Note from the Authors: This article is an excerpt from Chapter 19, Section 19.7 of *Perl How to Program, 1/e*. This article discusses Perl security and denial-of-service attacks. Readers should be familiar with Perl programming, regular expressions, basic CGI, file processing and have a basic understanding of HTTP requests and responses.The code examples included in this article show readers examples using the Deitel™ signature *LIVE-CODE™ Approach*, which presents all concepts in the context of complete working programs followed by the screen shots of the actual inputs and outputs.

Special Instructions for the code examples:

1. Program **fig19_08.pl** should be placed in your Web server's **cgi-bin** directory.
2. A **guestbook.log** file will need to be added to this directory as well.
This can be a blank text file.

19.7 Denial-of-Service Attacks

A *denial-of-service attack* occurs when a cracker attempts to use up system resources to deny normal users from being able to make use of a site or network. There are many forms of denial-of-service attacks; these vary by the resource being monopolized and the method being used to do so. Denial-of-service attacks can disrupt service on a Web site and even shut down critical systems such as telecommunications systems or flight-control centers.

Most denial-of-service attacks are not exploitations of poorly programmed pages. Instead, most attacks flood servers with data packets, overwhelming the servers and making it impossible for legitimate users to download information. These denial-of-service attacks usually require the power of a network of computers working simultaneously to generate enough packets to become a problem. When the packet flooding does not come from a single source, but from many separate computers, it is known as a *distributed denial-of-service attack*. In February 2000, distributed denial-of-service attacks shut down several high-traffic Web sites including Yahoo!, eBay, CNN Interactive and Amazon.

Such an attack is rarely a group effort. Instead, it is implemented by an individual who has installed viruses on various computers, designed to gain illegitimate use of the computers to carry out the attack. Distributed denial-of-service attacks can be difficult to stop, because it is not clear which requests on a network are from legitimate users and which are part of the attack. It is particularly difficult to catch the culprit of such an attack, because the attack is not carried out directly from the attacker's computer.

Another type of denial-of-service attack targets the *routing tables* of a network. Routing tables essentially form the road map of a network, providing directions for data to get from one computer to another. This type of attack is accomplished by modifying the routing tables, possibly sending all data to one address in the network.

Although not much can be done from a programming perspective to prevent most of these attacks, some attacks can be prevented. These attacks generally have to do with the size of the input being accepted.

For example, consider a guestbook example, where users input text into the form. What if a user posted a huge message with many megabytes of text? The CGI module would attempt to read all the text the user sent and assign it to a variable, potentially causing the server to run out of memory or to run slowly.

A remote user could also force your program to accept a large upload. This can occur even if your program is not explicitly programmed to accept uploads. The CGI module will accept the upload and store it in a temporary directory, perhaps filling all available hard-drive space. The file will be deleted when the program terminates, but the damage could have already been done.

Fortunately, the CGI module provides us with two ways to avoid this kind of a problem, through the use of special variables. The **`$CGI::POST_MAX`** variable can be used to limit the size of a **`post`** from a form. If the **`post`** is larger than the specified size in bytes, the program will quit and output an error message. The upper limit applies to uploads as well as ordinary posts. To disable uploads entirely, set **`$CGI::DISABLE_UPLOADS`** to **`1`**. This command prevents the program from accepting any file uploads. Other posting functions will still operate normally.

If you want to have the error message print back to users, so that users know what they did wrong, you can use the **CGI::Carp** module, as follows:

```
use CGI::Carp 'fatalsToBrowser';
```

Now, even though malicious users cannot use all your disk space in one large post, they can still do so through many individual posts if your program is writing them all to a file (as in a guestbook program). To prevent this problem, you can check the size of the file. If the file exceeds a certain size, you can trim it, or you can prevent the user from writing anything else to it. The guestbook program in Fig. 19.8 is rewritten with these concerns in mind.

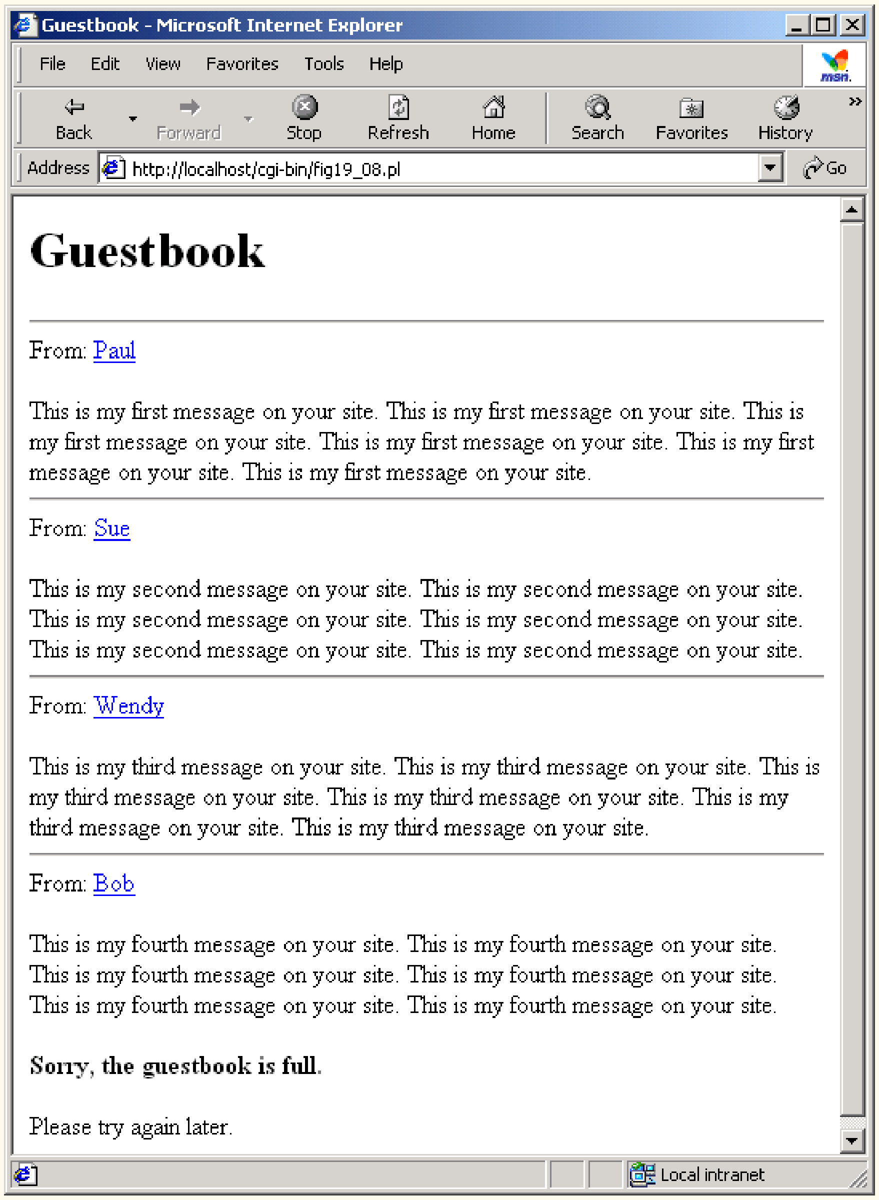
Line 11 sets the maximum posting size (**$CGI::POST_MAX**) to **512** bytes. Line 12 sets **$CGI::DISABLE_UPLOADS** to **1** (true) to prevent the user from uploading a file. We also have added a check on line 49 to ensure that the guestbook file is not too large before we output a form with which the user can sign the guestbook. Lines 61 and 62 print a message saying that the guestbook is full. The size limit on the guestbook is set at 1024 bytes for demonstration purposes.

```
1   #!/usr/bin/perl
2   # Fig. 19.8: fig19_08.pl
3   # Guestbook program that attempts to limit
4   # denial of service attacks.
5
6   use warnings;
7   use strict;
8   use CGI qw( :standard );
9   use CGI::Carp 'fatalsToBrowser';
10
11  $CGI::POST_MAX = 512;
12  $CGI::DISABLE_UPLOADS = 1;
13
14  print( header(), start_html( "Guestbook" ), h1( "Guestbook" ) );
15
16  if ( param() ) {
17     my $name = param( "name" );
18     my $email = param( "email" );
19     my $message = param( "message" );
20
21     if ( $name =~ /<.*>/ || $email =~ /<.*>/ ) {
22        print( h3( "HTML tags not allowed in Name or E-mail" ),
23           br(), "Please correct your entry and re-send", br(),
24           "To include the < or > symbols, use &ampgt or &amplt.",
25           br() );
26     }
```

Fig. 19.8 Avoiding denial-of-service attacks (part 1 of 3).

```
    else {
       print( h3( "Thank you for signing our guestbook!!!" ) );

       # filter to remove HTML tags
       print( $message =~
          s/<([^>]*)>/( $1 eq "BR" || $1 eq "br" ) ?
          "<$1>" : "\&lt$1\&gt"/ge );

       open( FILE, ">>guestbook.log" ) or
          die( "Cannot open guestbook" );
       print( FILE "\n", hr(), "From: ",
          a( { -href => "mailto:$email" }, $name ), br(), br(),
          $message );
       close( FILE );
    }
}

open( FILE, "guestbook.log" ) or die( "Cannot open guestbook" );

print while ( <FILE> );
close( FILE );

unless ( -s "guestbook.log" > 1024 ) {
   print( hr(), h4( "Please sign our guestbook:" ),
      start_form(), "Name: ", textfield( -name => "name" ),
      br(), "E-mail: ", textfield( -name => "email" ), br(),
      "Enter your message:", br(),
      textarea( -name => "message", -rows => 5, -columns => 50,
                -wrap => 1 ), br(),
      h4( "Warning: Filtering HTML tags except &ltbr&gt" ),
      br(), submit( -name => "Sign the Guestbook" ),
      end_form() );
}
else {
   print( h4( "Sorry, the guestbook is full." ),
      "Please try again later.\n" );
}

print( end_html() );
```

Fig. 19.8 Avoiding denial-of-service attacks (part 2 of 3).


Guestbook - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History
Address http://localhost/cgi-bin/fig19_08.pl
Go
Local intranet

Guestbook

From: Paul

This is my first message on your site. This is my first message on your site. This is my first message on your site. This is my first message on your site. This is my first message on your site. This is my first message on your site.

From: Sue

This is my second message on your site. This is my second message on your site. This is my second message on your site. This is my second message on your site. This is my second message on your site. This is my second message on your site.

From: Wendy

This is my third message on your site. This is my third message on your site. This is my third message on your site. This is my third message on your site. This is my third message on your site. This is my third message on your site.

From: Bob

This is my fourth message on your site. This is my fourth message on your site. This is my fourth message on your site. This is my fourth message on your site. This is my fourth message on your site. This is my fourth message on your site.

Sorry, the guestbook is full.

Please try again later.

Fig. 19.8 Avoiding denial-of-service attacks (part 3 of 3).