Input Validation

Input Validation is the primary task for every security specialist and software (web) developer. I will describe the problem using web development because it is my passion. Web programs get data from users using parameters. The parameters it is an entry point to your application. Hackers may impact on the parameters to impact on your application to intrude to your system. Parameters are the gate between your application and users. You have to take the gate protected and your application will be secure.

Input validation is not only security issue but it is the main one. You have to provide proper input data validation to make your system protected. There is no single solution to make the parameters secure. It depends on your application and how you use the data.

Assume all input is malicious. Do not trust to data received from users. Trust only to data received personally from you or your code that already checked the values. Other values must be checked.

All parameters must be checked by a centralized approach. It may be a class that validates every kind of parameters for all code in your application. The class must validate values using rules for every kind of parameters. When the one of the rule is changed you will need to change the validation class only. This approach will make your development and source code maintenance much easier. What kind of validation may be centralized? Take a look on the next list:

- numbers (integer or float) may be validated using regular expression;
- e-mail address may be checked using regular expression
- dictionary values – values that may be validated using dictionary (ZIP codes, country names and so on)

Some of the values may have unique validating rules. In this case you may implement the validation as much close as possible to the code where you use the values.

Where do we have to place the validation code? We can use client-side validation such as JavaScript and server-side. I like to use both methods. Client-side validation allows me to save network traffic and server resources because the code is executing on the user's computer. But NEVER trust to this kind of validation. Client code may be cut by hacker. I use JavaScript to make shallow validation. I check every value with profound validation when it comes to server.

How do viruses spread?

Every executable file has a header. This header contains the entry point: a program address from which it starts its execution. When a virus piggybacks on a program, it adds itself at the program's end and changes the entry point to itself, passing control to the old entry point only after the virus code executes. This way, when an infected executable file starts, the virus code executes first, after which control is passed to the program.

Some especially lazy virus writes do not like bothering with headers. They do it the other way around: They add the executable file to their virus, that is, the virus's body goes first.

This is the main manner of operation of most attachable viruses that were common until about 2000 – MS DOS viruses in particular. The least you can do to protect against this type of malicious code is to check the headers of executable files. A modified header is good cause for alarm, as this may have been done by a virus or a worm. Of course, keeping track of all file

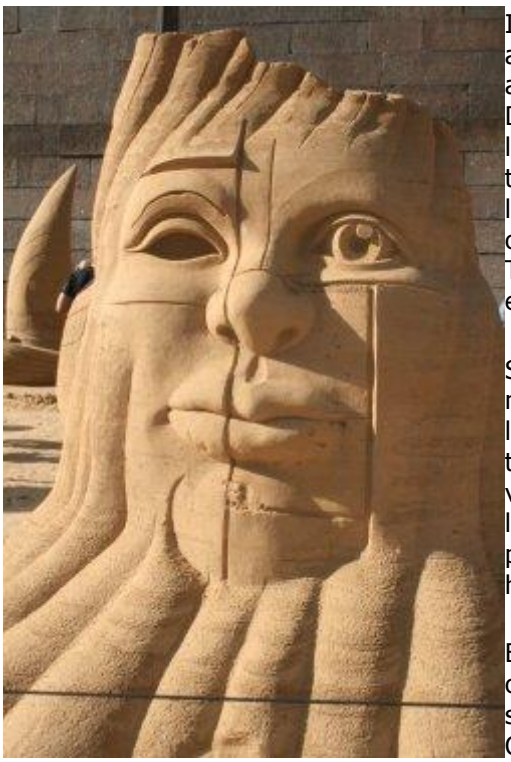
headers is a difficult task to carry out manually. At least the size of the main programs, however, can be checked, for it changes when a virus attaches itself to a file.

It is possible, though, for a virus to attach itself to an executable file without changing its header. In this case, the virus code can be contained in another program. The resulting effect is similar to the DLL execution: The infected program loads another program, which contains the executable code of the virus and executes this code.

As already mentioned, viruses of this type ruled until the end of the 1990s. In those days, the Internet was not as popular as it is now, and viruses spread mainly via diskettes or files downloaded from Bulletin Board Systems (BBS), a common means of information exchange at the time. Some viruses infect not programs, but boot sectors of disks or diskettes, and execute as soon as the computer boots. In this case, the virus loads into the memory and starts scanning for executable files, infecting all files of this type on the given computer.

Why infect all the files? The answer is very simple. In the MS DOS environment, there was only one means by which to start a program during the computer boot: listing the program needed in the autoexec.bat file. This file executes automatically when the computer boots and can launch other programs that are listed in it. But, if all viruses listed the programs they infected, they could easily be detected and neutralized by even the simplest of antivirus utilities. This is why a boot sector virus would try to infect all of the files on a computer. Launching any program would then also launch the virus.

In Windows, which has replaced DOS as the mainstream operating system for personal computers, viruses have more ways to camouflage themselves. They can also launch automatically when the computer boots, by placing themselves in the Windows' startup utility. So they no longer need to scan the file system in search of executable files to infect: The operating system will itself launch the malicious code every time it boots. But viruses of this type are no longer popular, because effective means of smoking them out of the startup area have been developed.



In addition to numerous ways of launching files automatically, Windows has many files that launch automatically when the system boots – the system Dynamic-Link Libraries (DLL), for example. This makes life easier for viruses. If in earlier times all programs had to be infected because it was not known which would be launched, and when, now it is enough to infect only one of the libraries or an important executable system file. This means no more tedious work scanning for executables.

So, if before the evil could hide only in executable files, now it has spread to dynamic libraries as well. Dynamic libraries have one big shortcoming: Code can be added to them that will execute when the library starts. Adding virus code to an automatically launched library will also launch this virus. Correspondingly, the number of potential loopholes by which the system can be accessed has increased greatly.

But dynamic libraries are not the only evils. To make lives of their customers easier, Microsoft has built Visual Basic support into many of its products. Almost all Microsoft Office programs make extensive use of the capabilities

provided by Visual Basic. Being able to expand the capabilities of your system and make your life easier is, undoubtedly, very convenient. For hackers, however, it is just another way to get into your computer.

Most Internet users had grown used to the idea that only executable files presented virus infection danger, while virtually nobody could imagine that a threat could be posed by text documents or electronic spreadsheets. This is why the first viruses embedded into Word documents or into e-mail processed by the Microsoft Outlook mail program infected a huge number of computers within a very short period of time. Neither users nor the best antiviral utility developers were ready for this turn of events.

I personally consider writing viruses one of the most stupid pursuits a person can engage in and, therefore, have never approved of it. This is the type of behavior exhibited by some children who, when they see that one of their playmates has a new toy, try to take it away or break it. This is exactly how hackers act in trying to deprive others of their toys or, what is even more threatening, their means for earning a living.

Today viruses spread mostly via the Internet, and diskettes or other removable media are practically never involved. E-mails in general, and mass mailings (read "spam") in particular, are the main vehicles. A typical scenario is something like the following. You receive a letter with an attachment and some intriguing message that leads you to open this attachment. But opening the attachment will launch the virus, which is what it actually is. What it does once it has been launched depends only on how imaginative or sick its creator was. One of its possible actions is to mail itself to all of the addresses in your address book.

Viruses like executable environment. They may look like executable scripts in the Microsoft Word program or any other program that may execute code. Some viruses spread using bugs in Internet Explorer. But that's another story for another one blog record.

Fundamentals of Hacker Attacks

There is no such thing as a universal method to break into an Internet site or server. Every time, an individual approach must be taken to open the necessary doorway. Although some attacks can overpower any defense, for example, a distributed denial-of-service (DoS) attack or brute-force password guessing, they can be too expensive in terms of time and computer resources to implement. Moreover, these attacks are as smart and subtle as driving a tank up to a bank and blasting the vault open. A hacker who breaks into a server using a brute-force attack to discover the password or who takes it out of commission by launching a distributed DoS attack against it will never be recognized as a professional; thus, these methods are used as a last resort and mostly by beginning crackers.

Why are attacks on computers increasing every year? The information about the security holes and vulnerabilities in computer systems used to be stored on bulletin board systems (BBSs), and only a few people with special privileges had access to it. So, it was hackers among these chosen few who carried out attacks with impunity, because their level of education and experience were quite high. It was difficult, often impossible, for a beginner or someone not belonging to the inner circle to gain access to such BBSs. This means that information about vulnerabilities and programs for implementing attacks were available only to a limited number of people.

Nowadays, this information and the necessary tools are available to anyone; thus, anyone can get into the cracking business. The situation is exacerbated by a host of utilities that automate the break-in process and are available to anyone at a number of Internet sites. With some of these utilities, all you have to do is to enter the address of the site to crack and click the Go button. The rest is done by the computer without any involvement on your part. You will not

know how the computer did this, but there are quite a few individuals who could not care less; the only thing they are interested in is the results.

For example: Using the CyD Network Utilities - Security tools software you don't have to know about vulnerabilities. You may search for vulnerabile WEB sites without knowing anything about SQL Injection. To do it you can use Automatic test for WEB site vulnerabilities module.

On one hand, information must be open to everyone. Thus, having information available on how their server can be compromised allows administrators organize proper defenses. That far from all administrators earn their pay by monitoring developments in the security field and applying them to make their networks more secure is another story.

On the other hand, shutting down sites that provide vulnerability information to any Tom, Dick, or Harry simply for asking would probably make the number of break-ins decrease significantly. Most break-ins are carried out not by professionals interested in the process but by wannabes for whom the results are of primary interest, making them quite content with point-and-click break-in techniques. Professional hackers seldom use publicly available break-in software.

I am torn between these two points of view and cannot make up my mind either for or against vulnerability information and break-in utilities being freely available to anyone. On one hand, information must be open so that administrators and programmers can use it to make their networks and software more secure. On the other hand, it should be restricted so that schoolkids are not tempted to earn a cool hacker reputation fast.

If pressed, I would probably cast my vote for the former but with a reservation that law-enforcement agencies exercise greater control about illegal use of such information and utilities. That's right; I am for regulation of the Internet within sensible limits. Quite a few aspects of social interaction require some sort of regulation. Why should such an important facet of the modern information society be allowed to meander unguided? By regulating, I don't mean that every click must be monitored; simply, everyone should be held responsible for their actions. We, as Internet users, must conduct ourselves in a civilized way; otherwise, just as cancer cells destroy their host body, we will destroy our common home.

Although it is interesting to observe two hacker teams breaking into each other's sites, a line must be drawn somewhere. Where this line should be drawn, I cannot say. The way things stand now, no one can decide how the Internet should be regulated because it's governed by different laws in different countries. But the Internet is a single body, and there must be a single law governing its use for everyone. Until such a law is passed and is enforced, the anarchy will continue.