

Security Requirements Engineering through Iterative Intrusion-Aware Design

Andrew P. Moore
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890

Fundamental to the Survivable Network Analysis (SNA) method, developed at the SEI [3], is the use of intrusion scenarios to improve the survivability of system designs. This position statement describes some relevant insights gained from applying SNA to several significant real-world systems. These insights help understand what is needed to use intrusion scenarios for security requirements engineering in a spiral-type, intrusion-aware development process. Specifically, I describe a scalable organization technique for intrusion scenarios that ameliorates some of the problems that we've encountered. I conclude with several key obstacles to using intrusion scenarios for security requirements engineering.

1. SNA INSIGHTS

The following describes requirements of the methods used to specify and analyze intrusion scenarios for survivable system analysis. These requirements arose as a result of SNA experiences and through detailed discussion with the SNA team members.

Methods must support documenting many diverse intrusions – The ever-increasing number of reported system vulnerabilities and exploits of these vulnerabilities argue for the need to be able to abstract, structure, and organize intrusion scenarios in a usable manner. Evidence indicates that often-ignored social engineering and physical attacks need to be taken at least as seriously as technological attacks [1]. In addition, attacks by individuals more sophisticated than the average recreational hacker, e.g., industrial spies and international cyber-terrorists, are becoming more likely and more difficult to counter.

Methods must help prioritize intrusion scenarios – Developers cannot afford to defend against all

possible attacks. Prioritizing intrusion scenarios according to their likelihood of occurrence and the impact to the enterprise mission is, therefore, critical. This requires understanding the likely adversaries of the enterprise in terms of their capabilities, resources, motivation, risk tolerance, and level of access. Only through this understanding can we derive requirements that provide the strongest defense and recovery of the enterprise mission at an affordable cost.

Methods must work in the face of changing intrusions – A vulnerability can progress through a number of states during its lifetime: birth, discovery, disclosure, correction, publicity, scripting, and death [2]. The state of a vulnerability impacts the importance, or even existence, of intrusions that rely on its exploit. Intrusion scenario documentation and analysis methods must be sensitive to the volatile nature of vulnerability discovery, exploit scripting, and system patching.

Methods must help improve design – The purpose of studying intrusion scenarios is to better understand how to defend against and recover from malicious attacks that could compromise an enterprise's mission. Such improved understanding should lead to improved design, and a more survivable mission operation. Without such an improved understanding, there is little benefit to studying intrusion scenarios.

2. A USEFUL ORGANIZATION

Attack trees are a useful way to organize intrusion scenarios. They have existed in various forms, and under various names, throughout the years, but their most recently published form describes them as a systematic method to characterize system security based on varying attacks [5]. Attack trees refine attacks on an enterprise by identifying the compromise

of mission-critical function as the root of the tree and refining the ways that an attacker can cause that compromise iteratively and incrementally as lower level nodes of the tree. Nodes may be decomposed either as a sequence of attack steps, represented as an AND-decomposition, or as alternative ways of executing the attack, represented as an OR-decomposition. An attack tree represents a set of intrusion scenarios, each of which accomplish the mission-critical compromise at the root node by different means.

The practicality of attack trees to characterize real-world systems depends on being able to reuse previously developed patterns of attack [4]. We organize such *attack patterns* into an encompassing *attack profile* with a common reference model. Different attack profiles may address different levels of attacker access, resources, and skills, as well as different configurations of system components. Therefore, different attack profiles may help refine an application-specific attack tree along different lines of attack.

3. CONCLUSIONS

Security requirements are best refined in parallel with the system operations and design in a spiral-type development process. Considering the possible avenues of attack during this process, i.e., making the process intrusion-aware, is critical to ensuring sufficient protection against and recovery from malicious attack. The methodological requirements described above, therefore, apply to methods for engineering security requirements.

Attack trees, and the use of attack patterns for reuse, help to achieve these requirements. They organize related intrusion scenarios in a compact way that relates back to the survivability of the enterprise mission. Attack trees allow the refinement of attacks to a level of detail chosen by the developer. The developer is free to explore certain attack paths in more depth than others while still being able to generate intrusion scenarios that make sense. Refining the leaves of the attack tree simply generates new leaves resulting in intrusion scenarios at the new lower level of abstraction. An attack pattern provides a structure to encode expert security knowledge for reuse. In addition, asking resistance, recognition, and

recovery questions at attack tree nodes may suggest improvements to both requirements and design.

Attack trees provide a powerful mechanism to document the multitude of diverse types of attacks, to abstract from intrusion details as a buffer against attack volatility, and to suggest improvements to requirements and design. They are, however, only a relatively small part of the answer as to how to use intrusion scenarios to improve security requirements engineering. The lack of accurate adversary models and risk analysis methods obstructs the progress on this front. A rich attack pattern library populated with attack patterns at the right level of abstraction is needed to build enterprise attack trees more systematically. Finally, the lack of robust resistance, recognition, and recovery countermeasures hampers our ability to improve designs and requirements. Overcoming these obstacles will require a truly interdisciplinary effort.

4. ACKNOWLEDGMENTS

My thanks to Robert Ellison, Richard Linger, John McHugh, Nancy Mead, and Tim Shimeall for sharing their insights on SNA experiences and their application to security requirements engineering.

5. REFERENCES

- [1] Anderson, R. Why cryptosystems fail. In Proceedings 1st Conf. On Computer and Communications Security, 1993.
- [2] Arbaugh, W.A., W.L. Fithen, and J. McHugh. Windows of vulnerability: a case study analysis. IEEE Computer, Vol. 33, No. 12, Dec. 2000.
- [3] Ellison R.J., R.C. Linger, T. Longstaff, and N.R. Mead. Survivable network system analysis: a case study. IEEE Software, Vol. 16, No. 4, July/August 1999.
- [4] Moore, A.P., R.J. Ellison, and R.C. Linger. Attack modeling for information security and survivability. To appear as Software Engineering Institute Technical Report CMU/SEI-2001-TN-001.
- [5] Schneier, B. *Secrets and Lies: Digital Security in a Networked World*. John Wiley & Sons, August 2000.