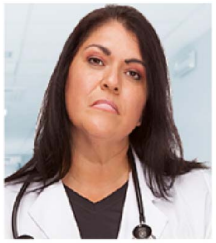
As a Mechanical Engineer, Marvin developed a new design for an implantable cardioverter-defibrillator (ICD) which he planned to patent. However, the MedsRUs company beat him to the punch and filed a patent for a similar design. They are now getting rich and Marvin is left feeling cheated and angry at his lost opportunity.

Recently divorced, and without the funds to support the life style he dreamed of, he has become increasingly bitter about his perceived loss.

Goals:
- To undermine the reputation of MedsRUs by disrupting the ICD behavior of random ICD users on the street.
- To accomplish the attack without detection.
- To cause discomfort to ICD users without killing them.

Skills:
- Strong coding/hacking skills
- Mechanical engineering/device building skills

Marvin's Misuse Cases which Threaten Correct Operation of the ICD	
1.	Snoop on the data transmitted along the serial cable between the ICDs reprogramming equipment and communication device in order to retrieve the patient's name, ID, and basic medical history which is all stored in the ICD.
2.	Transmit commands to replace the patient's personal information in the ICD.
3.	Transmit commands to shut off the device's ability to respond to cardiac events
4.	Transmit commands to switch to test mode so that a carefully-timed current triggers an arrhythmic test event which could stop the heart entirely.



Angela
Medical Doctor
Ambitious researcher

Angela is a medical researcher who believes that current practices in heart monitoring and defibrillation are not as effective as they might be. She has run some experiments on lab rats but wishes to experiment with human subjects. She knows that she needs to first go through the process of human-subject research approval, but before doing so, she has decided to try some informal experiments on her own group of patients. This way, she can short-circuit the otherwise laborious task of getting her study approved. She needs a break-through discovery to boost her career, which has currently stagnated.

Skills:
- Medical training to plan new therapy patterns.
- Lacks technical skills to tamper with the ICD command log in order to hide the unorthodox commands she sends to the device. However, she has persuaded her boyfriend, who is a skilled hacker, to write a program to purge entries from the log.

Goals:
- To perform a non-approved study on her patients by testing a variety of non-approved pulsing therapies.
- To cover her tracks by purging her illegal commands from the ICD logs.

Angela's Misuse Cases which Threaten Correct Operation of the ICD	
1.	Set heart rate cut-off rate and therapy initiation delays to experimental levels outside normal bounds.
2.	The ICD stores a record of any changes in setting, therefore Angela will tamper with the ICD command audit log on both the ICD and the external history database to remove all record of the current change in settings.
3.	During normal operation, the ICD stores a short-term log of ICD events. This log includes events such as ICD pulsing/therapy events, and sensor readings. Angela's boyfriend has written code to remove evidence of the illegal study by purging the historical log.

Fig. 1. Examples of two PNGs taken from the Electro-Cardio Converter Project.

For example, in our drone scenario it was stated that the drones would communicate directly with each other en-route, even if they were not able to communicate with the base station; however, communication protocols were not provided. 108 students engaged in the PnG activity. Of these students, 33 had no industrial IT experience, 29 had less than one year, 27 had between one and five

years, 11 had between five and ten years, and 8 had over ten years. Of those with industrial experience, the most common positions held were Programmer (15), Network Administrator (11), Security Analyst (10), Project Manager (10), Business Analyst (9), IT Support specialist (9), and Architect (9). As a result of the PnG creation process, the teams of students produced a total of 129 different PnGs for the drone system. Each PnG was structured using the template shown in Fig. 1. We refer to these students as the "crowd" from now on.

A. Analyzing PnG Quality

In our study, we compared the threat coverage for the set of PnGs created by each team against those created by four security experts, one from a major IT company and three working on government contracts. We found that individual teams often failed to identify all viable threats. From this, we hypothesized that the overall set of PnGs could be improved if individual results could be merged into a final set of PnGs. This paper describes the proof-of-concept approach we took for merging PnGs and then discusses lessons learned that will be applied in future crowd-sourced threat-modeling efforts.

B. PnG Merging Process

There are three primary goals in merging multiple PnGs: (i) to achieve broad coverage of threats; (ii) to minimize redundancy of threat descriptions; and (iii) to construct meaningful PnGs which provide useful tools for security experts, requirements engineers, and developers as they design, construct, test, and deploy a software system. Our approach balances automation with human analysis and decision making. We utilize machine learning and information techniques to collate information and to check for redundancies, and then present the collated data to an analyst to aid them in the process of PnG creation. Our approach is partially automated and involves the following steps.

Step 1: Discover Domain-Specific Concepts
The first goal is to discover the vocabulary used by the crowd to describe individual threats, attack mechanisms, and attack targets. Given a set of descriptions, such as the set of all PnG goals, we utilize natural language processing (NLP) to extract nouns and noun phrases (NNPs) [11]. Further, we compare the relative frequency at which an NNP occurs in PnG text versus its relative frequency in a large collection of generic documents covering topics such as science fiction, business, nature, and self-help. From this we compute *Domain Specificity (DS)* $DS(d,t)$ of NNP (term) t in PnG (document) d, as follows:

$$DS(t,d) = ln[\frac{freq(t,d)}{\sum_{u \in d} freq(u,d)} / \frac{freq(t,G)}{\sum_{v \in G} freq(v,G)}] \quad (1)$$

where the first element $(freq(t,d))/(\sum_{u \in d} freq(u,d))$ is the normalized number of occurrences of term t in the domain-specific document d, and the second element is the normalized number of occurrences of t in the general corpus of documents. Domain specificity (DS) is then calculated as the average value of all domain specificities from each document from the collection:

$$DS(t) = \frac{1}{|D|} \sum_{d \in D} DS(t,d) \quad (2)$$

The complete process is described in our prior work [11]. In Table I we report the descriptions of attackers mined from the *about* section of the individual PnGs. We manually organized them into five groups representing support services, drone operators, IT developers, terrorists and criminals, and external hackers. Other additional attackers were identified who did not fit into one of these main groups—for example, ex-manager and science graduate. While the categorizing task is currently performed manually, we plan to employ machine learning techniques to discover relationships between NNPs so as to automate it in the future [8]. Domain concepts associated with attackers are extracted from the about section of each PnG, while target-related concepts are extracted from the goal sections.

TABLE I
'Attackers' mined from the *about* section of the Crowd's PnGs and manually grouped into similar Actor classes.

Attack Grouping	**Mined Attacker Descriptions**
Support services	Account executive, Aviation Maintenance, Business owner, Company employee, Gear worker, Maintenance technician, Network administrator, Base controller , Security guard
Drone Operators	Drone company, Drone controller, Drone Operator, Drone pilot, Flight coordinator, Flight tester
IT Developers	Computer programmer, Developer Engineer, Drone software, Drone system, Software engineer, System Programmer, Systems analyst, Technology developer
Terrorists & Criminals	ISIS agent, ISIS recruit, Sovereign citizen, Theft ring, Fundamentalist group
Hackers	Wolf Hacker, System hacker

Step 2: Identify Attack Targets:
We observed that the students in our study tended to focus on potential targets first and then use them to anchor the PnG creation process. The PnG format did not specifically ask the users to specify threats. We therefore mined domain terms and phrases from the *goals* section of the PnG template to identify targets. A selection of these terms are shown in the column labeled 'Mined Target Descriptions' in Table II and manually placed into target groups. Based on domain frequency and specificity the top extracted targets were: (i) leader drone, (ii) drone technology, (iii) drone software, (iv) fly zone, (v) budget process, and (vi) drone user.

TABLE II
MANUALLY CLASSIFIED TARGET GROUPS AND SPECIFIC TARGETS MINED FROM THE PNG'S GOALS

Target Groups	**Mined Target Descriptions**
Drone Swarm	Drone Swarm, Leader Drone, Follower Drone
Drone	Drone Technology, Drone Software, Drone video, Drone camera, product quality, drone brain, power source
Drone controller	Decision maker, System operator, Drone pilot
GPS and Navigation	Fly zone, civilian traffic, traffic area, GPS coordinator
Communi-cations	Signal feed, data integrity, Drone signal
Organization	Budget process, Competitor's company, Company reputation, industry leader
Weapons*	Weapon system, Payload weapon
*The scenario did not include any reference to weapons. However we report it as documented by the users.	

Our current process presents targets to the user as a ranked list. We can observe that some represent real targets (e.g., leader drone), while others represent less tangible targets (e.g., fly zone). The analyst may select any of the candidate targets of interest.

Step 3: Visually Display Attack Mechanisms
Given a selected target, we parse all of the crowd's PnGs to identify and extract any threats associated with the target. From this we generate a spider web view showing specific threats, mechanisms, and actors as depicted in Figure 2. The threat descriptions displayed here are the raw text taken from individual PnG threats associated with the specified target—in this case, the leader drone.

The analyst manually reviews the threats and selects the most meaningful ones. In the future, we will provide tools that enable a user to modify or merge individual threats to improve their quality as well as reassign actors to threats. The iterative process of selecting targets and associated threats continues until the user is satisfied that all primary threats have been covered. While not yet implemented, in future work we plan to provide metrics based on the coverage of specific attacker groups and target groups.

Step 4: Merge Individual Threats into new PnGs
Once the analyst has completed the process of threat selection, we automatically merge all of the selected threats associated with an individual attacker into a separate PnG. For example, we would create one PnG based on the *Angry Activist* shown in Fig. 2. This PnG would contain the complete collection of threats associated with the activist across all spider web models.

Step 5: Check for redundancy
Finally, we compute the similarity between the generated

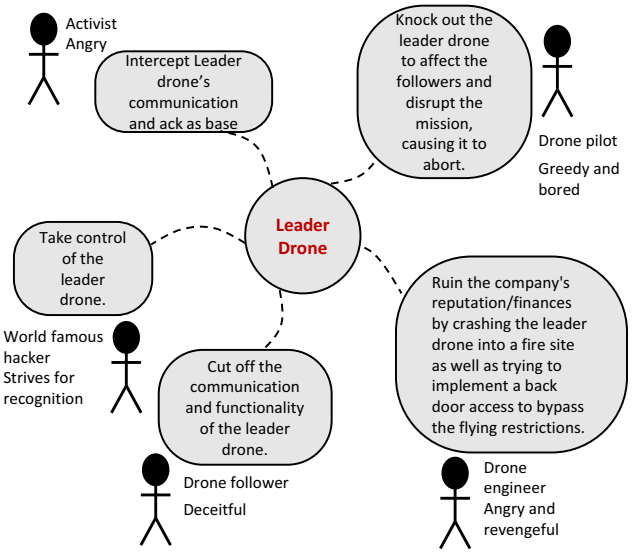


Fig. 2. Spider Web view of threats aimed at the Leader Drone. This information could be extracted and visualized automatically; however, this represents a manually constructed prototype.

PnGs to check for redundancy. In this proof of concept study, we have not yet implemented this step; however, we will use the Vector Space Model (VSM) to compute the similarity between threats assigned to different actors and generate a PnG similarity matrix. VSM is a standard approach described in most elementary information retrieval textbooks. It first transforms the text in two documents into a vector of terms, where each term is weighted according to its importance in the domain. Vector similarity is then computed by, for example, computing the cosine of the angle between the two vectors. By employing VSM or a similar approach, we can make the analyst aware of PnGs with similar goals and attack signatures so that they could consider merging them. We refer to the final set of merged PnGs as the *crowd-sourced* set.

V. DISCUSSION

In conducting the proof-of-concept study described in this paper, we demonstrated that machine learning techniques could be used to analyze individual PnGs created by a crowd. By combining data analysis and interactive visual displays, we could provide a meaningful approach for constructing a set of PnGs. These PnGs serve as input to the requirements process. For example, given a threat such as "intercept Leader drone's communication and acknowledge as base" the requirements engineer could identify knowledgeable stakeholders, bring them together in a brainstorming meeting, analayze how this threat could be technically realized and ultimately prevented, and finally specify appropriate mitigating requirements.

One challenge we faced in our proof of concept study was differentiating the roles of specific domain concepts. For example, given a threat specified in the PnG, we are easily

able to identify descriptions of attackers, targets, or attack mechanisms; however, we can not always differentiate between their roles. As such, the information we present to the user is not guaranteed to be correctly classified. In the future, we will collect more structured information during the initial PnG creation process. One appealing possibility is to use initial data collected using the Security Cards described in Section II as the inputs to the PnG creation process. Our prior study of three threat modeling methods [20] showed that teams using Security Cards identified a more diverse set of threats than those using PnGs. Further, the threats found by different Security Card teams were less consistent across teams. These observations suggest that considering threats produced by teams using Security Cards could ultimately lead to PnGs with better threat coverage.

VI. Threats to Validity

Our proof-of-concept study has several limitations, including the fact that we explored only one case study (drones) and that our crowd was composed of information systems students and not IT professionals. Furthermore, we have presented only a single illustrative example and have not evaluated it quantitatively. In future work, we will expand our scope to include a broader set of users, cover more than one scenario, and conduct a full user study. Despite its limitations, the study provides interesting insights that will allow us to modify our crowdsourcing technique prior to developing a complete tool and running a more extensive study.

VII. Conclusion

This paper is submitted to RE@Next! and serves as an initial proof-of-concept that we can utilize the crowd to develop PnG-based threat models. We have illustrated our approach in one project domain, but not yet fully evaluated it with users. In future work we plan to develop specific tooling to support all aspects of our process and to experiment in more diverse domains and projects.

VIII. Acknowledgments

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

[Distribution Statement A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution. DM-0004539SEI

We appreciate the help of Ole Villadsen at CMU for research on various threat modeling methods, and Nancy Ott at the SEI for editing the paper and assisting with the bibliographic entries.

References

[1] R. J. Anderson. Why Cryptosystems Fail. *Communications of the ACM*, 1994.

[2] J. Cleland-Huang. Meet elaine: A persona-driven approach to exploring architecturally significant requirements. *IEEE Software*, 30(4):18–21, 2013.

[3] J. Cleland-Huang. How Well Do You Know Your Personae Non Gratae? *IEEE Software*, 31(4):28–31, July 2014.

[4] A. Cooper. The inmates are running the asylum. In *Software-Ergonomie*, page 17, 1999.

[5] T. Denning, B. Friedman, and T. Kohno. Security cards: A security threat brainstorming toolkit. http://securitycards.cs.washington.edu/, 2013. [Online; accessed 23-February-2017].

[6] T. Denning, A. Lerner, A. Shostack, and T. Kohno. Control-alt-hack: the design and evaluation of a card game for computer security awareness and education. In *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*, pages 915–928, 2013.

[7] A. Dotan, N. A. M. Maiden, V. Lichtner, and L. Germanovich. Designing with only four people in mind? - a case study of using personas to redesign a work-integrated learning support system. In *INTERACT (2)*, pages 497–509, 2009.

[8] J. Guo, J. Cheng, and J. Cleland-Huang. Semantically enhanced software traceability using deep learning techniques. In *Proceedings of the 39th International Conference on Software Engineering ICSE 2017, Buenos Aires, Argentina*, 2017.

[9] T. B. Hilburn and N. R. Mead. Building security in: A road to competency. *IEEE Security & Privacy*, 11(5):89–92, 2013.

[10] L. Kohnfelder and P. Garg. The threats to our products. *Microsoft Interface*, April 1999.

[11] X. Lian, M. Rahimi, J. Cleland-Huang, L. Zhang, R. Ferrai, and M. Smith. Mining requirements knowledge from collections of domain documents. In *24th IEEE International Requirements Engineering Conference, RE 2016, Beijing, China, September 12-16, 2016*, pages 156–165, 2016.

[12] N. R. Mead and T. Stehney. Security quality requirements engineering (SQUARE) methodology. *ACM SIGSOFT Software Engineering Notes*, 30(4):1–7, 2005.

[13] M. M. Morana and T. UcedaVelez. *Risk Centric Threat Modeling: Process for Attack Simulation and Threat Analysis*. Wiley-Blackwell, Oxford, May 2015.

[14] L. Nielsen. *Personas - User Focused Design*, volume 15 of *Human-Computer Interaction Series*. Springer, 2013.

[15] A. L. Opdahl and G. Sindre. Experimental comparison of attack trees and misuse cases for security threat identification. *Information & Software Technology*, 51(5):916–932, 2009.

[16] C. Putnam, B. E. Kolko, and S. Wood. Communicating about users in ictd: leveraging hci personas. In *ICTD*, pages 338–349, 2012.

[17] S. Robertson and J. Robertson. *Mastering the Requirements Process*. Adison Wesley, 2006.

[18] P. Saitta, B. Larcom, and M. Eddington. Trike v.1 Methodology Document [Draft]. http://octotrike.org/papers/Trike_v1_Methodology_Document–draft.pdf, July 2005. [Online; accessed 22-February-2017].

[19] F. Shull. Evaluation of threat modeling methodologies. http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=474197, November 2016.

[20] F. Shull and N. Mead. Cyber threat modeling: An evaluation of three methods. https://insights.sei.cmu.edu/sei_blog/2016/11/cyber-threat-modeling-an-evaluation-of-three-methods.html, 2016.

[21] G. Sindre and A. Opdahl. Misuse cases for identifying system dependability threats. *Journal of Information Privacy and Security*, 4(2):3–22, 2008.

[22] J. Spears and S. Erete. "i have nothing to hide; thus nothing to fear": Defining a framework for examining the 'nothing to hide'persona. In *Symposium on Usable Privacy and Security (SOUPS)*, Menlo Park, CA, USA, 2014.

[23] T. UcedaVelez. Real world threat modeling using the pasta methodology. Technical report, Open Web Application Security Project (OWASP), 2012. [Online; accessed 23-February-2017].