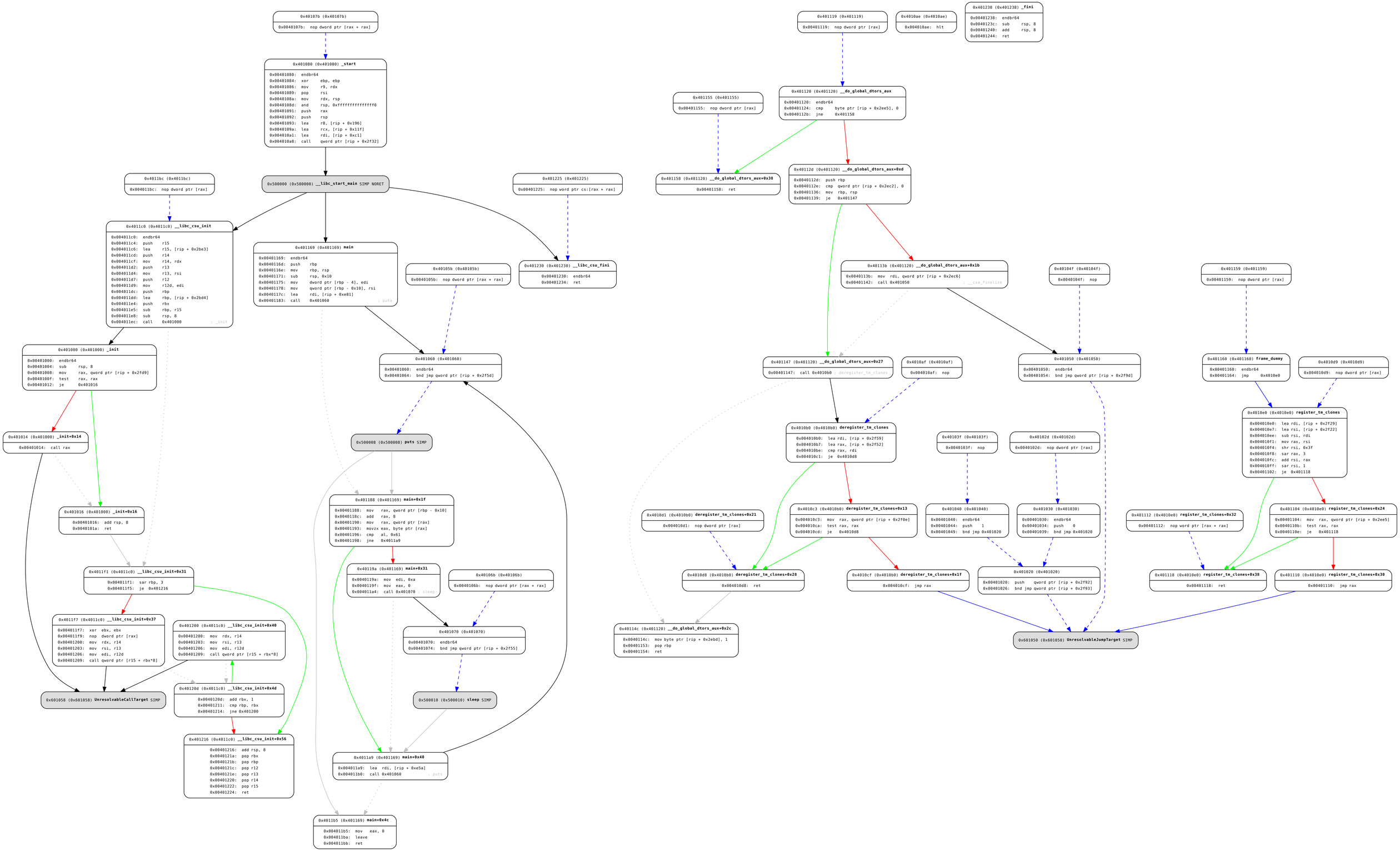
0x40107b (0x40107b)
0x0040107b: nop dword ptr [rax + rax]
0x401080 (0x401080) _start
0x00401080: endbr64
0x00401084: xor ebp, ebp
0x00401086: mov r9, rdx
0x00401089: pop rsi
0x0040108a: mov rdx, rsp
0x0040108d: and rsp, 0xfffffffffffffff0
0x00401091: push rax
0x00401092: push rsp
0x00401093: lea r8, [rip + 0x196]
0x0040109a: lea rcx, [rip + 0x11f]
0x004010a1: lea rdi, [rip + 0xc1]
0x004010a8: call qword ptr [rip + 0x2f32]
0x4011bc (0x4011bc)
0x004011bc: nop dword ptr [rax]
0x500000 (0x500000) __libc_start_main SIMP NORET
0x401225 (0x401225)
0x00401225: nop word ptr cs:[rax + rax]
0x4011c0 (0x4011c0) __libc_csu_init
0x004011c0: endbr64
0x004011c4: push r15
0x004011c6: lea r15, [rip + 0x2be3]
0x004011cd: push r14
0x004011cf: mov r14, rdx
0x004011d2: push r13
0x004011d4: mov r13, rsi
0x004011d7: push r12
0x004011d9: mov r12d, edi
0x004011dc: push rbp
0x004011dd: lea rbp, [rip + 0x2bd4]
0x004011e4: push rbx
0x004011e5: sub rbp, r15
0x004011e8: sub rsp, 8
0x004011ec: call 0x401000 ; _init
0x401169 (0x401169) main
0x00401169: endbr64
0x0040116d: push rbp
0x0040116e: mov rbp, rsp
0x00401171: sub rsp, 0x10
0x00401175: mov dword ptr [rbp - 4], edi
0x00401178: mov qword ptr [rbp - 0x10], rsi
0x0040117c: lea rdi, [rip + 0xe81]
0x00401183: call 0x401060 ; puts
0x40105b (0x40105b)
0x0040105b: nop dword ptr [rax + rax]
0x401230 (0x401230) __libc_csu_fini
0x00401230: endbr64
0x00401234: ret
0x401000 (0x401000) _init
0x00401000: endbr64
0x00401004: sub rsp, 8
0x00401008: mov rax, qword ptr [rip + 0x2fd9]
0x0040100f: test rax, rax
0x00401012: je 0x401016
0x401060 (0x401060)
0x00401060: endbr64
0x00401064: bnd jmp qword ptr [rip + 0x2f5d]
0x401014 (0x401000) _init+0x14
0x00401014: call rax
0x500008 (0x500008) puts SIMP
0x401016 (0x401000) _init+0x16
0x00401016: add rsp, 8
0x0040101a: ret
0x401188 (0x401169) main+0x1f
0x00401188: mov rax, qword ptr [rbp - 0x10]
0x0040118c: add rax, 8
0x00401190: mov rax, qword ptr [rax]
0x00401193: movzx eax, byte ptr [rax]
0x00401196: cmp al, 0x61
0x00401198: jne 0x4011a9
0x4011f1 (0x4011c0) __libc_csu_init+0x31
0x004011f1: sar rbp, 3
0x004011f5: je 0x401216
0x40119a (0x401169) main+0x31
0x0040119a: mov edi, 0xa
0x0040119f: mov eax, 0
0x004011a4: call 0x401070 ; sleep
0x40106b (0x40106b)
0x0040106b: nop dword ptr [rax + rax]
0x4011f7 (0x4011c0) __libc_csu_init+0x37
0x004011f7: xor ebx, ebx
0x004011f9: nop dword ptr [rax]
0x00401200: mov rdx, r14
0x00401203: mov rsi, r13
0x00401206: mov edi, r12d
0x00401209: call qword ptr [r15 + rbx*8]
0x401200 (0x4011c0) __libc_csu_init+0x40
0x00401200: mov rdx, r14
0x00401203: mov rsi, r13
0x00401206: mov edi, r12d
0x00401209: call qword ptr [r15 + rbx*8]
0x401070 (0x401070)
0x00401070: endbr64
0x00401074: bnd jmp qword ptr [rip + 0x2f55]
0x601058 (0x601058) UnresolvableCallTarget SIMP
0x40120d (0x4011c0) __libc_csu_init+0x4d
0x0040120d: add rbx, 1
0x00401211: cmp rbp, rbx
0x00401214: jne 0x401200
0x500010 (0x500010) sleep SIMP
0x401216 (0x4011c0) __libc_csu_init+0x56
0x00401216: add rsp, 8
0x0040121a: pop rbx
0x0040121b: pop rbp
0x0040121c: pop r12
0x0040121e: pop r13
0x00401220: pop r14
0x00401222: pop r15
0x00401224: ret
0x4011a9 (0x401169) main+0x40
0x004011a9: lea rdi, [rip + 0xe5a]
0x004011b0: call 0x401060 ; puts
0x4011b5 (0x401169) main+0x4c
0x004011b5: mov eax, 0
0x004011ba: leave
0x004011bb: ret
0x401119 (0x401119)
0x00401119: nop dword ptr [rax]
0x4010ae (0x4010ae)
0x004010ae: hlt
0x401238 (0x401238) _fini
0x00401238: endbr64
0x0040123c: sub rsp, 8
0x00401240: add rsp, 8
0x00401244: ret
0x401155 (0x401155)
0x00401155: nop dword ptr [rax]
0x401120 (0x401120) __do_global_dtors_aux
0x00401120: endbr64
0x00401124: cmp byte ptr [rip + 0x2ee5], 0
0x0040112b: jne 0x401158
0x401158 (0x401120) __do_global_dtors_aux+0x38
0x00401158: ret
0x40112d (0x401120) __do_global_dtors_aux+0xd
0x0040112d: push rbp
0x0040112e: cmp qword ptr [rip + 0x2ec2], 0
0x00401136: mov rbp, rsp
0x00401139: je 0x401147
0x40113b (0x401120) __do_global_dtors_aux+0x1b
0x0040113b: mov rdi, qword ptr [rip + 0x2ec6]
0x00401142: call 0x401050 ; __cxa_finalize
0x40104f (0x40104f)
0x0040104f: nop
0x401159 (0x401159)
0x00401159: nop dword ptr [rax]
0x401147 (0x401120) __do_global_dtors_aux+0x27
0x00401147: call 0x4010b0 ; deregister_tm_clones
0x4010af (0x4010af)
0x004010af: nop
0x401050 (0x401050)
0x00401050: endbr64
0x00401054: bnd jmp qword ptr [rip + 0x2f9d]
0x401160 (0x401160) frame_dummy
0x00401160: endbr64
0x00401164: jmp 0x4010e0
0x4010d9 (0x4010d9)
0x004010d9: nop dword ptr [rax]
0x4010b0 (0x4010b0) deregister_tm_clones
0x004010b0: lea rdi, [rip + 0x2f59]
0x004010b7: lea rax, [rip + 0x2f52]
0x004010be: cmp rax, rdi
0x004010c1: je 0x4010d8
0x4010e0 (0x4010e0) register_tm_clones
0x004010e0: lea rdi, [rip + 0x2f29]
0x004010e7: lea rsi, [rip + 0x2f22]
0x004010ee: sub rsi, rdi
0x004010f1: mov rax, rsi
0x004010f4: shr rsi, 0x3f
0x004010f8: sar rax, 3
0x004010fc: add rsi, rax
0x004010ff: sar rsi, 1
0x00401102: je 0x401118
0x40103f (0x40103f)
0x0040103f: nop
0x40102d (0x40102d)
0x0040102d: nop dword ptr [rax]
0x4010d1 (0x4010b0) deregister_tm_clones+0x21
0x004010d1: nop dword ptr [rax]
0x4010c3 (0x4010b0) deregister_tm_clones+0x13
0x004010c3: mov rax, qword ptr [rip + 0x2f0e]
0x004010ca: test rax, rax
0x004010cd: je 0x4010d8
0x401040 (0x401040)
0x00401040: endbr64
0x00401044: push 1
0x00401049: bnd jmp 0x401020
0x401030 (0x401030)
0x00401030: endbr64
0x00401034: push 0
0x00401039: bnd jmp 0x401020
0x401112 (0x4010e0) register_tm_clones+0x32
0x00401112: nop word ptr [rax + rax]
0x401104 (0x4010e0) register_tm_clones+0x24
0x00401104: mov rax, qword ptr [rip + 0x2ee5]
0x0040110b: test rax, rax
0x0040110e: je 0x401118
0x4010d8 (0x4010b0) deregister_tm_clones+0x28
0x004010d8: ret
0x4010cf (0x4010b0) deregister_tm_clones+0x1f
0x004010cf: jmp rax
0x401020 (0x401020)
0x00401020: push qword ptr [rip + 0x2f92]
0x00401026: bnd jmp qword ptr [rip + 0x2f93]
0x401118 (0x4010e0) register_tm_clones+0x38
0x00401118: ret
0x401110 (0x4010e0) register_tm_clones+0x30
0x00401110: jmp rax
0x114c (0x401120) __do_global_dtors_aux+0x2c
0x0040114c: mov byte ptr [rip + 0x2ebd], 1
0x00401153: pop rbp
0x00401154: ret
0x601050 (0x601050) UnresolvableJumpTarget SIMP