

Lesson 9 Outline – Security Issues

Security is (or at least, should be) a central issue in CGI programming. There are many angles to explore, and in fact, this material could probably fill an entire course by itself! This lesson will just give a broad overview of the issues to consider, and I encourage you to continue to explore deeper on your own.

- I. Types of Security Problems – the term “security” is very broad, and can refer to any of these
 - A. Corruption of your data
 1. innocent errors on your part
 2. malicious intent from crackers (“hackers” is not the correct term)
 - B. Unauthorized access to the web server
 - C. Exposure of sensitive information (credit card numbers, passwords)
 1. Access to files on your server
 2. Access to data in transit between the customer and the server
 - D. Denial of Service attack – overloading your server
 1. Remember the eBay outage in 1999?
 2. Note that CGI.pm has a vulnerability to this attack, refer to the CGI.pm homepage for instructions to remedy this.

- II. Server Setup – first things first, the server should be configured properly
 - A. Appropriate use of firewalls, etc (the details are beyond the scope of this class)
 - B. Make sure all known patches are installed for your operating system
 - C. Know how to configure your web server software properly
 1. Use CGI wrappers as appropriate
 2. Limit the permissions of the web user
 - D. Shared servers (ISPs) have unique vulnerabilities, especially for world readable/writable files
 - E. Regular and frequent backups

- III. Programming Issues
 - A. CGI scripts are especially vulnerable since they can be used by virtually anyone anywhere – this means you will get errors that you never expected or tested for!
 - B. Since CGI scripts provide a direct connection to the web server from the outside world, great care must be taken in writing secure code
 - C. Never trust input data – always check for valid format
 1. Strip out “bad” characters that could be used for shell commands
 2. It’s easier and safer to test for a valid case rather than searching for all invalid characters
 - D. Remember that hidden fields can be corrupted, too.
 1. The cracker can download the form to his PC, edit the hidden variables and then call your CGI script using the modified form.
 2. One approach is to check the ‘HTTP_REFERER’ variable and make sure the CGI was called from a form on your server. This is OK, but not all browsers set this variable, and it is not mandatory per the HTTP specification.
 3. Don’t rely on hidden fields to set prices, etc. (many shopping carts do this!) Use a database instead.
 - E. Avoid directly passing variables to the shell (ex. using system() or opening pipes)
 1. Check for valid format first (see C. above)
 2. Use fork() and exec() to execute server commands without opening a shell (this limits the exposure to a cracker sending malicious shell commands). Fork creates a new process on the server, but does not open a shell. You can then use exec() to execute a command in the new process.
 - F. Use built-in security features, such as Perl’s taint feature, whenever (I don’t think you can run this on NT unless the web server is configured to run all perl scripts with the -t flag)
 - G. Don’t store passwords in plain text format.
 1. Remember that snoopers can sometimes view the contents of your setup file or even perl script itself.

2. The crypt() function provides a one-way only encryption (i.e. you cannot “decrypt”, you can only check if two crypt’ed strings match).

IV. Script Configuration

- A. Move all sensitive files (usually order logs, password files) outside of the web-accessible directory tree
- B. Be careful of saving information in plain text files, especially if they are world readable. Anyone with access to your server (ex other accounts on your ISP) can read these!
- C. Put an index.html file in every directory so that snoopers can’t browse for interesting files
- D. Don’t put writeable directories or files inside your cgi-bin directory – a cracker could upload a malicious script
- E. Only grant permissions that are needed – avoid world readable/writeable unless you need them
- F. Always change default passwords, etc – a number of sites were cracked in 1999 because the programmers didn’t change the default directory and password for Microsoft SQL server!
- G. Setup password protection to limit access to admin functions. .htaccess is the most common method for doing this.
- H. Installing pre-written CGI scripts has additional vulnerabilities
 1. Check that the script was written securely
 2. If possible, don’t use the default paths and file names – crackers who know how a certain script works can take advantage of any known weaknesses if you leave everything with the default settings
 3. Most of the time, CGI scripts are bundled in a single directory for ease in installation. Always move log files outside of the web-accessible directory

V. ECommerce Issues – really this is just a specific case of protecting sensitive data in transit between the browser and the server

- A. Use an SSL connection when transferring sensitive data
 1. It is only required that the ACTION part of the form be called using an https link, but most customers expect for the form to be served via https as well
 2. This protects the data from being intercepted during transmission between the browser and the server – this is really one of the least likely types of security attacks, although it’s subject to the most paranoia.
- B. Sending sensitive data via email is not secure, even if it comes from an SSL form. The only way to send secure email is to use encryption software, such as PGP.
- C. An alternative to sending information via email is to use another SSL web connection to view the order. In this case, you must be sure to have password protection on this admin function and also be careful about how the data is stored on the web server (see IV.)