Paul Burgwardt

#0304700

COMM442 Final

Advanced WWW
Professor Horn

Question #3:

```perl
#!/usr/bin/perl
# Paul Burgwardt
# COMM442 - Professor Horn
# Final
# Question 3

# Glob All Images In /pics
@pictures = glob "pics/*";

# Filter And Resize
foreach( @pictures )
{
	# Skip Images With *-400*
	if ( $_ =~ m/-400\./ ) { next; }
	# Save Filename Information
	$_ =~ m/(.*)(\....)$/;
	$name = $1;
	$extension = $2;
	$extension = lc($extension);
	# Only Allow Images
	if( $extension !~ /\.jpg$|\.gif$|\.png$/ ) { next; }
	# Change Spaces To Underscores
	$name =~ s/\s+/_/;
	$newname = "$name-400" . $extension;
	# Resize It
	`convert -resize 400 "$_" $newname`;
}
```

Question #4:

```
#!/usr/bin/perl

# Paul Burgwardt
# COMM442 - Professor Horn
# Final
# Question 4

# Glob -400 Pictures In /pics
@pictures = glob "pics/*-400.gif pics/*-400.jpg pics/*-400.png";

# Counter For Hash DB
$index = 0;

# Open DBM Hash
dbmopen (%PICTURES, 'dow', 0664) || die("Can't open dow: $!\n");

# Filter And Resize
foreach( @pictures )
{
            # Strip -400 And Get File Information
           $_ =~ m/(.*)\/(.*)-400(\....)$/;
           $name = $2;
            # Restore Spaces
           $name =~ s/_/ /;
           # Get Extension
           $extension = $3;
           $extension = lc($extension);
           # Fill Database
           $PICTURES{$index} = $index . "^" . $name . "^" . $_;
           # Increment index Counter
           $index++;
}

# Close DB
dbmclose(%PICTURES);
```

Question #5:

```perl
#!/usr/bin/perl

# Paul Burgwardt
# COMM442 - Professor Horn
# Final
# Question 5

# Create Days Array
@days = qw(Sunday Monday Tuesday Wednesday Thursday Friday Saturday);

# Get Information From localtime()
($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst) = localtime();
dbmopen (%PICTURES, 'dow', 0664) || die("Can't open dow: $!\n");

# Get Name And Source
$picture = $PICTURES{$wday};
$picture =~ m/(.*)\^(.*)\^(.*)$/;
$name = $2;
$source = $3;

# HTML Header
print "Content-type: text/html\n\n";

# Print JSON
print "{ \"dow\":\"$days[$wday]\", \"name\":\"$name\", \"path\":\"$source\" }\n";
```

Question #6:

```
#!/usr/bin/perl

# Paul Burgwardt
# COMM442 - Professor Horn
# Final
# Question 6

use CGI qw(:standard);
$query = new CGI;

# Get dow Parameter
$day = $query->url_param("dow");

# Create Days Array
@days = qw(Sunday Monday Tuesday Wednesday Thursday Friday Saturday);

dbmopen (%PICTURES, 'dow', 0664) || die("Can't open dow: $!\n");

# Get Name And Source
$picture = $PICTURES{$day};
$picture =~ m/(.*)\^(.*)\^(.*)$/;
$name = $2;
$name =~ s/_+/\s/;
$source = $3;

# HTML Header
print "Content-type: text/html\n\n";

# Print JSON
print "{ \"dow\":\"$days[$day]\", \"name\":\"$name\", \"path\":\"$source\" }\n";
```

Question #7:

It is becoming increasingly easier nowadays for hackers to introduce malware onto remote systems, as well as posing real threats and possible damage to these systems. It can be as easy as allowing users to upload or edit files, even though you have pure intentions. However, there are some tactics that can severely limit the chances of an intrusion.

There are some specific filetypes that definitely should not be uploaded, in their native format. Certain filetypes like PHP should be prevented, or modified to have a different extension. Once an executable file like a PHP file gets uploaded into your system, it has the potential to actually be ran, and the capacity to be malicious. For example, if someone uploads an executable file that prints out your passwd file, that is a huge security flaw, and your system becomes vulnerable. When it comes to users, you should trust them, but not their input, so account for unexpected input.

There a couple easy filters someone could implement to prevent malicious code from being hosted on their system. First off, a simple filter using regex to discard any attempt at uploading a file that contains non-word characters: “$filename =~ s/\W//g;”. If you are developing an application that only works with images, you could strictly limit the uploads to only accept images. Otherwise, we should filter the prospective file to check if it’s a PHP or TXT file, and if it is, we don’t want them on our system: “$ext =~ s/php/txt/i;”. The idea here being that we grab the extension, and if it matches case-insensitive to PHP or TXT, we throw out an error and the file. If a genuine file passes all these tests, make sure it doesn’t overwrite your information. If you’re working with a small group of users, you could implement a simple password authentication in order to upload files. Some people might suggest using the HTTP_REFERER as a means of preventing unwanted people, however, it has become too easy to spoof the referrer. Another extremely important filter would be to prevent traversing directories: “$filename=~ s/\/|\.\.//g;”. You definitely don’t want a malicious file being placed in a more serious location on your system.

When dealing with uploading of files from people other than your machine, it is crucially important that they are unable to cause harm. Once your machine is exposed to the internet, who knows who could be interested in getting in. These simple filters and other security tips can keep your system more secure and in working-form.