# 原理：

The rise of [64-bit x86](https://en.wikipedia.org/wiki/X64" \o "X64) processors brought with it a change to the subroutine calling convention that required the first argument to a function to be passed in [registers](https://en.wikipedia.org/wiki/Processor_register" \o "Processor register) instead of on the stack. This meant that an attacker could no longer set up a library function call with desired arguments just by manipulating the call stack via a buffer overrun exploit. Shared library developers also began to remove or restrict library functions that performed functions particularly useful to an attacker, such as [system call](https://en.wikipedia.org/wiki/System_call" \o "System call) wrappers. As a result, return-into-library attacks became much more difficult to successfully mount.

The next evolution came in the form of an attack that used chunks of library functions, instead of entire functions themselves, to exploit buffer overrun vulnerabilities on machines with defenses against simpler attacks.This technique looks for functions that contain instruction sequences that pop values from the stack into registers. Careful selection of these code sequences allows an attacker to put suitable values into the proper registers to perform a function call under the new calling convention. The rest of the attack proceeds as a return-into-library attack.

# 分析过程：

x86中参数都是保存在栈上,但在x64中前六个参数依次保存在RDI, RSI, RDX, RCX, R8和 R9寄存器里，如果还有更多的参数的话才会保存在栈上。

利用漏洞程序ggtest将system()在内存中的地址打印出来或者调试获得

（system()在内存中的地址不会被随机化），这样就不需要考虑ASLR的问题了。

利用攻击执行system(“/bin/sh”)，其中system()所需的参数”/bin/sh”保存在rdi寄存器中。

利用ROPGadget在/lib/x86\_64-linux-gnu/libc.so.6中搜素

gadget：pop rdi; ret

命令： ROPgadget --binary /lib/x86\_64-linux-gnu/libc.so.6 --only "pop|ret" | grep rdi

从而获得gadget的地址。

构造payload：首先找到漏洞程序中返回地址rip距离buf的偏移(40)，

然后确定gadget的位置以及参数”/bin/sh”的位置，system()在内存中的地址由漏洞程序发送过来。

payload = "a"\*136 + p64(pop\_ret\_addr) + p64(binsh\_addr) + p64(system\_addr)

# 所用gadget：

ret1: pop rdi; ret

# 栈：

Payload在栈中的情况

|  |
| --- |
| a  .  地址递增  .  . |
| ret1 |
| The address of string ‘/bin/sh’ |
| The address of system function |