

Proactive Methods for Secure Design

Privacy, Security and Cryptography

9 December 2009

Michael Lissner

Building a secure web site requires that organizations take a holistic approach to security. The security of the system must be planned in advance, built into the code itself, and then tested and maintained after the system is complete. Accomplishing each of these tasks can be challenging when other priorities are competing for the limited resources that an organization may have, however by investing in security at the outset, and then maintaining consistent and sufficient funding, a secure system should ensue.

The first step that an organization must take when beginning a project is to analyze the threat model that they anticipate facing. By analyzing the security objectives, application components, known threats, threat agents, and business impacts of an exploit, it is possible to determine the extent to which resources should be allocated to security. There are a number of models that can be used to analyze the threat model, many of which are listed in the OWASP Guide to Building Secure Web Applications.¹

When analyzing the quantity of investment that should be placed in security, it is important to think about incentives that can be created within the organization that will yield the desired results, and how best to create structures that achieve those

1 Adam Wiesmann et al., eds., "A Guide to Building Secure Web Applications and Web Services: 2.0 Blackhat Edition" (Open Web Application Security Project, July 25, 2005), 38.

incentives. One step that should be taken by organizations to create such incentives is to allow the security team to plan for the long term. This can be accomplished by various methods, but among them is creating a full-time security team and securing long-term funding for that team. This legitimizes security in the organization and allows the implementation of long term security planning. It also makes the security team stable and accountable, creating incentives for the team to take ownership over security concerns and to do its job well.

Another step that should be taken before the development of a system begins is to determine the organization's approach when working with government agencies. Depending on the purpose of the system, and the kinds of information that it may contain, this could be an important decision for the reputation of the system and the protection of its users. Some systems, such as those that allow users to make human rights complaints against their government, require that even the administrators of the system be unable to access some parts of the system. Thus, even if government officials request information from the system administrators, the organization can comply with the government by being unable to produce the information. While many organizations will want to reap the benefits of being able to access the data in the system and of working with the government, for those that do not, this may be an important decision that must be made at the outset.

The final step that should be taken before the organization begins creating their system is to formalize their disaster plan (as much as possible at such an early stage). This includes understanding what parts of the system may fail, how they will respond to those failures, and what kind of backup requirements they may have. While a proactive approach to security is the best defense, it is also vital to have a reactive approach as well, for those times that the defenses may fail.

With all of these elements and procedures in place, engineers can begin building the system itself. When building the system, there are a number of principles that should be followed. It is important that the engineers and designers of the system be familiar with these principles at the outset so that they can anticipate what kinds of problems they may encounter and what kinds of mistakes they may make. A short list of such principles is given in the OWASP Guide to Building Secure Web Applications, parts of which I explain here:²

- Defense in depth – have several layers of security for things that are of high importance. Thus, if one is breached, another may prevail.
- Least privilege – give only as much privilege to something as it needs to accomplish its task, but no more.
- Secure defaults – when choosing default values in your system, choose those which are the most secure, since they will likely never be changed.

² Ibid., 34.

- Minimized surface area – reduce attack vectors whenever possible by simplifying and narrowing your technology profile.
- Separation of duties – when possible, have multiple programs or people be required for those tasks that may be socially or economically tempting to a single person acting in isolation.
- Security through obscurity – Never rely on security through obscurity – assume that your secrets will be revealed sooner or later.
- Design to protect against false repudiation – Keep logs of all the actions that may be taken in the system, and plan on being able to track the actions of the system and its users.

The list itself goes on for many pages, and is useful reading for those interested. The point however, is that designers of systems should be trained in and should use security principles.

Another useful exercise when building systems is formalized code reviews. By having each coder be accountable to another person, and by having several people review each line of code, it is more likely that secure code will be the result.

In a similar vein, software solutions should be used that will analyze for common coding errors. Three such tools are immediately useful for software engineers. The first tool is prepackaged libraries that have been tried and tested in the real world by other organizations. These libraries are often quite easy to implement, save time, and

result in a more secure environment. They do however increase the likelihood that the engineers will not understand the system that they are using, and that misconfigurations or misuses will be possible. (As an example, consider the benefits and costs of creating your own web server vs. using an off-the-shelf version.)

The second tool that engineers should utilize when building a secure web application is a fuzzer. Fuzzers are programs that input data into programs with the goal of identifying security vulnerabilities. By choosing random or invalid data, fuzzers test programs in ways that may take humans an exceptionally long time. In addition, by building a large corpus of test data, fuzzers are able to systematically test input values and determine whether the program can be broken in that way.

The final software tool that web application engineers should use are security tests. These systems can vary in complexity, but generally the idea, like fuzzers, is to proactively scan for bad practices in code, and to alert the programmer before the code is placed into production. Like fuzzers, a large corpus of tests can be created, and many bad programming habits can be eliminated.

Two additional considerations to take when building the system are to reduce the quantity of sensitive data that the system contains, and to carefully encrypt what little sensitive data it must have. These practices are in line with the above principles of reducing the vulnerable surface area of your application and practicing defense in depth.

If all of the above techniques are carefully utilized by an organization, the resulting web application should be secure on the whole, however there are a few steps that should be taken after the application is complete.

The first step that should be completed is to create a plan for vulnerability testing. This plan could include regular tests by black hat teams, additional fuzz testing, or even a contest with prizes given to any person or team that can penetrate the security of the system. Each of these approaches has its own benefits and costs, but a thorough system of security testing will likely involve elements of each.

Finally, three steps that must be taken in order to maintain the original security of the system are to (1) subscribe to the announcement lists for the technology that is utilized by the application, to (2) follow the recommended patching and maintenance in those announcements, and to (3) revisit and revise the original threat model periodically. If all of these practices are done before, during and after building the application, is it likely that a secure application will ensue.