MONTANA STATE UNIVERSITY

10/30/2025

Malicious Code Analysis

Fangtian Zhong
CSCI 591

Gianforte School of Computing
Norm Asbjornson College of Engineering
E-mail: fangtian.zhong@montana.edu

Overview

01 Factory

02 Analysis

03 User Mode Debugging

Part One

01

Factory

2025-10-13

Factory

```
“””
factory
“””
block = proj.factory.block(proj.entry)
print(block.pp())
print(block.instructions)
print(block.instruction_addrs)
print(block.size)
block.capstone.pp()
block.vex.pp()
```

Break Type

```
#get our state
s = proj.factory.entry_state()
“””
on the other hand, we can have a breakpoint trigger right *after* a memory write happens.
we can also have a callback function run instead of opening ipdb
“””
def debug_func(state):
        print(“State %s is about to do a syscall!”)

s.inspect.b(‘syscall’, when=angr.BP_BEFORE, action=debug_func)
#or, you can have it drop you in an embedded IPython!
s.inspect.b(‘syscall’, when=angr.BP_BEFORE, action=angr.BP_IPYTHON)
```

Break Type

```
def track_reads(state):
        print('Read', state.inspect.mem_read_expr, 'from', state.inspect.mem_read_address)

s.inspect.b('mem_read', when=angr.BP_AFTER, action=track_reads)

"""
This will break before a memory write if 0x1000 is a possible
value of its target expression
"""
s.inspect.b('mem_write', mem_write_address=0x1000)
"""
This will break before a memory write if 0x1000 is the only
value of its target expression
"""
s.inspect.b('mem_write', mem_write_address=0x1000, mem_write_address_unique=true)
"""
This will break after instruction 0x8000, but only 0x1000 is possible value of the last expression
was read from memory
"""
s.inspect.b('instruction', when=angr.BP_AFTER, instruction=0x8000, mem_read_expr=0x1000)
```

Break Type

```
“””
this is a comple condition that could do anything! In this case, it makes sure that
RAX is 0x41414141 and that the basic block starting at 0x8004 was executed
sometime in this path’s history
“””
def cond(state):
    return state.eval(state.regs.rax, cast_to=str)==’AAAA’ and 0x8004 in state.inspect.backtrace

s.inspect.b(‘mem_write’, condition=cond)
```

Part Two

02

Analysis

Analyses

angr provides a number of analyses:

```
>> p.analyses.[TAB]
p.analyses.BackwardSlice        p.analyses.CFGFast            p.analyses.Reassembler
p.analyses.BinaryOptimizer      p.analyses.CongruencyCheck
p.analyses.reload_analyses
p.analyses.BinDiff              p.analyses.DDG                p.analyses.StaticHooker
p.analyses.BoyScout             p.analyses.DFG                p.analyses.VariableRecovery
p.analyses.CalleeCleanupFinder  p.analyses.Disassembly
p.analyses.VariableRecoveryFast
p.analyses.CDG                  p.analyses.GirlScout          p.analyses.Veritesting
p.analyses.CFG                  p.analyses.Identifier         p.analyses.VFG
p.analyses.CFGAccurate          p.analyses.LoopFinder         p.analyses.VSA_DDG
```

The details about the analysis can be found in the API documentation

CFG

```
“””
analysis
“””
cfg = proj.analysis.CFGFast()
graph = cfg.graph
print(len(graph.nodes()))
entry_node = cfg.get_any_node(proj.entry)
print(len(list(cfg.graph.successors(entry_node))))
```

CFG

```
cfg = proj.analysis.CFGEmulated()
graph = cfg.graph
print(len(graph.edges))
entry_node = cfg.get_any_node(proj.entry)
print(len(list(cfg.graph.successors(entry_node))))
```

Part Three

03

User Mode Debugging

2025-10-13

install dependencies

```
git clone https://github.com/axt/bingraphvis
pip install -e ./bingraphvis
git clone https://github.com/axt/angr-utils
pip install -e ./angr-utils
```

CG

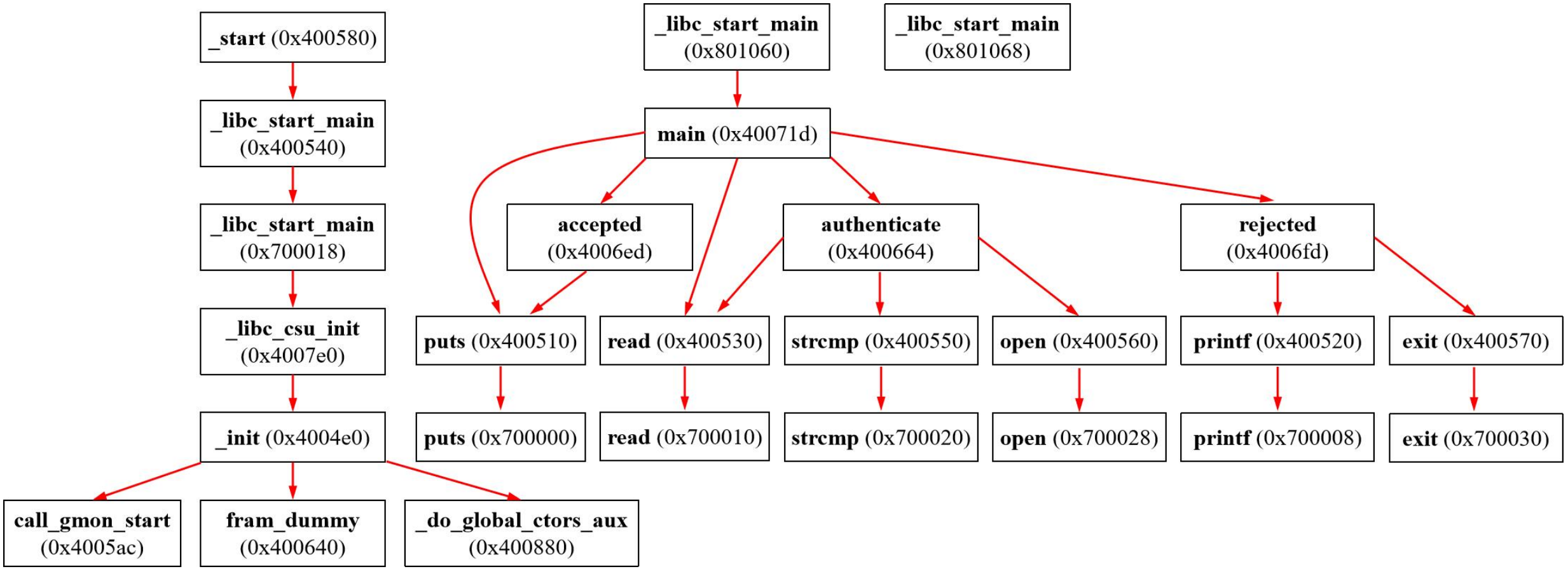
```
def analyze(b, addr, name=None):
    start_state = b.factory.blank_state(addr=addr)
    start_state.stack_push(0x0)
    plot_cg(b.kb, "%s_callgraph" % name, format="pdf")
    plot_cg(b.kb, "%s_callgraph_verbose" % name, format="pdf", verbose=True)


proj = angr.Project('backdoor1', auto_load_libs=False)
analyze(proj, proj.entry, "backdoor")
```

Source Code

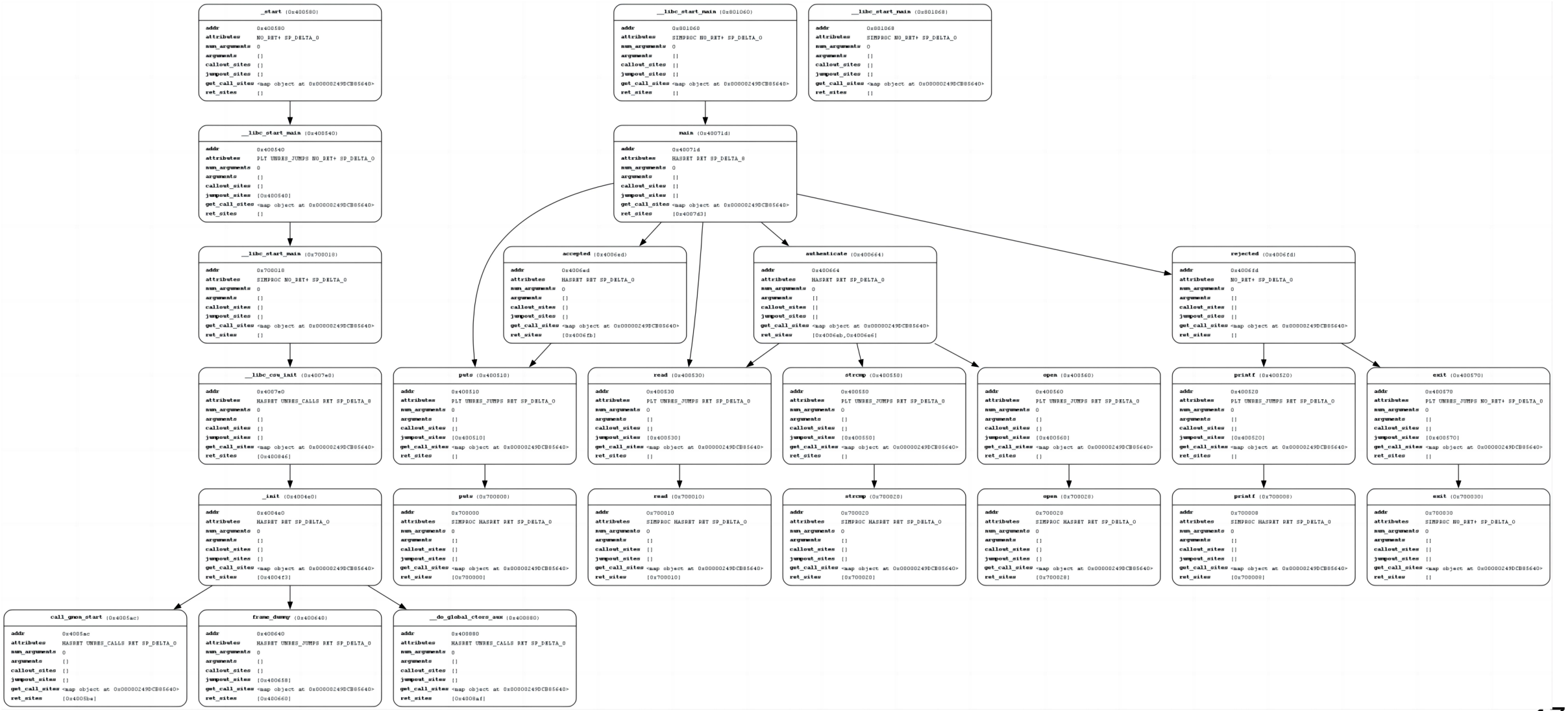
```
char *sneaky = "SOSNEAKY";
int authenticate(char *username, char *password)
{
        char stored_pw[9];
        stored_pw[8] = 0;
        int pwfile;

        // evil back d00r
        if (strcmp(password, sneaky) == 0) return 1;

        pwfile = open(username, O_RDONLY);
        read(pwfile, stored_pw, 8);

        if (strcmp(password, stored_pw) == 0) return 1;
        return 0;
}

int accepted()
{
        printf("Welcome to the admin console, trusted user!\n");
}
int rejected()
{
        printf("Go away!");
        exit(1);
}
```

```
int main(int argc, char **argv)
{
        char username[9];
        char password[9];
        int authed;

        username[8] = 0;
        password[8] = 0;

        printf("Username: \n");
        read(0, username, 8);
        read(0, &authed, 1);
        printf("Password: \n");
        read(0, password, 8);
        read(0, &authed, 1);

        authed = authenticate(username, password);
        if (authed) accepted();
        else rejected();
}
```

CG


_start (0x400580)
_libc_start_main (0x400540)
_libc_start_main (0x700018)
_libc_csu_init (0x4007e0)
_init (0x4004e0)
call_gmon_start (0x4005ac)
fram_dummy (0x400640)
_do_global_ctors_aux (0x400880)
_libc_start_main (0x801060)
_libc_start_main (0x801068)
main (0x40071d)
accepted (0x4006ed)
authenticate (0x400664)
rejected (0x4006fd)
puts (0x400510)
read (0x400530)
strcmp (0x400550)
open (0x400560)
printf (0x400520)
exit (0x400570)
puts (0x700000)
read (0x700010)
strcmp (0x700020)
open (0x700028)
printf (0x700008)
exit (0x700030)

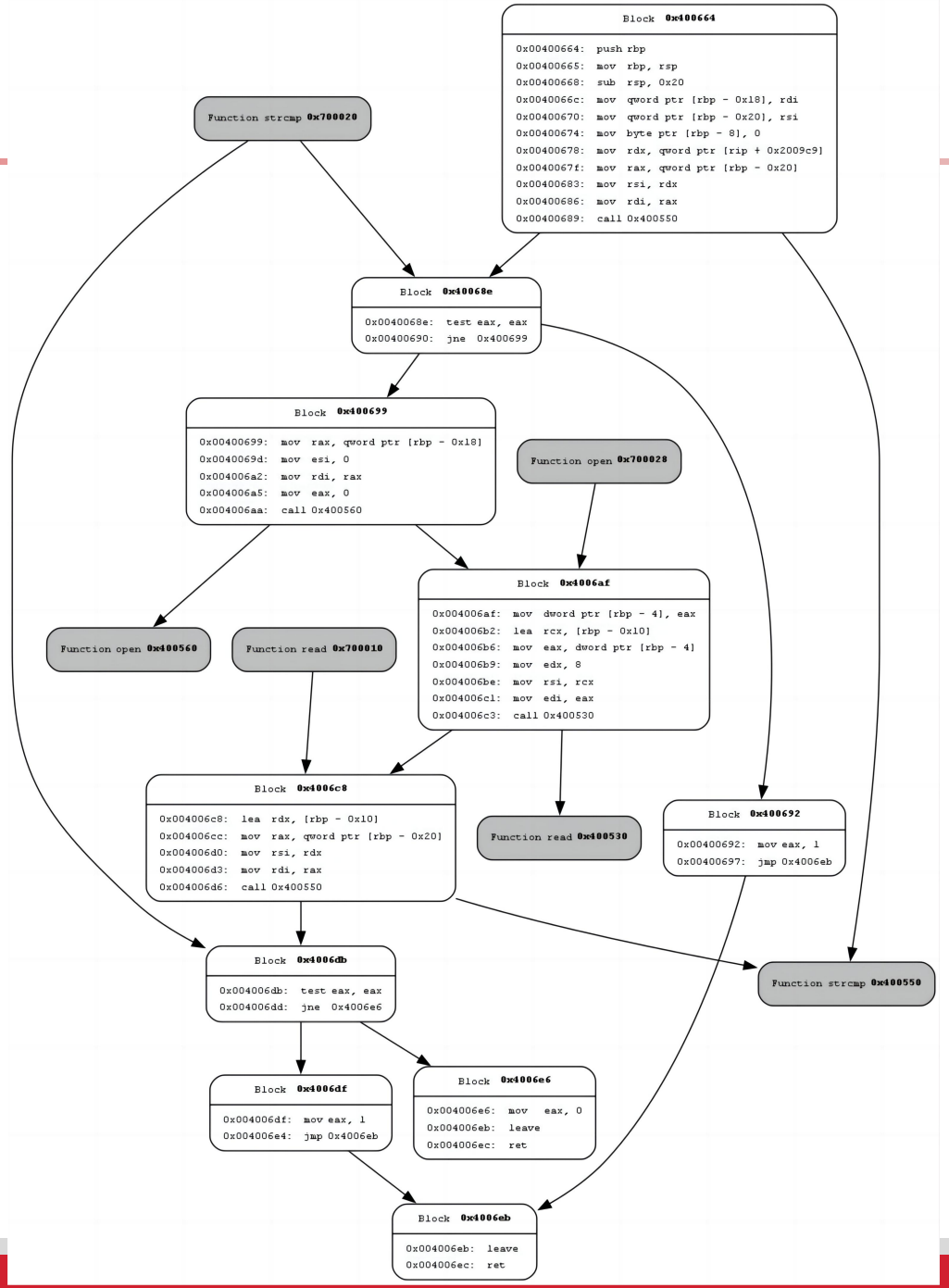
Call Graph (CG)



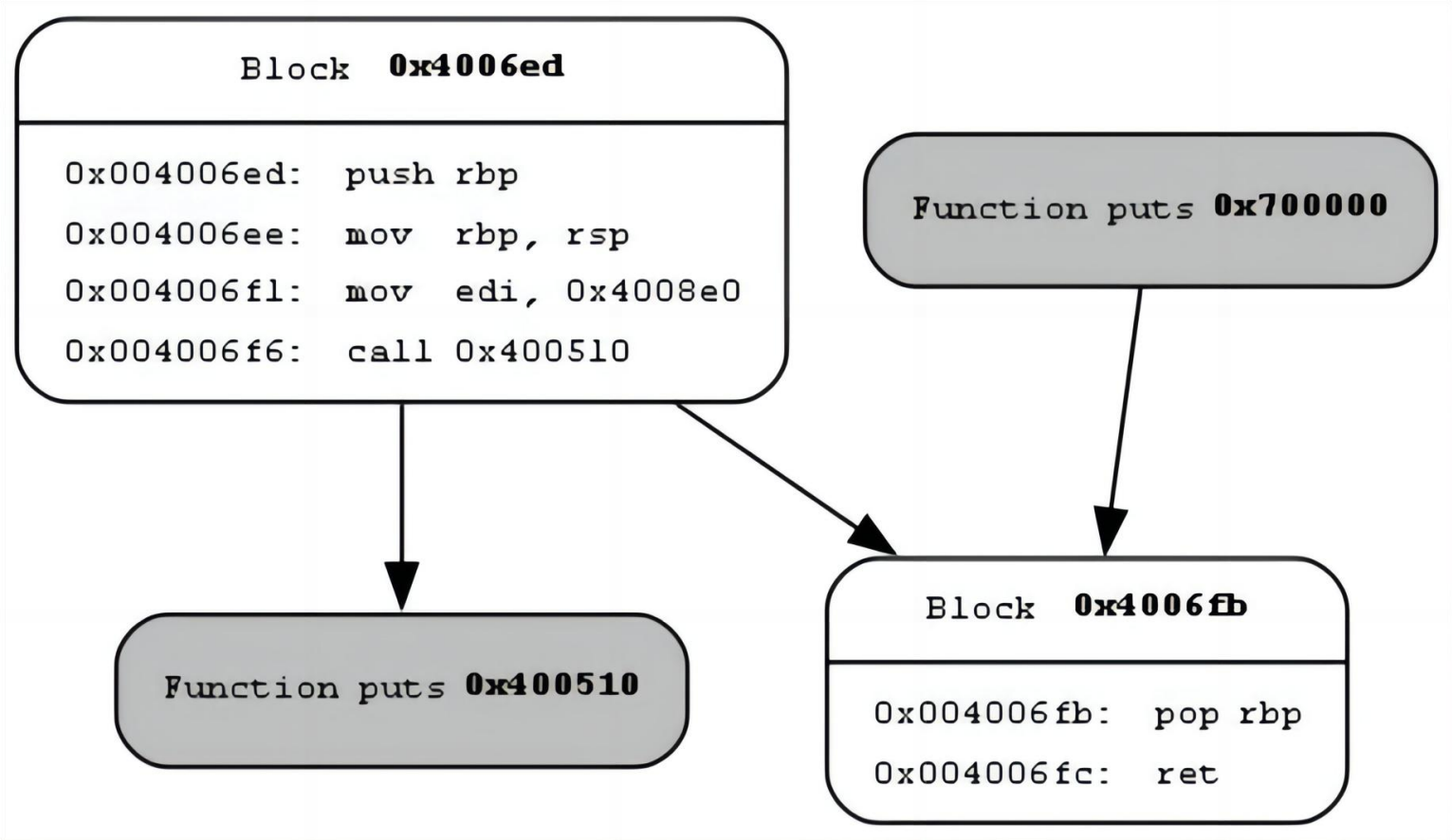
CFG-function

```
def analyze(b, name):
    for func in proj.kb.functions.values():
        if func.name.find(name) == 0:
            plot_func_graph(b, func.transition_graph, "%s_%s_cfg" % (name, func.name), asminst=True, vexinst=False)


proj = angr.Project('backdoor1', auto_load_libs=False)
analyze(proj, "authenticate")
```

CFG-authenticate



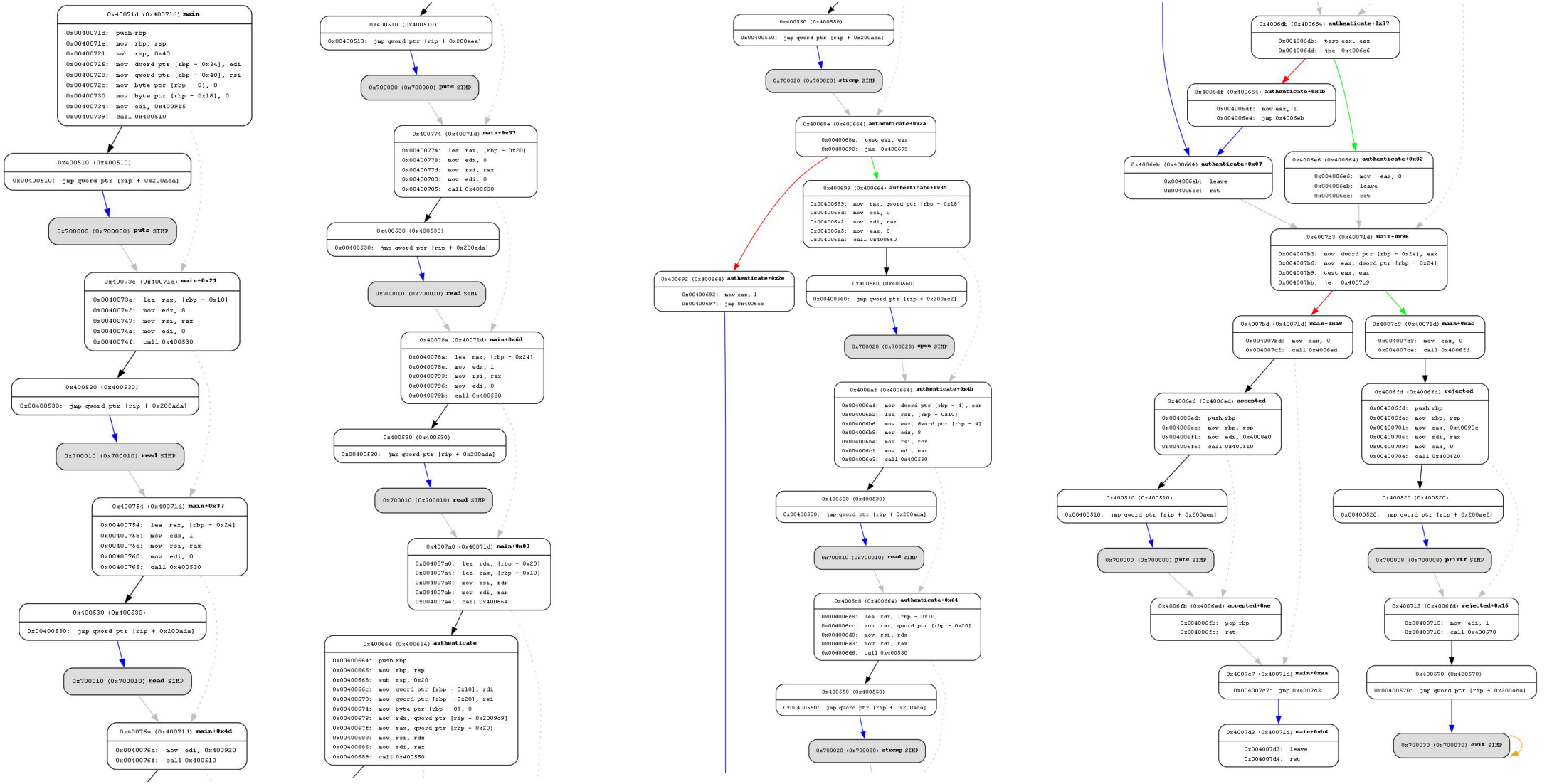
CFG-accepted



CFG-binary

```
    def analyze(b, name, func_name):
        main = b.loader.main_object.get_symbol(func_name)
        start_state = b.factory.blank_state(addr=main.rebased_addr)
        cfg = b.analyses.CFGEmulated(fail_fast=True,
        starts=[main.rebased_addr], initial_state=start_state)
        plot_cfg(cfg, "name", asminst=True, remove_imports=True,
remove_path_terminator=True)


proj = angr.Project('backdoor1', auto_load_libs=False)
analyze(proj, “backdoor_cfg”, “main”)
```

CFG


0x40071d (0x40071d) main
0x0040071d: push rbp
0x0040071e: mov rbp, rsp
0x00400721: sub rsp, 0x40
0x00400725: mov dword ptr [rbp - 0x34], edi
0x00400728: mov qword ptr [rbp - 0x40], rsi
0x0040072c: mov byte ptr [rbp - 8], 0
0x00400730: mov byte ptr [rbp - 0x18], 0
0x00400734: mov edi, 0x400915
0x00400739: call 0x400510
0x400510 (0x400510)
0x00400510: jmp qword ptr [rip + 0x200aea]
0x700000 (0x700000) puts SIMP
0x40073e (0x40071d) main+0x21
0x0040073e: lea rax, [rbp - 0x10]
0x00400742: mov edx, 8
0x00400747: mov rsi, rax
0x0040074a: mov edi, 0
0x0040074f: call 0x400530
0x400530 (0x400530)
0x00400530: jmp qword ptr [rip + 0x200ada]
0x700010 (0x700010) read SIMP
0x400754 (0x40071d) main+0x37
0x00400754: lea rax, [rbp - 0x24]
0x00400758: mov edx, 1
0x0040075d: mov rsi, rax
0x00400760: mov edi, 0
0x00400765: call 0x400530
0x400530 (0x400530)
0x00400530: jmp qword ptr [rip + 0x200ada]
0x700010 (0x700010) read SIMP
0x40076a (0x40071d) main+0x4d
0x0040076a: mov edi, 0x400920
0x0040076f: call 0x400510
0x400510 (0x400510)
0x00400510: jmp qword ptr [rip + 0x200aea]
0x700000 (0x700000) puts SIMP
0x400774 (0x40071d) main+0x57
0x00400774: lea rax, [rbp - 0x20]
0x00400778: mov edx, 8
0x0040077d: mov rsi, rax
0x00400780: mov edi, 0
0x00400785: call 0x400530
0x400530 (0x400530)
0x00400530: jmp qword ptr [rip + 0x200ada]
0x700010 (0x700010) read SIMP
0x40078a (0x40071d) main+0x6d
0x0040078a: lea rax, [rbp - 0x24]
0x0040078e: mov edx, 1
0x00400793: mov rsi, rax
0x00400796: mov edi, 0
0x0040079b: call 0x400530
0x400530 (0x400530)
0x00400530: jmp qword ptr [rip + 0x200ada]
0x700010 (0x700010) read SIMP
0x4007a0 (0x40071d) main+0x83
0x004007a0: lea rdx, [rbp - 0x20]
0x004007a4: lea rax, [rbp - 0x10]
0x004007a8: mov rsi, rdx
0x004007ab: mov rdi, rax
0x004007ae: call 0x400664
0x400664 (0x400664) authenticate
0x00400664: push rbp
0x00400665: mov rbp, rsp
0x00400668: sub rsp, 0x20
0x0040066c: mov qword ptr [rbp - 0x18], rdi
0x00400670: mov qword ptr [rbp - 0x20], rsi
0x00400674: mov byte ptr [rbp - 8], 0
0x00400678: mov rdx, qword ptr [rip + 0x2009c9]
0x0040067f: mov rax, qword ptr [rbp - 0x20]
0x00400683: mov rsi, rdx
0x00400686: mov rdi, rax
0x00400689: call 0x400550
0x400550 (0x400550)
0x00400550: jmp qword ptr [rip + 0x200aca]
0x700020 (0x700020) strcmp SIMP
0x40068e (0x400664) authenticate+0x2a
0x0040068e: test eax, eax
0x00400690: jne 0x400699
0x400692 (0x400664) authenticate+0x2e
0x00400692: mov eax, 1
0x00400697: jmp 0x4006eb
0x400699 (0x400664) authenticate+0x35
0x00400699: mov rax, qword ptr [rbp - 0x18]
0x0040069d: mov esi, 0
0x004006a2: mov rdi, rax
0x004006a5: mov eax, 0
0x004006aa: call 0x400560
0x400560 (0x400560)
0x00400560: jmp qword ptr [rip + 0x200ac2]
0x700028 (0x700028) open SIMP
0x4006af (0x400664) authenticate+0xdb
0x004006af: mov dword ptr [rbp - 4], eax
0x004006b2: lea rcx, [rbp - 0x10]
0x004006b6: mov eax, dword ptr [rbp - 4]
0x004006b9: mov edx, 8
0x004006be: mov rsi, rcx
0x004006c1: mov edi, eax
0x004006c3: call 0x400530
0x400530 (0x400530)
0x00400530: jmp qword ptr [rip + 0x200ada]
0x700010 (0x700010) read SIMP
0x4006c8 (0x400664) authenticate+0x64
0x004006c8: lea rdx, [rbp - 0x10]
0x004006cc: mov rax, qword ptr [rbp - 0x20]
0x004006d0: mov rsi, rdx
0x004006d3: mov rdi, rax
0x004006d6: call 0x400550
0x400550 (0x400550)
0x00400550: jmp qword ptr [rip + 0x200aca]
0x700020 (0x700020) strcmp SIMP
0x4006db (0x400664) authenticate+0x77
0x004006db: test eax, eax
0x004006dd: jne 0x4006e6
0x4006df (0x400664) authenticate+0x7b
0x004006df: mov eax, 1
0x004006e4: jmp 0x4006eb
0x4006eb (0x400664) authenticate+0x87
0x004006eb: leave
0x004006ec: ret
0x4006e6 (0x400664) authenticate+0x82
0x004006e6: mov eax, 0
0x004006eb: leave
0x004006ec: ret
0x4007b3 (0x40071d) main+0x96
0x004007b3: mov dword ptr [rbp - 0x24], eax
0x004007b6: mov eax, dword ptr [rbp - 0x24]
0x004007b9: test eax, eax
0x004007bb: je 0x4007c9
0x4007bd (0x40071d) main+0xa0
0x004007bd: mov eax, 0
0x004007c2: call 0x4006ed
0x4007c9 (0x40071d) main+0xac
0x004007c9: mov eax, 0
0x004007ce: call 0x4006fd
0x4006ed (0x4006ed) accepted
0x004006ed: push rbp
0x004006ee: mov rbp, rsp
0x004006f1: mov edi, 0x4008e0
0x004006f6: call 0x400510
0x4006fd (0x4006fd) rejected
0x004006fd: push rbp
0x004006fe: mov rbp, rsp
0x00400701: mov eax, 0x40090c
0x00400706: mov rdi, rax
0x00400709: mov eax, 0
0x0040070e: call 0x400520
0x400510 (0x400510)
0x00400510: jmp qword ptr [rip + 0x200aea]
0x700000 (0x700000) puts SIMP
0x400520 (0x400520)
0x00400520: jmp qword ptr [rip + 0x200ae2]
0x700008 (0x700008) printf SIMP
0x4006fb (0x4006ed) accepted+0xe
0x004006fb: pop rbp
0x004006fc: ret
0x400713 (0x4006fd) rejected+0x16
0x00400713: mov edi, 1
0x00400718: call 0x400570
0x4007c7 (0x40071d) main+0xaa
0x004007c7: jmp 0x4007d3
0x400570 (0x400570)
0x00400570: jmp qword ptr [rip + 0x200aba]
0x4007d3 (0x40071d) main+0xb6
0x004007d3: leave
0x004007d4: ret
0x700030 (0x700030) exit SIMP

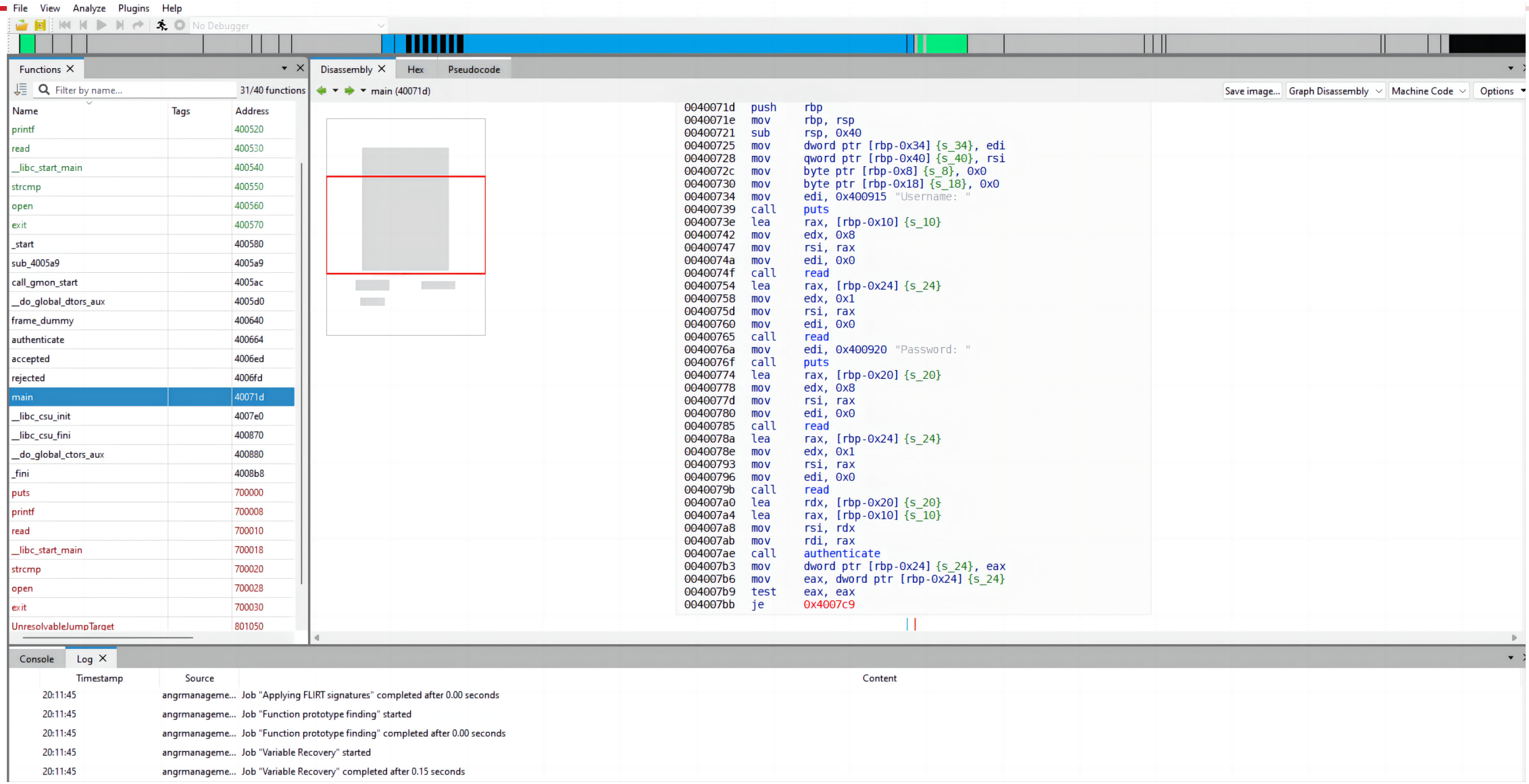
angr-management

- installation:

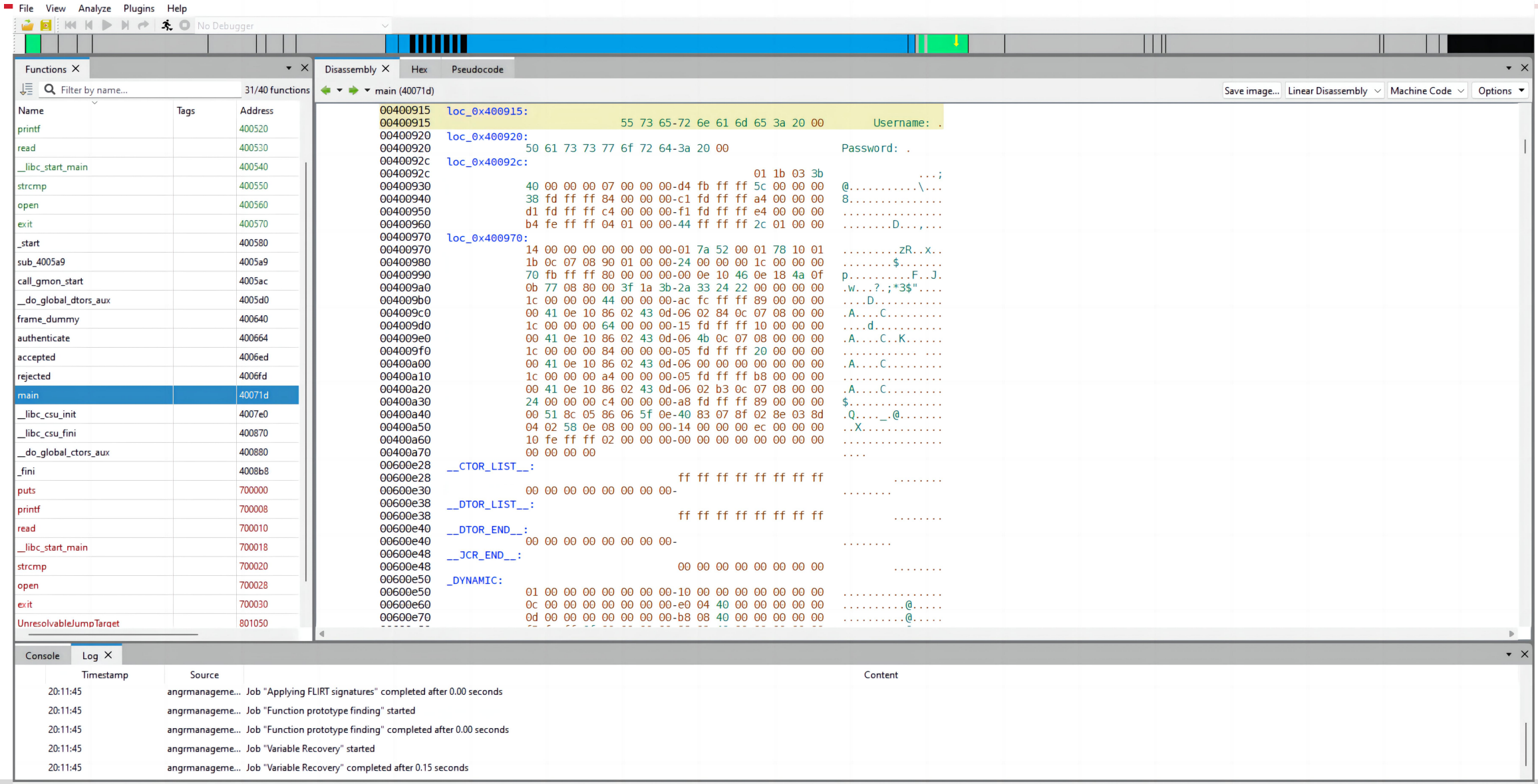
  pip install angr-management

This tool is designed to support interactive CFGs

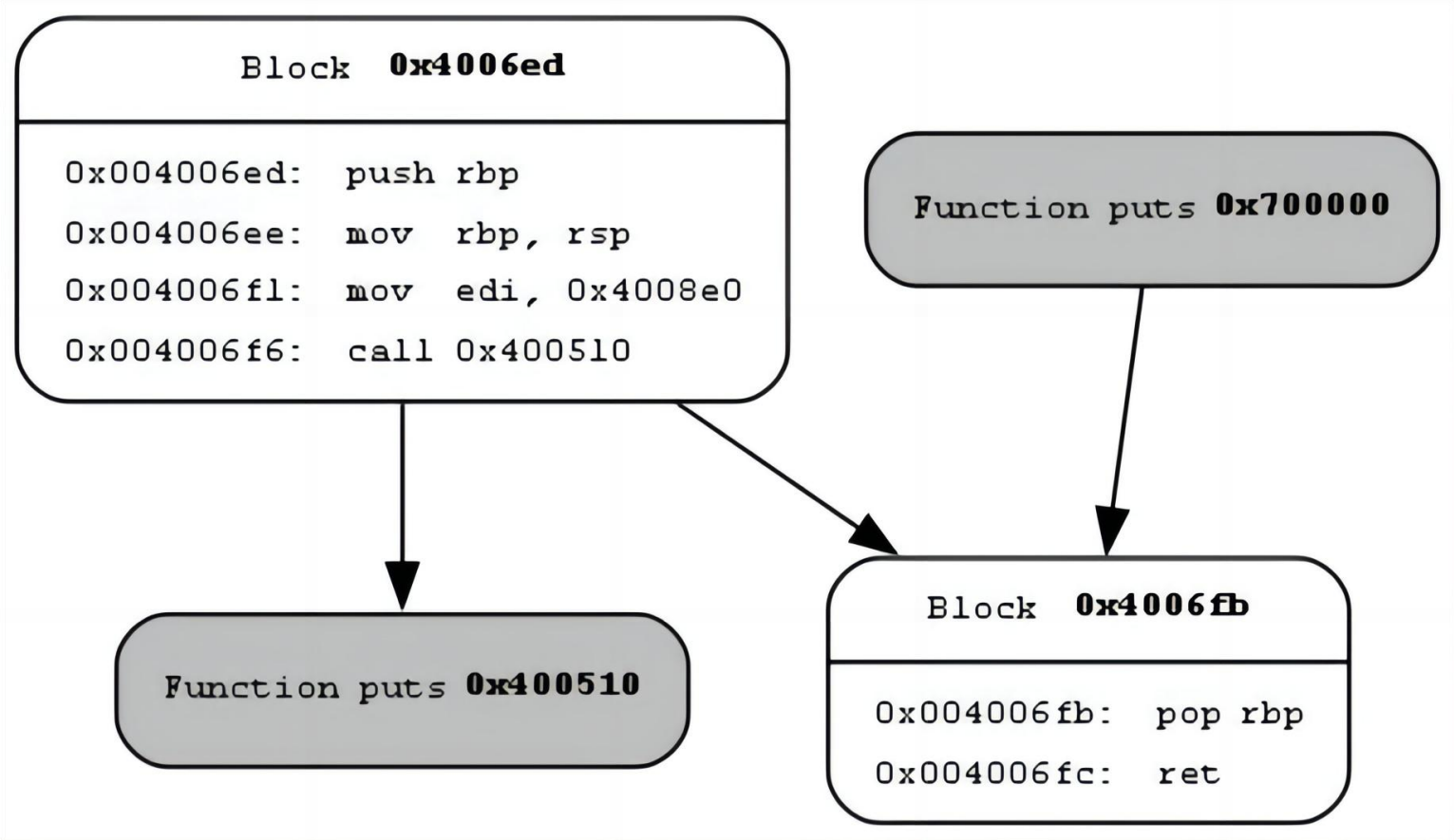
angr-management



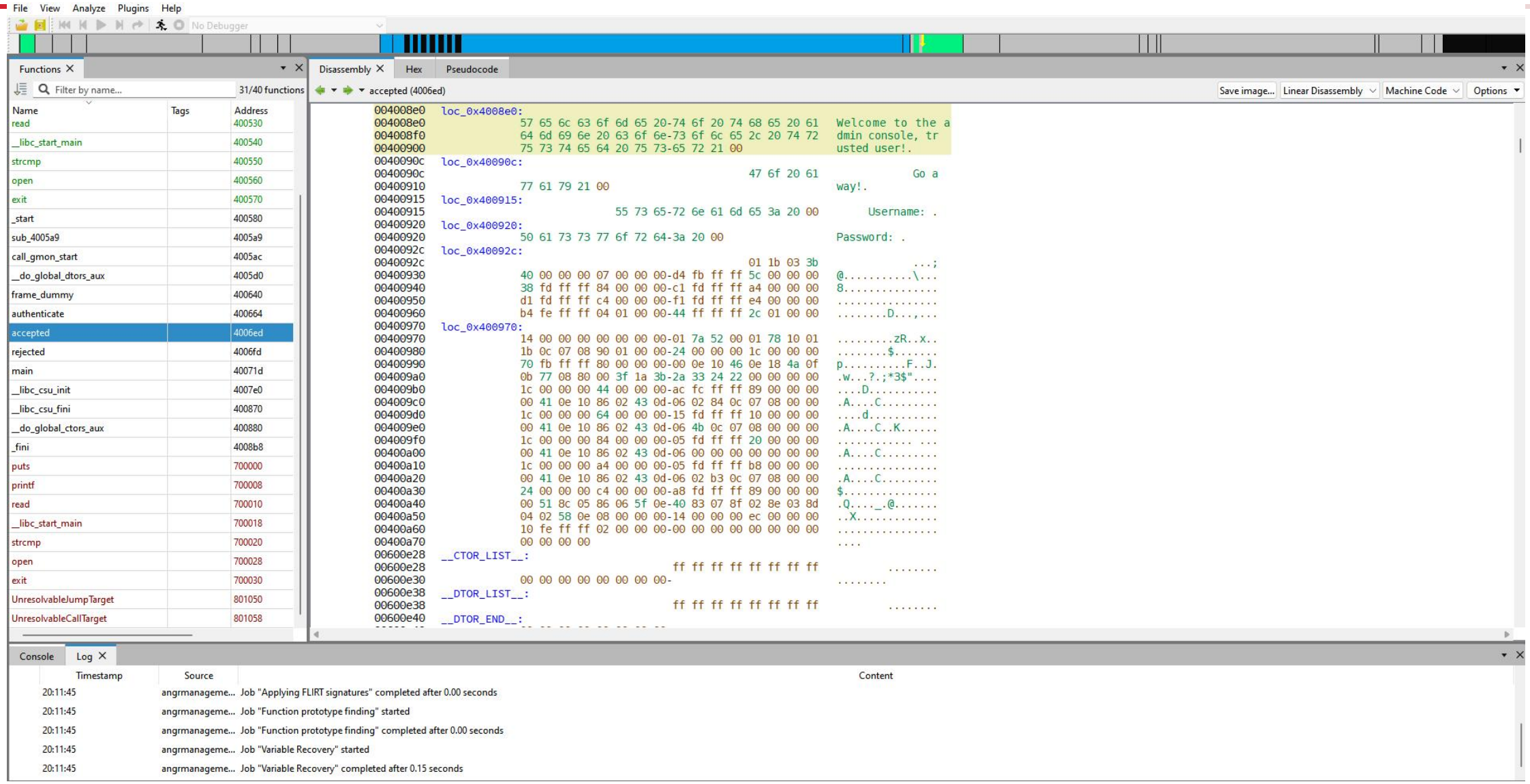
angr-management



CFG-accepted



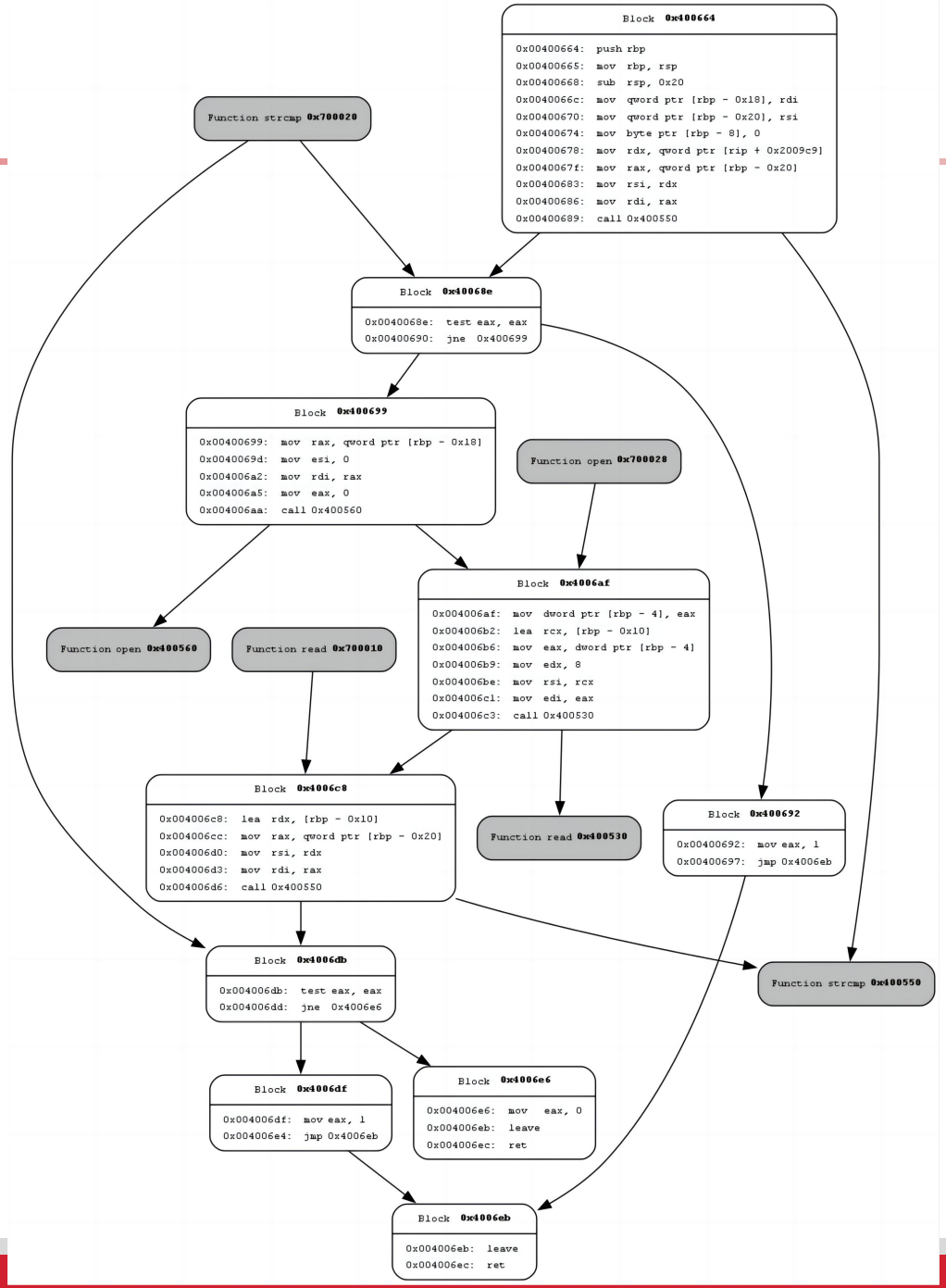
debug


File View Analyze Plugins Help
No Debugger
Functions
Filter by name...
31/40 functions
Name Tags Address
read 400530
__libc_start_main 400540
strcmp 400550
open 400560
exit 400570
_start 400580
sub_4005a9 4005a9
call_gmon_start 4005ac
__do_global_dtors_aux 4005d0
frame_dummy 400640
authenticate 400664
accepted 4006ed
rejected 4006fd
main 40071d
__libc_csu_init 4007e0
__libc_csu_fini 400870
__do_global_ctors_aux 400880
_fini 4008b8
puts 700000
printf 700008
read 700010
__libc_start_main 700018
strcmp 700020
open 700028
exit 700030
UnresolvableJumpTarget 801050
UnresolvableCallTarget 801058
Disassembly
Hex
Pseudocode
accepted (4006ed)
Save image...
Linear Disassembly
Machine Code
Options
004008e0 loc_0x4008e0:
004008e0 57 65 6c 63 6f 6d 65 20-74 6f 20 74 68 65 20 61 Welcome to the a
004008f0 64 6d 69 6e 20 63 6f 6e-73 6f 6c 65 2c 20 74 72 dmin console, tr
00400900 75 73 74 65 64 20 75 73-65 72 21 00 usted user!.
0040090c loc_0x40090c:
0040090c 47 6f 20 61 Go a
00400910 77 61 79 21 00 way!.
00400915 loc_0x400915:
00400915 55 73 65-72 6e 61 6d 65 3a 20 00 Username: .
00400920 loc_0x400920:
00400920 50 61 73 73 77 6f 72 64-3a 20 00 Password: .
0040092c loc_0x40092c:
0040092c 01 1b 03 3b ...;
00400930 40 00 00 00 07 00 00 00-d4 fb ff ff 5c 00 00 00 @.....\...
00400940 38 fd ff ff 84 00 00 00-c1 fd ff ff a4 00 00 00 8.....
00400950 d1 fd ff ff c4 00 00 00-f1 fd ff ff e4 00 00 00 .....
00400960 b4 fe ff ff 04 01 00 00-44 ff ff ff 2c 01 00 00 ....D...,...
00400970 loc_0x400970:
00400970 14 00 00 00 00 00 00 00-01 7a 52 00 01 78 10 01 .....zR..x..
00400980 1b 0c 07 08 90 01 00 00-24 00 00 00 1c 00 00 00 ....$.....
00400990 70 fb ff ff 80 00 00 00-00 0e 10 46 0e 18 4a 0f p.....F..J.
004009a0 0b 77 08 80 00 3f 1a 3b-2a 33 24 22 00 00 00 00 .w...?.;*3$"....
004009b0 1c 00 00 00 44 00 00 00-ac fc ff ff 89 00 00 00 ....D.....
004009c0 00 41 0e 10 86 02 43 0d-06 02 84 0c 07 08 00 00 .A....C.....
004009d0 1c 00 00 00 64 00 00 00-15 fd ff ff 10 00 00 00 ....d.....
004009e0 00 41 0e 10 86 02 43 0d-06 4b 0c 07 08 00 00 00 .A....C..K.....
004009f0 1c 00 00 00 84 00 00 00-05 fd ff ff 20 00 00 00 ....
00400a00 00 41 0e 10 86 02 43 0d-06 00 00 00 00 00 00 00 .A....C.....
00400a10 1c 00 00 00 a4 00 00 00-05 fd ff ff b8 00 00 00 .....
00400a20 00 41 0e 10 86 02 43 0d-06 02 b3 0c 07 08 00 00 .A....C.....
00400a30 24 00 00 00 c4 00 00 00-a8 fd ff ff 89 00 00 00 $.....
00400a40 00 51 8c 05 86 06 5f 0e-40 83 07 8f 02 8e 03 8d .Q...._.@.....
00400a50 04 02 58 0e 08 00 00 00-14 00 00 00 ec 00 00 00 ..X.....
00400a60 10 fe ff ff 02 00 00 00-00 00 00 00 00 00 00 00 .....
00400a70 00 00 00 00 ....
00600e28 __CTOR_LIST__:
00600e28 ff ff ff ff ff ff ff ff .....
00600e30 00 00 00 00 00 00 00 00- .....
00600e38 __DTOR_LIST__:
00600e38 ff ff ff ff ff ff ff ff .....
00600e40 __DTOR_END__:
Console
Log
Timestamp Source Content
20:11:45 angrmanageme... Job "Applying FLIRT signatures" completed after 0.00 seconds
20:11:45 angrmanageme... Job "Function prototype finding" started
20:11:45 angrmanageme... Job "Function prototype finding" completed after 0.00 seconds
20:11:45 angrmanageme... Job "Variable Recovery" started
20:11:45 angrmanageme... Job "Variable Recovery" completed after 0.15 seconds

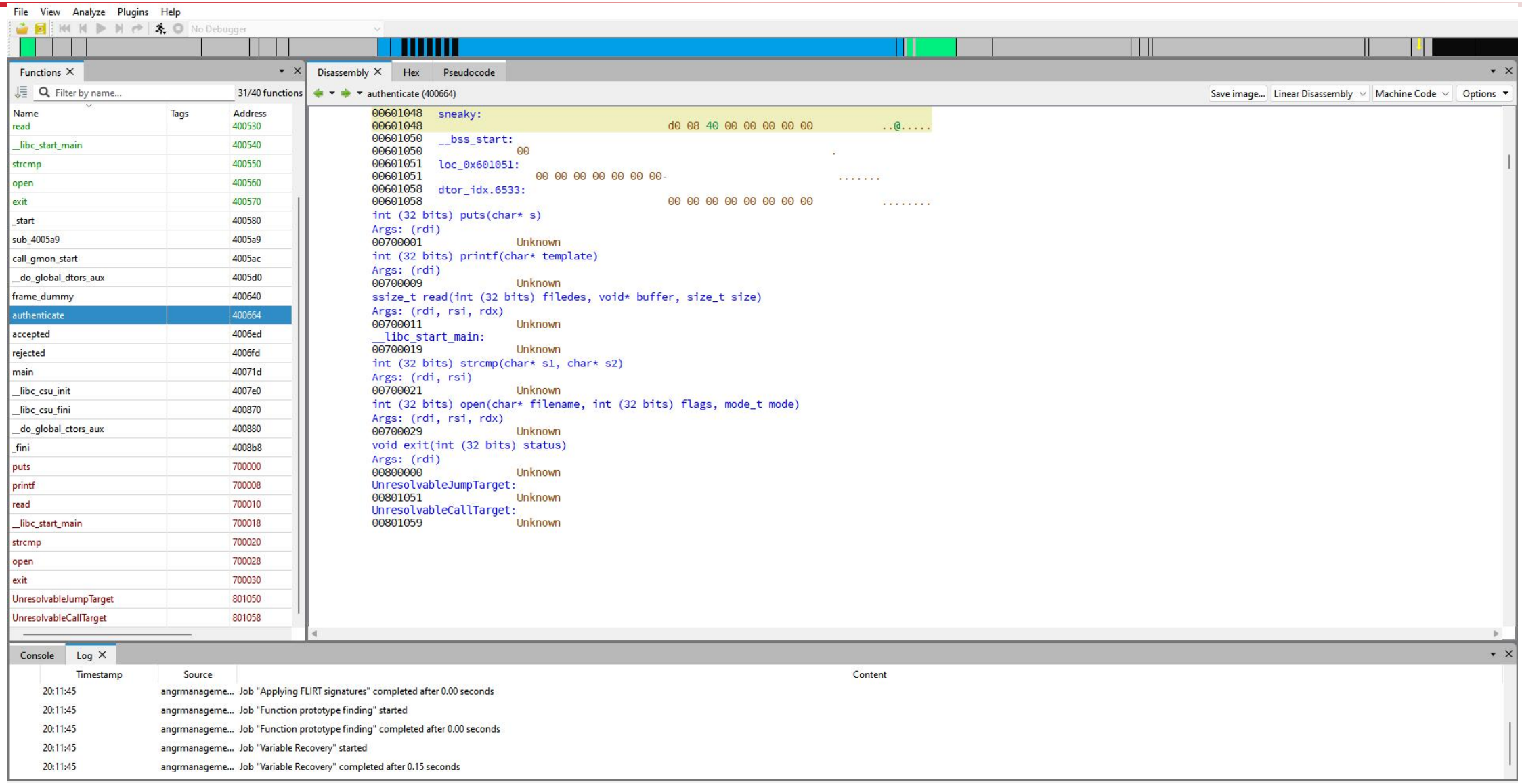
debug

```
proj = angr.Project('backdoor1', auto_load_libs=False)
simgr = proj.factory.simgr()
simgr.explore(find=lambda s: b"Welcome" in s.posix.dumps(1))
s = simgr.found[0]
print(s.posix.dumps(1))
flag = s.posix.dumps(0)
print(flag)


#output:
b'Username: \nPassword: \nWelcome to the admin console, trusted user!\n'
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00SOSNEAKY\x00'
```

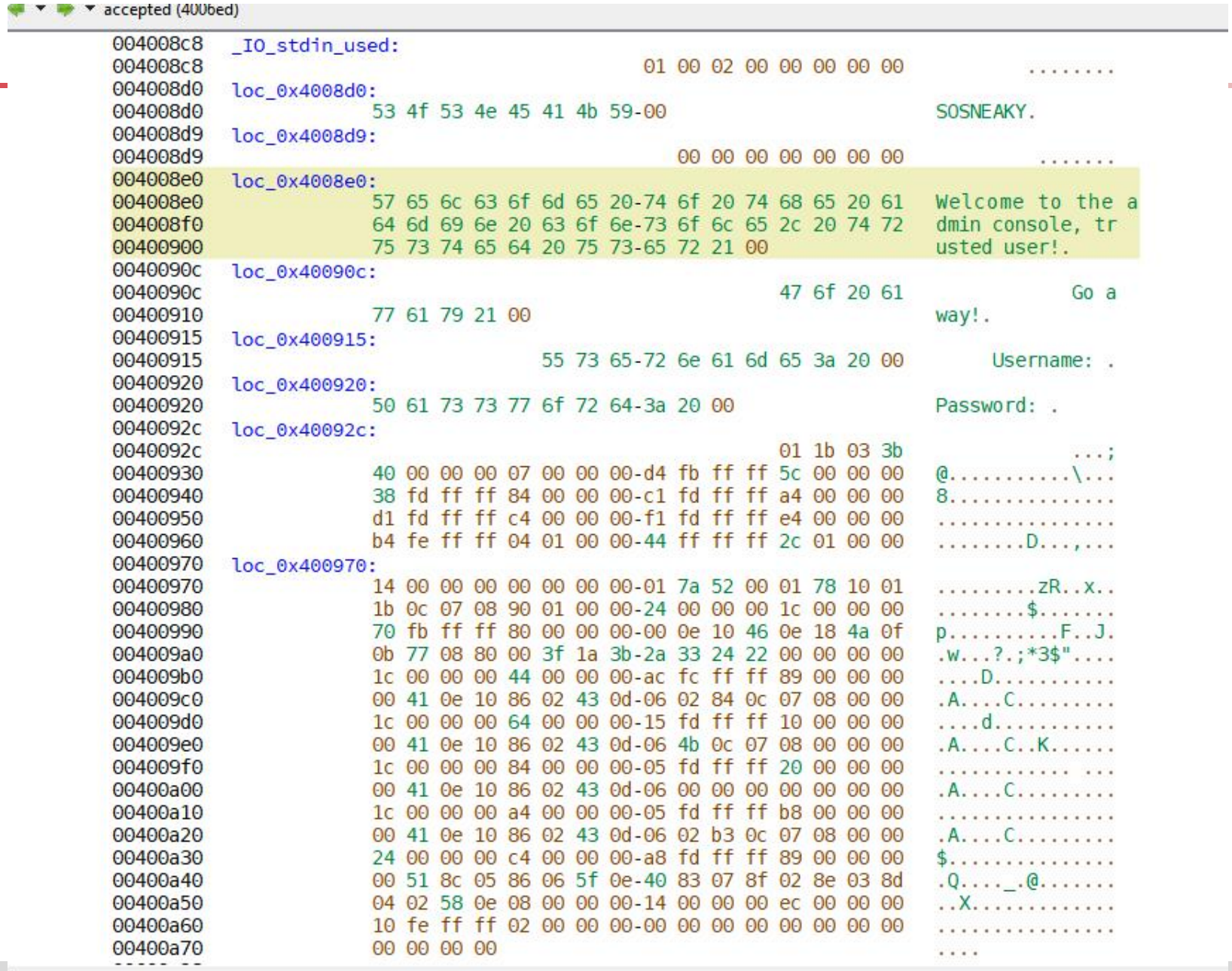
CFG-authenticate



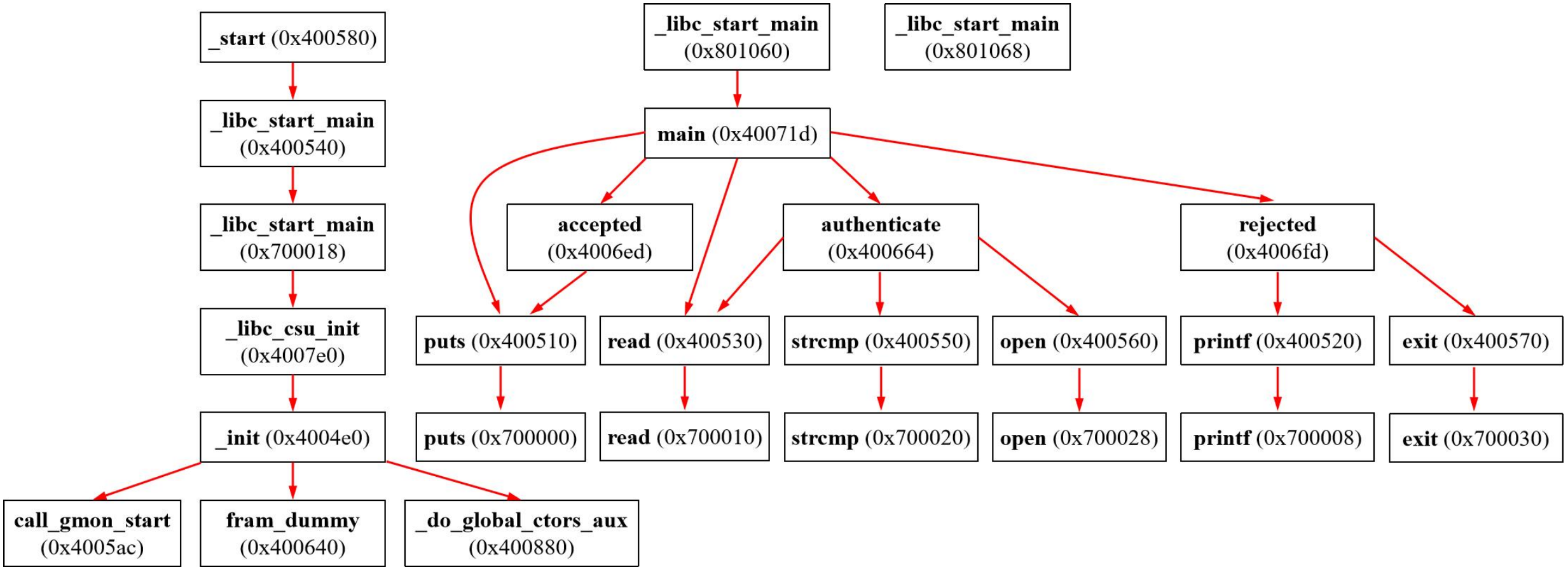
debug


File View Analyze Plugins Help
No Debugger
Functions
Filter by name...
31/40 functions
Name Tags Address
read 400530
__libc_start_main 400540
strcmp 400550
open 400560
exit 400570
_start 400580
sub_4005a9 4005a9
call_gmon_start 4005ac
__do_global_dtors_aux 4005d0
frame_dummy 400640
authenticate 400664
accepted 4006ed
rejected 4006fd
main 40071d
__libc_csu_init 4007e0
__libc_csu_fini 400870
__do_global_ctors_aux 400880
_fini 4008b8
puts 700000
printf 700008
read 700010
__libc_start_main 700018
strcmp 700020
open 700028
exit 700030
UnresolvableJumpTarget 801050
UnresolvableCallTarget 801058
Disassembly
Hex
Pseudocode
authenticate (400664)
Save image...
Linear Disassembly
Machine Code
Options
00601048 sneaky:
00601048 d0 08 40 00 00 00 00 00 ..@.....
00601050 __bss_start:
00601050 00 .
00601051 loc_0x601051:
00601051 00 00 00 00 00 00 00- ....
00601058 dtor_idx.6533:
00601058 00 00 00 00 00 00 00 00 ....
int (32 bits) puts(char* s)
Args: (rdi)
00700001 Unknown
int (32 bits) printf(char* template)
Args: (rdi)
00700009 Unknown
ssize_t read(int (32 bits) filedes, void* buffer, size_t size)
Args: (rdi, rsi, rdx)
00700011 Unknown
__libc_start_main:
00700019 Unknown
int (32 bits) strcmp(char* s1, char* s2)
Args: (rdi, rsi)
00700021 Unknown
int (32 bits) open(char* filename, int (32 bits) flags, mode_t mode)
Args: (rdi, rsi, rdx)
00700029 Unknown
void exit(int (32 bits) status)
Args: (rdi)
00800000 Unknown
UnresolvableJumpTarget:
00801051 Unknown
UnresolvableCallTarget:
00801059 Unknown
Console
Log
Timestamp Source Content
20:11:45 angrmanageme... Job "Applying FLIRT signatures" completed after 0.00 seconds
20:11:45 angrmanageme... Job "Function prototype finding" started
20:11:45 angrmanageme... Job "Function prototype finding" completed after 0.00 seconds
20:11:45 angrmanageme... Job "Variable Recovery" started
20:11:45 angrmanageme... Job "Variable Recovery" completed after 0.15 seconds

debug


accepted (400bed)
004008c8 _IO_stdin_used:
004008c8 01 00 02 00 00 00 00 00 ....
004008d0 loc_0x4008d0:
004008d0 53 4f 53 4e 45 41 4b 59-00 SOSNEAKY.
004008d9 loc_0x4008d9:
004008d9 00 00 00 00 00 00 00 ....
004008e0 loc_0x4008e0:
004008e0 57 65 6c 63 6f 6d 65 20-74 6f 20 74 68 65 20 61 Welcome to the a
004008f0 64 6d 69 6e 20 63 6f 6e-73 6f 6c 65 2c 20 74 72 dmin console, tr
00400900 75 73 74 65 64 20 75 73-65 72 21 00 usted user!.
0040090c loc_0x40090c:
0040090c 47 6f 20 61 Go a
00400910 77 61 79 21 00 way!.
00400915 loc_0x400915:
00400915 55 73 65-72 6e 61 6d 65 3a 20 00 Username: .
00400920 loc_0x400920:
00400920 50 61 73 73 77 6f 72 64-3a 20 00 Password: .
0040092c loc_0x40092c:
0040092c 01 1b 03 3b ...;
00400930 40 00 00 00 07 00 00 00-d4 fb ff ff 5c 00 00 00 @.....\...
00400940 38 fd ff ff 84 00 00 00-c1 fd ff ff a4 00 00 00 8.....
00400950 d1 fd ff ff c4 00 00 00-f1 fd ff ff e4 00 00 00 .....
00400960 b4 fe ff ff 04 01 00 00-44 ff ff ff 2c 01 00 00 .....D...,...
00400970 loc_0x400970:
00400970 14 00 00 00 00 00 00 00-01 7a 52 00 01 78 10 01 .....zR..x..
00400980 1b 0c 07 08 90 01 00 00-24 00 00 00 1c 00 00 00 .....$.....
00400990 70 fb ff ff 80 00 00 00-00 0e 10 46 0e 18 4a 0f p.....F..J.
004009a0 0b 77 08 80 00 3f 1a 3b-2a 33 24 22 00 00 00 00 .w...?.;*3$"....
004009b0 1c 00 00 00 44 00 00 00-ac fc ff ff 89 00 00 00 ....D.....
004009c0 00 41 0e 10 86 02 43 0d-06 02 84 0c 07 08 00 00 .A....C.....
004009d0 1c 00 00 00 64 00 00 00-15 fd ff ff 10 00 00 00 ....d.....
004009e0 00 41 0e 10 86 02 43 0d-06 4b 0c 07 08 00 00 00 .A....C..K.....
004009f0 1c 00 00 00 84 00 00 00-05 fd ff ff 20 00 00 00 ....
00400a00 00 41 0e 10 86 02 43 0d-06 00 00 00 00 00 00 00 .A....C.....
00400a10 1c 00 00 00 a4 00 00 00-05 fd ff ff b8 00 00 00 .....
00400a20 00 41 0e 10 86 02 43 0d-06 02 b3 0c 07 08 00 00 .A....C.....
00400a30 24 00 00 00 c4 00 00 00-a8 fd ff ff 89 00 00 00 $.....
00400a40 00 51 8c 05 86 06 5f 0e-40 83 07 8f 02 8e 03 8d .Q...._.@.....
00400a50 04 02 58 0e 08 00 00 00-14 00 00 00 ec 00 00 00 ..X.....
00400a60 10 fe ff ff 02 00 00 00-00 00 00 00 00 00 00 00 .....
00400a70 00 00 00 00 ....

CG


_start (0x400580)
_libc_start_main (0x400540)
_libc_start_main (0x700018)
_libc_csu_init (0x4007e0)
_init (0x4004e0)
call_gmon_start (0x4005ac)
fram_dummy (0x400640)
_do_global_ctors_aux (0x400880)
_libc_start_main (0x801060)
_libc_start_main (0x801068)
main (0x40071d)
accepted (0x4006ed)
authenticate (0x400664)
rejected (0x4006fd)
puts (0x400510)
read (0x400530)
strcmp (0x400550)
open (0x400560)
printf (0x400520)
exit (0x400570)
puts (0x700000)
read (0x700010)
strcmp (0x700020)
open (0x700028)
printf (0x700008)
exit (0x700030)

Alternative

```
proj = angr.Project('backdoor1', auto_load_libs=False)
state = proj.factory.entry_state(stdin=angr.SimFile)
while True:
    succ = state.step()
    if len(succ.successors)==2:
        break
    state = succ.successors[0]

state1, state2 = succ.successors
input_data = state1.posix.stdin.load(0, state1.posix.stdin.size)
print(state1.solver.eval(input_data, cast_to=bytes))
#output:
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00SOSNEAKY\x00'
```

THE END

Fangtian Zhong

CSCI 591

Gianforte School of Computing
Norm Asbjornson College of Engineering
E-mail: fangtian.zhong@montana.edu

10/30/2025