

The 5 Most Common Mistakes Made When Developing a Web Application

Johannes B. Ullrich, Ph.D., SANS Technology Institute, jullrich@sans.org

The Open Web Application Security Project (OWASP) maintains a very comprehensive list of the top ten most critical web application security flaws (Open Web Application Security Project, 2007). This list has been incorporated into industry standards like the Payment Card Industry (PCI) standard (PCI Standards Council, 2006). With this document, we will approach the problem from a different perspective to understand why these particular problems remain so common even though most web developers are aware of them and consider them in writing new applications.

Inconsistent Input Validation.

Web developers do have a general awareness of the need to validate user input. However, the implementation of user input validation is still lacking. When I recently installed a new network camera at home, I of course immediately examined the web based user interface for flaws. A first cursory check came back clean. However, digging deeper I found one interesting problem: The application allowed the user to specify the language to be used for the interface. The parameter specifying the language was included 3 times as part of each returned page. Each time, it was validated differently, leaving one instance open to cross site scripting attacks. Obviously, the developer for the application did not implement a centralized and consistent set of input validation functions.

This mistake is frequently the result of a “code as you go” approach. The developer didn’t take the time to plan and layout the application. A consistent set of input validation routines will not only lead to fewer mistakes. It will ease code review. In the end, it will reduce development time and improve code quality. This fact alone should make the implementation of consistent and centralized input validation a no-brainer. Hardly ever are you able to save time and money while improving security at the same time.

A barrier to this approach is a rushed project definition phase, and high developer turnover. Developers are asked to show results too early, leading to rushed “proof of concept” implementations which then turn into production applications after bolting on additional features. A more measured approach focuses on building the right infrastructure, like input validation libraries, first. Building input validation libraries does not provide the visible result developers are usually asked for from non-technical project participants. The inability of many companies to retain accomplished web development talent leads to orphaned code and many opportunities to standardize and reuse code are wasted as a result.

Some popular web application security text books focus too much on specific attacks. SQL Injection, Remote File Inclusion, and Cross Site Scripting are less a failure of validating particular input parameters. Most of the time, the developer understood the need to validate input, but the lack of a consistent framework made it too easy to miss a critical parameter.

Not Understanding the Technology

Web applications are changing rapidly and tools to develop web applications change as fast. Everybody involved in the web development process has to live up to the challenge to understand the security aspects of particular frameworks and development environments. This process is made harder if organizations try to chase the latest fad in web technology to just keep up with the industry. Over the last couple of years, we are seeing a substantial increase in the number and sophistication of SQL injection attacks. A common advice found in this context is the use of input validation and functions to escape malicious characters like single quotes. This advice is certainly important and valid. What about using database technology to your advantage? Stored procedures and prepared statements are so much better when it comes to avoiding SQL injection. Understand protections provided to you by the technology you are using. Another effective method to limit the damage of SQL injection is tightly controlled database privileges. Often developers are asked to configure database connection without providing them with the insight to understand how to do this correctly. The result is websites using the “sa” or “root” account. .Net offers a number of security enhancements like input validators, anti-csrf tokens and most. Sadly, many .Net developers do not know about these safeguards.

A missing understanding of the technology is in particular obvious when it comes to less common attacks like header response splitting. The web server and the HTTP protocol are frequently not sufficiently understood to realize the scope of these attacks.

Management on the other hand doesn't always understand the complexity of what they are asking developers to do. The result is a set of convoluted ever changing business rules which are not well explained and not thought true. Many times, even developers are surprised how easily HTTP data is manipulated and client site input validation is bypassed using the right browser plug-in or a proxy server.

Not Understanding the Business

Another problem that frequently arises from high turnover and sloppy specifications is a missing understanding of the business rules. Developers are too removed from the business, in particular if they do not work for the company whose website they are creating. Management has to do a better job in communicating the business rules to the developers and to help them understand the business. I was once involved in maintaining a web site for a clothing retailer. One of the items sold on the website was women shoes. To me, the website was yet another “run of the mill” e-commerce website. One of the anti-fraud features we typically added was a limit on how many items someone could purchase. To me, 5 pair of shoes was about as much as anybody would ever want to buy. This company was not too happy when we canceled the order of a famous actress who ordered a dozen shoes to wear on a movie set.

At developer will not be able to accurately model threats unless the developer is keenly aware of what the business objectives are and which critical information assets have to be protected by the application. Management has to be able to explain to the developer what the critical assets are and who should have access to them.

Underestimating the Threat

How would someone ever find my website? Why would someone be interested in it at all? You will hear this a lot from smaller websites. Some non e-commerce website will just not understand the threat. A few weeks ago, a local neighborhood website was compromised. The site had no assets of value. It did not

process credit cards and did not hold any confidential information. Instead, the attacker used the website to deliver malware to visitors. It is not known if this attack was successful in installing any malware. But the site was probably visited by a few dozen visitors while it was compromised and it is likely that at least one of them was infected with the password stealing bot distributed by the site. In these cases, the attacker is looking to borrow the trust users have into websites like this to increase the chances to infect clients. A regular visitor to a neighborhood website may not think twice to install a video codec if asked to do so by a popup. Trust is an important asset which is easily lost due to a compromise like this. On the other hand, the cost to the attacker was minimal. In this particular attack, many of the 100+ sites hosted on the same server had been compromised.

The same is true for larger websites, who just don't consider forced browsing a threat and believe that obscure URLs will protect administrator pages. The same websites think they can get away with lax access control and administrators sharing accounts and password.

Underestimating the User

In business, we do have a desire to assume that our customer or supplier is honest. Business transactions wouldn't be possible without giving others the benefit of the doubt. In particular customer service departments are usually trained to "make things happen" and routinely overwrite business rules to allow a customer to order a product or return a damaged item. This works well enough for physical transactions, but it can be catastrophic for online transactions. Weak password reset procedures are just one common result (Grossman, 2008). Web based applications should be designed with an "all users are evil" philosophy. Application design always has to consider the "what if" scenarios and second guess protection procedures put in place. Sometimes outside review is needed to better understand various scenarios.

The "insider threat" is a special case of underestimating the user. Calling our colleagues evil is nothing that comes easy. Insider actions are hard to restrict. For the Internet Storm Center, all handlers are able to add and edit diaries at will. This is an important feature that allows us to act and publish quickly. Of course, this is a very dangerous feature and a rogue handler, or a compromised handler account, could be used to do a lot of damage. In cases like this, the ability to log becomes very important. Logging will not prevent the attack. But logging will make it much easier to see what happened, prevent a repeat and cleanup the damage.

We do frequently underestimate the skills of our adversaries and the sophistication of the tools they have available. Hacking web application has become a hobby for many of them. One the most sophisticated web hacking tools, the Internet browser, comes with every modern system pre-installed. In some cases, you have several flavors to choose from right as you power up the system.

Conclusion

Among the issues listed above, you may find an overarching theme. You can call it the single most important issue in security: Know and understand what you are doing. As a developer myself, I caught myself writing a lot about what a developer has to do to improve security. I certainly contributed to the library of web vulnerabilities over the years. Sometimes, there is nobody else to blame but myself. But everybody involved in developing web applications has to understand essential web security principles.

Works Cited

Grossman, J. (2008, April 12). *Hacking Sprint Accounts Online Made Easy*. Retrieved May 5, 2008, from Jeremiah Grossman's Blog: <http://jeremiahgrossman.blogspot.com/2008/04/hacking-sprint-accounts-online-made.html>

Open Web Application Security Project. (2007, June 10). *OWASP Top 10 2007*. Retrieved May 1, 2008, from OWASP Web Site: http://www.owasp.org/images/e/e8/OWASP_Top_10_2007.pdf

PCI Standards Council. (2006, September). *PCI Standard Version 1.1*. Retrieved May 4, 2008, from PCI Standards Council: https://www.pcisecuritystandards.org/pdfs/pci_dss_v1-1.pdf