**When a problem comes along, you must zip it!**

**0 1 1 2 3 5**

**4 x 710**

**14 7**

**fffffh**

**1 4 5 2 3 6**

**Phase 3--SOLVED**

Lots of conditionals, solved by tracing back through program and seeing what values would allow for a smooth passage through without exploding bomb

Found out it was looking for the ascii value of x in one of the cases

Rbp-4 =input, 4 bits, most likely integer

Rbp-9 = input, 1 bit, most likely character

Rbp-8=input, 4 bits, most likely integer

0x0000000000401013 <+0>: push %rbp //create memory for rbp, 64 bit

0x0000000000401014 <+1>: mov %rsp,%rbp //move the value of rsp to rbp

0x0000000000401017 <+4>: sub $0x10,%rsp //subtract rsp-16, moves rsp down two address blocks

0x000000000040101b <+8>: lea -0x4(%rbp),%r8 //load address of r8 into rbp-4

0x000000000040101f <+12>: lea -0x9(%rbp),%rcx //load address of rcx into rbp-9

0x0000000000401023 <+16>: lea -0x8(%rbp),%rdx //load address of rdx into rbp-8

0x0000000000401027 <+20>: mov $0x402806,%esi //move mem address to esi, 32 bit

0x000000000040102c <+25>: mov $0x0,%eax //move 0 to eax

0x0000000000401031 <+30>: callq 0x400cb0 <\_\_isoc99\_sscanf@plt> //call function isoc… (see below)

0x0000000000401036 <+35>: cmp $0x2,%eax //compare eax value to 2

**0x0000000000401039 <+38>: jg 0x401040 <phase\_3+45> //jump if eax>2, go to line 45**

**\*\*\*\*need at least 2 inputted values for the function to continue\*\*\*\***

0x000000000040103b <+40>: callq 0x401755 <explode\_bomb> //if eax<2, explode bomb

**0x0000000000401040 <+45>: cmpl $0x7,-0x8(%rbp) //compare rbp-8 to 7**

0x0000000000401044 <+49>: ja 0x401136 <phase\_3+291> //if rbp-8 is above 7, jump to explode bomb

0x000000000040104a <+55>: mov -0x8(%rbp),%eax //move rbp-8 into eax

0x000000000040104d <+58>: jmpq \*0x402820(,%rax,8) //

0x0000000000401054 <+65>: mov $0x68,%eax //move 104 to eax

0x0000000000401059 <+70>: cmpl $0x385,-0x4(%rbp) //compare 901 to rbp-4

0x0000000000401060 <+77>: je 0x401140 <phase\_3+301> if equal, jump to later comparison on way out

0x0000000000401066 <+83>: callq 0x401755 <explode\_bomb> //calls explode bomb

0x000000000040106b <+88>: mov $0x68,%eax //move 104 to eax

0x0000000000401070 <+93>: jmpq 0x401140 <phase\_3+301> //jump to later comparison on way out

0x0000000000401075 <+98>: mov $0x7a,%eax //move 122 to eax

0x000000000040107a <+103>: cmpl $0x1ee,-0x4(%rbp) //compare 494 to rbp-4

0x0000000000401081 <+110>: je 0x401140 <phase\_3+301> //if equal jump to comparison on way out

0x0000000000401087 <+116>: callq 0x401755 <explode\_bomb> //calls explode bomb

0x000000000040108c <+121>: mov $0x7a,%eax //move 122 to eax

0x0000000000401091 <+126>: jmpq 0x401140 <phase\_3+301> //jump to later comparison

0x0000000000401096 <+131>: mov $0x6a,%eax //move 106 to eax

0x000000000040109b <+136>: cmpl $0x1fd,-0x4(%rbp) //compare 159 to rbp-4

0x00000000004010a2 <+143>: je 0x401140 <phase\_3+301> //if equal, jump to later comparison

0x00000000004010a8 <+149>: callq 0x401755 <explode\_bomb> //calls explode bomb

0x00000000004010ad <+154>: mov $0x6a,%eax //move 106 to eax

0x00000000004010b2 <+159>: jmpq 0x401140 <phase\_3+301> //jump to line 301

0x00000000004010b7 <+164>: mov $0x64,%eax //move 100 to eax

0x00000000004010bc <+169>: cmpl $0xdb,-0x4(%rbp) //compare 209 to rbp-4

0x00000000004010c3 <+176>: je 0x401140 <phase\_3+301> //if equal, move to line 301

0x00000000004010c5 <+178>: callq 0x401755 <explode\_bomb> //calls explode bomb

0x00000000004010ca <+183>: mov $0x64,%eax //move 100 to eax

0x00000000004010cf <+188>: jmp 0x401140 <phase\_3+301> //jump to line 301

0x00000000004010d1 <+190>: mov $0x78,%eax //ascii of x to eax

0x00000000004010d6 <+195>: cmpl $0x2c6,-0x4(%rbp) //compare 710 to rbp-4

0x00000000004010dd <+202>: je 0x401140 <phase\_3+301> //if equal jump to line 301

0x00000000004010df <+204>: callq 0x401755 <explode\_bomb> //calls explode bomb

0x00000000004010e4 <+209>: mov $0x78,%eax //move 120 to eax

0x00000000004010e9 <+214>: jmp 0x401140 <phase\_3+301> //jump to line 301

0x00000000004010eb <+216>: mov $0x6f,%eax //move 111 to eax

0x00000000004010f0 <+221>: cmpl $0x2f0,-0x4(%rbp) //compare 752 to rbp-4

0x00000000004010f7 <+228>: je 0x401140 <phase\_3+301> //if equal jump to line 301

0x00000000004010f9 <+230>: callq 0x401755 <explode\_bomb> //calls explode bomb

0x00000000004010fe <+235>: mov $0x6f,%eax //move 111 to eax

0x0000000000401103 <+240>: jmp 0x401140 <phase\_3+301> //jump to line 301

0x0000000000401105 <+242>: mov $0x64,%eax //move 100 to eax

0x000000000040110a <+247>: cmpl $0x38,-0x4(%rbp) //compare 56 to rbp-4

0x000000000040110e <+251>: je 0x401140 <phase\_3+301> //if equal, jump to line 301

0x0000000000401110 <+253>: callq 0x401755 <explode\_bomb> //calls explode bomb

0x0000000000401115 <+258>: mov $0x64,%eax //move 100 to eax

0x000000000040111a <+263>: jmp 0x401140 <phase\_3+301> //jump to line 301

0x000000000040111c <+265>: mov $0x6f,%eax //move 111 to eax

0x0000000000401121 <+270>: cmpl $0x3df,-0x4(%rbp) //compare 991 to rbp-4

0x0000000000401128 <+277>: je 0x401140 <phase\_3+301> //if equal jump to line 301

0x000000000040112a <+279>: callq 0x401755 <explode\_bomb> //call explode bomb

0x000000000040112f <+284>: mov $0x6f,%eax //move 111 to eax

0x0000000000401134 <+289>: jmp 0x401140 <phase\_3+301> //jump to line 301

0x0000000000401136 <+291>: callq 0x401755 <explode\_bomb> //calls explode bomb

0x000000000040113b <+296>: mov $0x65,%eax //move 101 to eax

**0x0000000000401140 <+301>: cmp -0x9(%rbp),%al //compare rbp-9 to al**

0x0000000000401143 <+304>: je 0x40114a <phase\_3+311> //if equal, jump to line 311 to leave

0x0000000000401145 <+306>: callq 0x401755 <explode\_bomb> //calls explode bomb

0x000000000040114a <+311>: leaveq //leave function

0x000000000040114b <+312>: retq //return value

ISOC(?)

0x0000000000400cb0 <+0>: jmpq \*0x2033fa(%rip) # 0x6040b0 <\_\_isoc99\_sscanf@got.plt>

0x0000000000400cb6 <+6>: pushq $0x13 //push 19

0x0000000000400cbb <+11>: jmpq 0x400b70 //

**Phase 4—SOLVED**

**First check was seeing that we would explode if we had more than two inputs**

**Next check was seeing that we needed the first input to be <=14**

**Next step was seeing that we make a call to func4**

**Func4 is a recursive function that was checking sign bit**

**Next check after func4 is whether second input is equal to 7 after adding sign bits to it**

**Positive sign bit would not affect number, so positive seven would breeze through recursive function**

0x0000000000401187 <+0>: push %rbp

0x0000000000401188 <+1>: mov %rsp,%rbp //set up stack

0x000000000040118b <+4>: sub $0x10,%rsp //subtract 10 from rsp

0x000000000040118f <+8>: lea -0x4(%rbp),%rcx //point rbp-4 to rcx

0x0000000000401193 <+12>: lea -0x8(%rbp),%rdx //point rbp-8 to rdx

0x0000000000401197 <+16>: mov $0x402aed,%esi //move address value to esi

0x000000000040119c <+21>: mov $0x0,%eax //move 0 to eax

0x00000000004011a1 <+26>: callq 0x400cb0 <\_\_isoc99\_sscanf@plt> //call isoc to read inputs

0x00000000004011a6 <+31>: cmp $0x2,%eax //compare # of inputs to 2

0x00000000004011a9 <+34>: jne 0x4011b1 <phase\_4+42> //if input!=2, explode bomb

0x00000000004011ab <+36>: cmpl $0xe,-0x8(%rbp) //compare rbp-8 to 14

0x00000000004011af <+40>: jbe 0x4011b6 <phase\_4+47> //jump if = or less than (unsigned comparison) to line 47

0x00000000004011b1 <+42>: callq 0x401755 <explode\_bomb> //calls explode bomb

0x00000000004011b6 <+47>: mov $0xe,%edx // move 14 to edx

0x00000000004011bb <+52>: mov $0x0,%esi // move 0 to esi

0x00000000004011c0 <+57>: mov -0x8(%rbp),%edi //move rbp-8 to edi

0x00000000004011c3 <+60>: callq 0x40114c <func4> //calls func4 (see below)

0x00000000004011c8 <+65>: cmp $0x7,%eax //compare 7 to eax

0x00000000004011cb <+68>: jne 0x4011d3 <phase\_4+76> //if not equal, jump to explode bomb

0x00000000004011cd <+70>: cmpl $0x7,-0x4(%rbp) //compare 7 to rbp-4

0x00000000004011d1 <+74>: je 0x4011d8 <phase\_4+81> //if equal, jump to leave

0x00000000004011d3 <+76>: callq 0x401755 <explode\_bomb> //explodes bomb

0x00000000004011d8 <+81>: leaveq //leaves function

0x00000000004011d9 <+82>: retq //returns value

Func4

0x000000000040114c <+0>: push %rbp

0x000000000040114d <+1>: mov %rsp,%rbp //set up stack

0x0000000000401150 <+4>: mov %edx,%eax

0x0000000000401152 <+6>: sub %esi,%eax //subtract eax from esi

0x0000000000401154 <+8>: mov %eax,%ecx //move value of eax to ecx

0x0000000000401156 <+10>: shr $0x1f,%ecx //divide (right shift, unsigned) ecx by 31

0x0000000000401159 <+13>: add %ecx,%eax //add sign of ecx to eax (+1 or +0)

0x000000000040115b <+15>: sar %eax //right shift eax (signed)

0x000000000040115d <+17>: lea (%rax,%rsi,1),%ecx //load address of rsi(1)-rax to ecx

0x0000000000401160 <+20>: cmp %edi,%ecx //compare edi and ecx

0x0000000000401162 <+22>: jle 0x401170 <func4+36> //jump if edi<ecx to line 36

0x0000000000401164 <+24>: lea -0x1(%rcx),%edx //load address of rcx-1 to edx

0x0000000000401167 <+27>: callq 0x40114c <func4> //recursive call to func4

0x000000000040116c <+32>: add %eax,%eax //add eax to itself

0x000000000040116e <+34>: jmp 0x401185 <func4+57> //unconditional jump to line 57 (end of one case)

0x0000000000401170 <+36>: mov $0x0,%eax //move 0 to eax

0x0000000000401175 <+41>: cmp %edi,%ecx //compare edi to ecx

0x0000000000401177 <+43>: jge 0x401185 <func4+57> //if edi>ecx, jump to 57

0x0000000000401179 <+45>: lea 0x1(%rcx),%esi //load address rcx+1 to esi

0x000000000040117c <+48>: callq 0x40114c <func4> //recursive call to func4

0x0000000000401181 <+53>: lea 0x1(%rax,%rax,1),%eax //load address of (rax-rax)+1 to eax

0x0000000000401185 <+57>: pop %rbp //pop stack

0x0000000000401186 <+58>: retq //return

**Phase 5 –SOLVED**

**First check is that we need 6 inputs**

**“Anding” by 0xf truncates bits into 16 bit size**

**Array spans from 0 to 15**

**Add value from array (accessed in memory location) to each inputs**

**Array was accessed with x/16c**

**Created a table with A[1,1] on far left and A[n,n] on far right**

**Wrote down values and converted hex values to each number to find the cipher**

**Compare string value to 0x31=dec 49**

**Need to find inputs letters that would be added through array math to equal 49**

0x00000000004011da <+0>: push %rbp

0x00000000004011db <+1>: mov %rsp,%rbp //set up stack

0x00000000004011de <+4>: push %rbx

0x00000000004011df <+5>: sub $0x8,%rsp //subtract rsp-8

0x00000000004011e3 <+9>: mov %rdi,%rbx //move rdi to rbx

0x00000000004011e6 <+12>: callq 0x40143b <string\_length> //call to string length

0x00000000004011eb <+17>: cmp $0x6,%eax //compare 6 to eax

0x00000000004011ee <+20>: je 0x4011f5 <phase\_5+27> //if eax=6, jump past explode bomb

0x00000000004011f0 <+22>: callq 0x401755 <explode\_bomb> //call to explode bomb

0x00000000004011f5 <+27>: mov $0x0,%eax //move 0 to eax

0x00000000004011fa <+32>: mov $0x0,%edx //move 0 to edx

0x00000000004011ff <+37>: movzbl (%rbx,%rax,1),%ecx //move rax+rbx to ecx while changing from a byte to a long

0x0000000000401203 <+41>: and $0xf,%ecx //and ecx and 15

0x0000000000401206 <+44>: add 0x402860(,%rcx,4),%edx //add 4(rcx) +hex value to edx

0x000000000040120d <+51>: add $0x1,%rax //add one to rax

0x0000000000401211 <+55>: cmp $0x6,%rax //compare 6 to rax

0x0000000000401215 <+59>: jne 0x4011ff <phase\_5+37> //if not equal, jump back to data conversion on line 36

0x0000000000401217 <+61>: cmp $0x31,%edx //compare edx to 49

0x000000000040121a <+64>: je 0x401221 <phase\_5+71> //if equal, jump to line 71

0x000000000040121c <+66>: callq 0x401755 <explode\_bomb> //call to explode bomb

0x0000000000401221 <+71>: add $0x8,%rsp //add 8 to rsp

0x0000000000401225 <+75>: pop %rbx //

0x0000000000401226 <+76>: pop %rbp

0x0000000000401227 <+77>: retq

String length

0x000000000040143b <+0>: push %rbp //

0x000000000040143c <+1>: mov %rsp,%rbp //set up stack

0x000000000040143f <+4>: cmpb $0x0,(%rdi) //compare input to 0

0x0000000000401442 <+7>: je 0x401456 <string\_length+27> //if 0, jump to line 27

0x0000000000401444 <+9>: mov %rdi,%rdx //move input to rdx

0x0000000000401447 <+12>: add $0x1,%rdx //add one to rdx

0x000000000040144b <+16>: mov %edx,%eax //move edx to eax

0x000000000040144d <+18>: sub %edi,%eax //subtract eax-edi

0x000000000040144f <+20>: cmpb $0x0,(%rdx) //compare rdx to 0

0x0000000000401452 <+23>: jne 0x401447 <string\_length+12> //if not equal, jump back to line 12

0x0000000000401454 <+25>: jmp 0x40145b <string\_length+32> //unconditional jump to line 32

0x0000000000401456 <+27>: mov $0x0,%eax //move 0 to eax

0x000000000040145b <+32>: pop %rbp //pop rbp

0x000000000040145c <+33>: retq //return back to phase\_5

**Phase 6—SOLVED**

**How I solved it: entered 123456 knowing I needed 6 inputs**

**Saw that it was looping around checking each input**

**Need to figure out: why the jump to x/3c**

**Saw register rdx was being incremented by 8**

**x/3c on rdx, then plus 8, until added 5\*8**

**saw values stored in nodes, connected nodes to linked list**

**thought that they were comparing values in linked list**

**pointers on end told me that 3 was a null pointer**

**this was not needed however**

**used binary calculator to find each value and manually sorted by descending order**

**this was wrong, so I tried ascending order and it worked**

**Node 1:0x3f=63**

**Node 2: 0x1bd= 445**

**Node 3: 0x25c= 604**

**Node 4: 0x15d= 349**

**Node 5: 0x197=407**

**Node 6 :0x363=867**

**Order (ascending) = 1 4 5 2 3 6**

0x0000000000401228 <+0>: push %rbp

0x0000000000401229 <+1>: mov %rsp,%rbp //set up stack

0x000000000040122c <+4>: push %r14 //put r14 on stack

0x000000000040122e <+6>: push %r13 //put r13 on stack

0x0000000000401230 <+8>: push %r12 //put r12 on stack

0x0000000000401232 <+10>: push %rbx //put rbx on stack

0x0000000000401233 <+11>: sub $0x50,%rsp //stack is 80 bytes long

0x0000000000401237 <+15>: lea -0x70(%rbp),%rsi //load address of lea- (rbp-112)

0x000000000040123b <+19>: callq 0x40178b <read\_six\_numbers> //calls function read six numbers

0x0000000000401240 <+24>: lea -0x70(%rbp),%r14 //load address r14- (rbp-112)

0x0000000000401244 <+28>: mov $0x0,%r13d //move 0 to r13

0x000000000040124a <+34>: mov %r14,%r12 //move r14 to r12

0x000000000040124d <+37>: mov (%r14),%eax //move r14 address to eax

0x0000000000401250 <+40>: sub $0x1,%eax //subtract 1 from eax

0x0000000000401253 <+43>: cmp $0x5,%eax //compare eax to 5

0x0000000000401256 <+46>: jbe 0x40125d <phase\_6+53> //jump if lesser or equal (unsigned comparison)

0x0000000000401258 <+48>: callq 0x401755 <explode\_bomb> //calls explode bomb

0x000000000040125d <+53>: add $0x1,%r13d //add 1 to r13d

0x0000000000401261 <+57>: cmp $0x6,%r13d //compare r13d to 6

0x0000000000401265 <+61>: jne 0x40126e <phase\_6+70> //jump if not equal to line 70

0x0000000000401267 <+63>: mov $0x0,%esi //move 0 to esi

0x000000000040126c <+68>: jmp 0x4012b2 <phase\_6+138> //unconditional jump to line 138

0x000000000040126e <+70>: mov %r13d,%ebx //move r13d to ebx

0x0000000000401271 <+73>: movslq %ebx,%rax //move (change long to quad) ebx to rax

0x0000000000401274 <+76>: mov -0x70(%rbp,%rax,4),%eax //move (4rax-rbp)-112 to eax

0x0000000000401278 <+80>: cmp %eax,(%r12) //compare eax to address at r12

0x000000000040127c <+84>: jne 0x401283 <phase\_6+91> //jump if not equal to line 91, bypass explode bomb

0x000000000040127e <+86>: callq 0x401755 <explode\_bomb> //calls explode bomb

0x0000000000401283 <+91>: add $0x1,%ebx //add 1 to ebx

0x0000000000401286 <+94>: cmp $0x5,%ebx //compare 5 to ebx

0x0000000000401289 <+97>: jle 0x401271 <phase\_6+73> //jump if less than or equal back to line 73 (LOOP)

0x000000000040128b <+99>: add $0x4,%r14 //add 4 to r14

0x000000000040128f <+103>: jmp 0x40124a <phase\_6+34> //unconditional jump to line 34 (LOOP)

0x0000000000401291 <+105>: mov 0x8(%rdx),%rdx //move rdx+8 to rdx

0x0000000000401295 <+109>: add $0x1,%eax //add 1 to eax

0x0000000000401298 <+112>: cmp %ecx,%eax //compare ecx to eax

0x000000000040129a <+114>: jne 0x401291 <phase\_6+105> //jump if not equal to line 105

0x000000000040129c <+116>: jmp 0x4012a3 <phase\_6+123> //unconditional jump to line 123

0x000000000040129e <+118>: mov $0x604310,%edx //move address to edx

0x00000000004012a3 <+123>: mov %rdx,-0x50(%rbp,%rsi,2) //move rdx to (2rsi+rbp)-80

0x00000000004012a8 <+128>: add $0x4,%rsi //add 4 to rsi

0x00000000004012ac <+132>: cmp $0x18,%rsi //compare 24 to rsi

0x00000000004012b0 <+136>: je 0x4012c7 <phase\_6+159> //jump if equal to line 159, bypass loop

0x00000000004012b2 <+138>: mov -0x70(%rbp,%rsi,1),%ecx //move (rsi+rbp)-112 to ecx

0x00000000004012b6 <+142>: cmp $0x1,%ecx //compare 1 to ecx

0x00000000004012b9 <+145>: jle 0x40129e <phase\_6+118> //jump if less or equal to line 118 (LOOP)

0x00000000004012bb <+147>: mov $0x1,%eax //move 1 to eax

0x00000000004012c0 <+152>: mov $0x604310,%edx //move address to edx

0x00000000004012c5 <+157>: jmp 0x401291 <phase\_6+105> //unconditional jump to line 105

0x00000000004012c7 <+159>: mov -0x50(%rbp),%rbx //move rbp-80 to rbx

0x00000000004012cb <+163>: lea -0x48(%rbp),%rax //load address rbp-72 to rax

0x00000000004012cf <+167>: lea -0x20(%rbp),%rsi //load address rbp-32 to rsi

0x00000000004012d3 <+171>: mov %rbx,%rcx //move rbx to rcx

0x00000000004012d6 <+174>: mov (%rax),%rdx //move address of rax to rdx

0x00000000004012d9 <+177>: mov %rdx,0x8(%rcx) //move rdx to rcx+8

0x00000000004012dd <+181>: add $0x8,%rax //add rax+8

0x00000000004012e1 <+185>: cmp %rsi,%rax //compare rsi to rax

0x00000000004012e4 <+188>: je 0x4012eb <phase\_6+195> //jump if even to line 195

0x00000000004012e6 <+190>: mov %rdx,%rcx //move rdx to rcx

0x00000000004012e9 <+193>: jmp 0x4012d6 <phase\_6+174> unconditional jump to 174

0x00000000004012eb <+195>: movq $0x0,0x8(%rdx) //move 0 to rdx+8

0x00000000004012f3 <+203>: mov $0x5,%r12d //move 5 to r12d

0x00000000004012f9 <+209>: mov 0x8(%rbx),%rax //move rbx+8 to rax

0x00000000004012fd <+213>: mov (%rax),%eax //move address of rax to eax

0x00000000004012ff <+215>: cmp %eax,(%rbx) //compare eax to address to rbx

0x0000000000401301 <+217>: jle 0x401308 <phase\_6+224> //jump if less than equal to line 224, bypass explode bomb

0x0000000000401303 <+219>: callq 0x401755 <explode\_bomb> //call explode bomb

0x0000000000401308 <+224>: mov 0x8(%rbx),%rbx //move rbx+8 to rbx

0x000000000040130c <+228>: sub $0x1,%r12d //subtract r12d-1

0x0000000000401310 <+232>: jne 0x4012f9 <phase\_6+209> //jump if not equal to line 209, (LOOP)

0x0000000000401312 <+234>: add $0x50,%rsp //add rsp+80

0x0000000000401316 <+238>: pop %rbx //remove rbx

0x0000000000401317 <+239>: pop %r12 //remove r12

0x0000000000401319 <+241>: pop %r13 //remove r13

0x000000000040131b <+243>: pop %r14 //remove r14

0x000000000040131d <+245>: pop %rbp //remove rbp

0x000000000040131e <+246>: retq //return

**Secret phase:**

0x000000000040135a <+0>: push %rbp

0x000000000040135b <+1>: mov %rsp,%rbp //set up stack

0x000000000040135e <+4>: push %rbx //push rbx onto stack

0x000000000040135f <+5>: sub $0x8,%rsp //stack is bounded within 8 bytes

0x0000000000401363 <+9>: callq 0x4017d1 <read\_line> //call function read line

0x0000000000401368 <+14>: mov $0xa,%edx //move 10 into edx

0x000000000040136d <+19>: mov $0x0,%esi //move 0 into esi

0x0000000000401372 <+24>: mov %rax,%rdi //move 10 into rdi

0x0000000000401375 <+27>: callq 0x400c90 <strtol@plt> //call this…thing

0x000000000040137a <+32>: mov %rax,%rbx //move rax into rbx

0x000000000040137d <+35>: lea -0x1(%rax),%eax //load address of rax-1 into eax

0x0000000000401380 <+38>: cmp $0x3e8,%eax //compare 1000 to eax

0x0000000000401385 <+43>: jbe 0x40138c <secret\_phase+50> //if less than or equal (unsigned), jump to line 50, bypassing explode bomb

0x0000000000401387 <+45>: callq 0x401755 <explode\_bomb> //calls explode bomb

0x000000000040138c <+50>: mov %ebx,%esi //move ebx into esi

0x000000000040138e <+52>: mov $0x604130,%edi //move this address into edi

0x0000000000401393 <+57>: callq 0x40131f <fun7> //call fun7

0x0000000000401398 <+62>: cmp $0x7,%eax //compare 7 to eax

0x000000000040139b <+65>: je 0x4013a2 <secret\_phase+72> //if eax==7, jump to line 72, bypass explode bomb

0x000000000040139d <+67>: callq 0x401755 <explode\_bomb> //calls explode bomb

0x00000000004013a2 <+72>: mov $0x4027e0,%edi //move this address into edi

0x00000000004013a7 <+77>: callq 0x400bc0 <puts@plt> //call puts function, does…

0x00000000004013ac <+82>: callq 0x4018f4 <phase\_defused> //call phase defused to defuse entire bomb

0x00000000004013b1 <+87>: add $0x8,%rsp //begin removing memory from stack

0x00000000004013b5 <+91>: pop %rbx

0x00000000004013b6 <+92>: pop %rbp

0x00000000004013b7 <+93>: retq //victory

Fun7:

0x000000000040131f <+0>: push %rbp //push rbp

0x0000000000401320 <+1>: mov %rsp,%rbp //move rsp to rbp, set up further memory in stack

0x0000000000401323 <+4>: test %rdi,%rdi //rdi - rdi

0x0000000000401326 <+7>: je 0x401353 <fun7+52> //if ZF=0, then jump to line 52

0x0000000000401328 <+9>: mov (%rdi),%edx //move address of rdi into edx

0x000000000040132a <+11>: cmp %esi,%edx //compare esi to edx

0x000000000040132c <+13>: jle 0x40133b <fun7+28> //if less than or equal, jump to line 28

0x000000000040132e <+15>: mov 0x8(%rdi),%rdi //move rdi-8 to rdi

0x0000000000401332 <+19>: callq 0x40131f <fun7> //recursive call to fun7

0x0000000000401337 <+24>: add %eax,%eax //add eax to eax

0x0000000000401339 <+26>: jmp 0x401358 <fun7+57> //unconditional jump to line 57

0x000000000040133b <+28>: mov $0x0,%eax //move 0 to eax

0x0000000000401340 <+33>: cmp %esi,%edx //compare esi to edx

0x0000000000401342 <+35>: je 0x401358 <fun7+57> //if equal, jump to line 57

0x0000000000401344 <+37>: mov 0x10(%rdi),%rdi //move rdi+16 to rdi

0x0000000000401348 <+41>: callq 0x40131f <fun7> //recursive call to fun7

0x000000000040134d <+46>: lea 0x1(%rax,%rax,1),%eax //load address of rax+rax+1 to eax

0x0000000000401351 <+50>: jmp 0x401358 <fun7+57> //unconditional jump to line 57

0x0000000000401353 <+52>: mov $0xffffffff,%eax //move mask of ones to eax

0x0000000000401358 <+57>: pop %rbp //pop off rbp from stack

0x0000000000401359 <+58>: retq //return from fun7

Phase\_defused

0x00000000004018f4 <+0>: push %rbp

0x00000000004018f5 <+1>: mov %rsp,%rbp //set up stack

0x00000000004018f8 <+4>: sub $0x70,%rsp //stack is 112 bytes long

**0x00000000004018fc <+8>: mov %fs:0x28,%rax //black magic**

0x0000000000401905 <+17>: mov %rax,-0x8(%rbp) //move rax into rbp-8

0x0000000000401909 <+21>: xor %eax,%eax //exclusive or eax against itself

0x000000000040190b <+23>: mov $0x1,%edi //move 1 into edi

0x0000000000401910 <+28>: callq 0x401651 <send\_msg> //call function send\_msg

0x0000000000401915 <+33>: cmpl $0x6,0x202ea0(%rip) # 0x6047bc <num\_input\_strings> //compare 6 to rip+ some address

**0x000000000040191c <+40>: jne 0x401987 <phase\_defused+147> //If not equal, jump to line 147 and bypass setup and call to secret phase**

0x000000000040191e <+42>: lea -0x60(%rbp),%r8 //load address of rbp-96 to r8

0x0000000000401922 <+46>: lea -0x64(%rbp),%rcx //load address of rbp-100 to rcx

0x0000000000401926 <+50>: lea -0x68(%rbp),%rdx //load address of rbp-104 to rdx, possible array of ints?

0x000000000040192a <+54>: mov $0x402b37,%esi //move address to esi

0x000000000040192f <+59>: mov $0x6048d0,%edi //move address to edi

0x0000000000401934 <+64>: mov $0x0,%eax //move 0 to eax

0x0000000000401939 <+69>: callq 0x400cb0 <\_\_isoc99\_sscanf@plt> //call input scanner

0x000000000040193e <+74>: cmp $0x3,%eax //compare 3 to eax

**0x0000000000401941 <+77>: jne 0x401973 <phase\_defused+127> //if not equal, jump to line 127 and bypass secret phase**

0x0000000000401943 <+79>: mov $0x402b40,%esi //move address to esi

0x0000000000401948 <+84>: lea -0x60(%rbp),%rdi //load address rbp-96 to rdi

0x000000000040194c <+88>: callq 0x40145d <strings\_not\_equal> //call strings not equal to check input

0x0000000000401951 <+93>: test %eax,%eax //eax-eax

0x0000000000401953 <+95>: jne 0x401973 <phase\_defused+127> //if not equal

0x0000000000401955 <+97>: mov $0x402998,%edi

0x000000000040195a <+102>: callq 0x400bc0 <puts@plt>

0x000000000040195f <+107>: mov $0x4029c0,%edi

0x0000000000401964 <+112>: callq 0x400bc0 <puts@plt>

0x0000000000401969 <+117>: mov $0x0,%eax

0x000000000040196e <+122>: callq 0x40135a <secret\_phase>

0x0000000000401973 <+127>: mov $0x4029f8,%edi

0x0000000000401978 <+132>: callq 0x400bc0 <puts@plt>

0x000000000040197d <+137>: mov $0x402a28,%edi

0x0000000000401982 <+142>: callq 0x400bc0 <puts@plt>

0x0000000000401987 <+147>: mov -0x8(%rbp),%rax

0x000000000040198b <+151>: xor %fs:0x28,%rax

0x0000000000401994 <+160>: je 0x40199b <phase\_defused+167>

0x0000000000401996 <+162>: callq 0x400be0 <\_\_stack\_chk\_fail@plt>

0x000000000040199b <+167>: leaveq

0x000000000040199c <+168>: nopl 0x0(%rax)

0x00000000004019a0 <+172>: retq