Information Assurance Applications in Software Engineering Projects

Lt. Col. Thomas A. Augustine (Ret.) and Dr. Lori L. DeLooze
United States Naval Academy

Four recent capstone projects by students in the U.S. Naval Academy's (USNA) Department of Computer Science offer some interesting insights into methodologies for information assurance (IA). This article looks at the tasks and challenges of each project and consolidates the experiences into lessons learned for designing and implementing software or systems that incorporate the IA concepts of confidentiality, data integrity, authentication, and system availability [1].

One USNA requirement (for computer science or IT undergraduates) is a capstone project. Students—in groups of three or four on a project of their choosing—must find a customer, define requirements, and meet key milestone dates in providing a software or system artifact. Projects require about 150 hours per person and must be completed and fully documented within the 15-week semester.

Over the past two years, there has been increased student motivation to choose IA-related projects. Like software or systems engineering projects in other fields, students found it especially challenging to define customer requirements and meet expectations and milestones. Faculty use these challenges as learning opportunities by allowing students to make their own project decisions, even if poor decision-making leads to a mid-project failure, because these failures will teach the students much more than a perfectly executed plan. Students found that taking on projects in the IA field of study created additional challenges in subject matter knowledge, system design, and implementation.

Team 1: Training Database with Multi-Level Views

The first project required students to develop a training and personnel database that provides proper authentication at an undetermined number of organizational or data visibility levels and minimizes repetition of data entry by using data normalization. This group found the requirements-gathering process to be fairly straightforward, as the customer understood the concept of a database with multiple levels of security. These requirements included allowing designated individuals to input and view multiple training courses as well as providing status reports to higher-level managers.

This team designed the software with an initial authentication scheme and then created a secure session that verified credentials before displaying data. *Read*, *write*, and *modify* rules were given based on database views, combining multiple tables in various views. The database administrator programmed these views, which had the capability of providing granular permissions. Originally, the team planned to hard-code permissions as *read* and *modify* for all levels higher than the supervisor, meeting the customer requirements. However, after initial design review, they realized that these requirements may later change; therefore, the team redesigned access control by giving the database administrator the ability to set visibility by level or by allowing the overriding of permissions. This additional flexibility added the capability to produce a report by giving full permissions by person, group, and supervisory levels, as well as highlighting all overridden permissions.

Team 2: Emergency Notification System

The next project[1] required an undetermined number of individuals and organizations to receive emergency notification of events based on input from city, state, nationwide, or worldwide sensors. Some events only need to be seen by the local emergency services while others required visibility for governors, the military, or federal officials. Some events are simply logged, while higher-priority events may require confirmation from the appropriate source and the ability to forward data to higher levels for additional action.

This team also had a challenge with the design and implementation of permission and authentication methods. Unlike the first group, however, these students did not initially incorporate authentication at the design phase. While they understood the requirements to have multiple levels of reporting based on the users' position and authentication, the team decided to put this off until the implementation phase. Because specifying the design is arguably the most difficult step in the software engineering process, many students simply want to get started with the implementation phase. These students started implementing code based on a poorly elaborated design. They split up work, each building separate Web pages for a type of emergency reporting required. This resulted in disparate pages that looked and operated differently and had no means of accepting dynamic changes. In addition, the team realized that they designed authentication by page or view rather than providing a consolidated, centralized means of authentication and a data permission schema.

Though reluctant to scrap six weeks of work, the team ultimately chose to begin from scratch and start again at the requirements phase. At this point, they developed formal interview questions for the customer and wrote a concise statement that encompassed all requirements. They then took each sentence or phrase and turned it into a well-defined requirement placed in a requirements implementation and testing matrix. From this matrix, the students created a design that incorporated every requirement. These requirements specified authentication and visibility schemas for each view. After further analysis, the team was able to design a method for centralized authentication and visibility with a small change to the database schema.

The team was able to re-accomplish requirements analysis and design in only a week, and was able to implement the backend database in another week. The team admitted that they had their doubts about whether they could finish the project on time, but were surprised to see the ease of implementing and testing well-defined requirements and design.

Team 3: Cybersecurity Competition Framework

This project—stemmed from a Polytechnic Institute of New York University competition—had students from numerous schools downloading various cybersecurity and digital forensics exercises that were timed and graded for accuracy and completeness.

USNA students felt that they could

improve the competition by designing a better interface for serving and grading the cyber challenges. As such, they set out to create a Web-based software framework that served various scenarios and received team responses for an IA competition. Requirements included authentication and proper visibility of scenarios for an undetermined number of teams, competition referees, scenarios, and team responses. Additionally, since they were creating a framework for a hacking competition, they had to design a system that maintained the integrity and availability of the data despite possible hijack attempts from less scrupulous teams.

In consultation with the faculty, students chose to both build the framework that was to serve the cyber challenges and create individual challenges as a proof of concept for their serving framework. As such, they assigned two team members to build the framework, while two others independently built challenges. The requirements called for a broad range of possible cyber challenges including: digital forensics of disk images, analysis of network traffic, analysis of software code vulnerabilities, and the identification and mitigation of operating system and applications security configurations.

Like the others, this team started by gaining detailed requirements for which protection against common software vulnerabilities was key. They quickly realized that system security had to be built into the design process. With this additional requirement, they realized the design would be the most difficult aspect of their project and allocated additional time for this milestone. Before designing in security, the team—both to protect their infrastructure and to develop challenges for the competition framework—had to understand how hackers use vulnerabilities to get into systems. Each student chose to specialize in specific network, operating system, application, and database security techniques. To better understand these techniques, students reviewed previous coursework, examined DoD and National Security Agency (NSA) security guides[2], and interviewed network security administrators. They found that the most difficult portion of securing an application against hackers is not the actual implementation of a specific configuration or fix, but in thinking like the hacker and predicting how people will use potential vulnerabilities to disrupt operations.

Though the team found implementation of the database and associated views to be relatively trivial, they found the documentation process to be challenging. Documentation had to include reasons for their design decisions and security settings, so that future maintainers could add features but still capitalize on the security features built into the framework.

Students noted that there was great value in understanding and implementing the security guides. While their IA course had many hands-on experiences, it was only through the course of this project that they realized the complexity of securing applications.

Team 4: Cyber Defense Exercise

In this project, students were required to design and implement a complete network based on an intricate set of constraints. After implementation, students had to operate and defend this network against NSA experts posing as attackers. Called the Cyber Defense Exercise [2], the competition is modified annually to increase the cybersecurity skills required of student participants. Several years ago, the focus was on active defense, while the more recent exercises have focused on the tradeoffs that need to be made between limited resources, operations of a network, security, time, and expertise required.

Students were provided with a 40-page directive spelling out the rules of the competition along with listing the network services that must be provided during a week-long exercise (and the points to be deducted if these services were either not operational or had security compromises). Essentially, students were given a very detailed requirements document with total freedom to produce any design. Though students were asked to turn in their designs, referees only verified that they met budgetary constraints. Cross-referencing requirements to design was a task left totally to each competing student group.

Though students were not required to gather requirements from an external customer, they did have to interpret the directive and design a complete network given the assumptions, constraints, and requirements. Students were challenged with creating a design that could provide users with a number of services, such as e-mail, chat, Web, databases, file servers, and mission-specific applications. This design had to remain operational while withstanding attacks from NSA network experts posing as hackers.

Student team members had taken both networking and IA courses, yet there was still a great deal of knowledge needed for the secure design of an operational network. Students augmented their knowledge of secure design with NSA security guides, Defense Information Systems Agency security checklists[3], and various security-specific books and references. Despite these numerous references, students were still challenged with consolidating this information and meeting the required constraints. Perhaps the students' greatest challenge was verifying the security of their design and implementation, which was done creating a test environment that mimicked the actions of the attackers. The team used security testing tools like the Metasploit Framework (which provides pre-packaged exploits) to test if the system is vulnerable to attack. Students also used other sites like <www.milw0rm.com> to test their system against additional potential exploits; however, the use of these more advanced techniques required great experience and training.

In addition to the testing of security, students had challenges in ensuring that the tightened security did not impact network operations. The students found this to be a great challenge. This balance between continued operations versus increased security involves business case and risk analysis, a skill that generally requires expertise in both network security and the mission area supported by the system.

Lessons Learned

Through these student projects, we can learn a number of security-related lessons about gaining requirements, as well as designing and implementing IA-focused systems or software.

Design Authentication and a Data Permissions Schema Early

Nearly all application or system development requires authentication methods. Students found that the best results were achieved by planning for both position-level and personal-level authentication for data visibility during the design phase. Even when requirements only call for simple authentication, customers tend to ask for a layered authentication by data type, organizational position, or data view. Rework tends to be extensive and time-consuming. Planning for authentication in the design phase of systems will likely save time and resources in the long run.

Use Security Guides

Students found that despite more than 100 classroom hours spent learning about networks and IA techniques, additional application-specific information is required when designing and developing IA-focused applications or systems. The NSA and the Defense Information Systems

Agency produce security guides for various operating systems and applications. These guides have been produced and tested by numerous experts and can complement the developers' knowledge to meet design specifications for applications requiring a cybersecurity focus.

Test Applications Against Known Security Frameworks

Relatively few software or system developers have the skills required to test system designs and implementations against well-known attacks using exploits in systems availability, data confidentiality, and integrity. Rather than recreating exploits that may require a greater effort than actually securing the system or application, system testers are encouraged to use existing security testing frameworks. While these testing frameworks can demonstrate potential holes in security, they should be used in concert with secure programming techniques, design, as well as documented and tested security techniques.

Plan for Regression Testing

Students working on these projects noted the need for an updated, descriptive test plan and follow-on regression software testing. This is true in any software application, but tends to be highlighted in security-focused applications. Security enhancements are rarely made in a single place. Instead, these changes are made in the operating system, database framework, application, and various configuration files. Students found that a single change in any one of these areas forced regression errors that were very difficult to detect without a well-formulated and implemented test plan. Students learned that this testing needs to be done as changes are made, or it becomes necessary to back out entire blocks of changes to find the root cause of bugs.

Manage Security Expectations

Customers generally understand those requirements and expected outcomes that are directly related to their subject matter expertise. Through these projects, students noted that customers expect an application to be secure, but do not understand the resource costs or operations tradeoffs required to make this a reality. Students noted the need to manage customer security expectations in the requirements phase and later in the design phase. Students believed that the best way to manage security requirements and associated customer expectations was to provide a security, operations, and resource matrix that cross-references security trade-offs.

Software Defense Application

Through the lessons learned by USNA students, this article is a refresher for defense software developers on why it is important to design authentication and data permissions early, follow security guides, use existing techniques for security testing, do regression testing as changes are made, ensure that customers understand security costs and tradeoffs, recognize the unintended impact of users on security, and thoroughly document all elements of security architecture.

Understand Security Impact on Operations

Even after managing customers' security expectations and implementing security (expected to exceed requirements), students found that users can have the greatest unintended impact on security. In testing these applications with actual users (as they would use them), students found that users will bypass security, in turn impairing operations or user-expected procedures. These user-caused workarounds can reduce security effectiveness and give the application owner a false sense of security. Students learned that for security controls to remain effective, designers must understand existing user processes and procedures—and then design security architectures around these or build them in a user-friendly alternative.

Document Reasons for Security Architecture

Like many software professionals, students found project documentation among the most challenging processes. As a learning tool, students were required to make changes to projects based on documentation that either they created or (in some cases) was created by other student teams. Though effective documentation is always challenging, students found this difficulty was magnified when trying to modify security architectures. Ultimately, they noted that understanding the security architecture documentation is not enough to effectively make changes to security without impacting operations or functionality. Instead, students found it easier to effectively manage security changes when they had documentation that also explained the reason for decisions, limitations in technology, the state of the intended operating system's security, and operational or process tradeoffs associated with security decisions.◆

References

1. Maconachy, W. Victor, et al. *A Model for Information Assurance: An Integrated Approach*. Proc. of the 2001 Workshop on Information Assurance and Security. West Point, N.Y. 5-6 June 2001.
2. Augustine, Thomas, and Ronald C. Dodge, Jr. *Cyber Defense Exercise: Meeting Learning Objectives thru Competition*. Proc. of the 10th Colloquium for Information Systems Security Education. Adelphi, MD: 61–67.

Notes

1. This project was inspired by the 2007 film "Live Free or Die Hard," where terrorists took control of emergency services, the electric grid, and city-wide traffic signals.
2. For access to these guides, visit: <www.nsa.gov/ia/guidance/security_configuration_guides>.
3. See <http://iase.disa.mil/stigs/checklist>.

About the Authors

Lt. Col. Thomas A. Augustine, D.Sc., is retired from the United States Air Force after a career in communications and information. His most recent assignment was as an assistant professor of computer science at the USNA, specializing in networks and IA.

E-mail: thomas.augustine@hotmail.com

Lori L. DeLooze, Ph.D., retired from the United States Navy as a career information professional. She is currently an assistant professor of computer science at the USNA, specializing in software engineering and IA.

572 Holloway RD, Stop 9F
Annapolis, MD 21402
Phone: (410) 293-6820
E-mail: delooze@usna.edu