SMT in reverse engineering, for dummies

Carl Svensson

September 4, 2016

SEC-T 2016

About me

- Carl Svensson, 25
- MSc in Computer Science, KTH
- IT Security consultant, Bitsec AB
- CTF-player, HackingForSoju
- ✉ calle.svensson@zeta-two.com
- 🐦 @zetatwo
- 🌐 https://zeta-two.com

Reverse engineering in 15 seconds?

- Take stuff, e.g. software, apart
- Understand how it works
- Many possible goals
    - How can I reach a specific state?

What is SMT?

- Satisfiability modulo theories, SMT
- A bunch of variables
- A bunch of theories
  - Theory = A bunch of rules
- A bunch of formulas
- Can we find values for all values s.t. all formulas are satisifed?

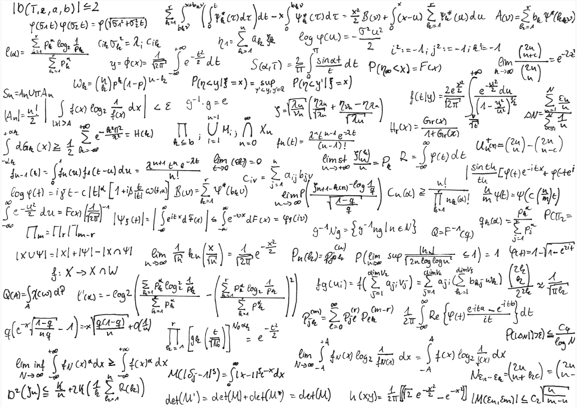
SMT: Example 1

$$x + 13 = 37$$

SMT: Example 2

$$x + y + 13 = 37 - z$$
$$x - 2 \cdot y + 10 = 10 \cdot z$$
$$4 \cdot x - z + 13 = 37 + y$$

SMT: Example 3



Microsoft to the rescue

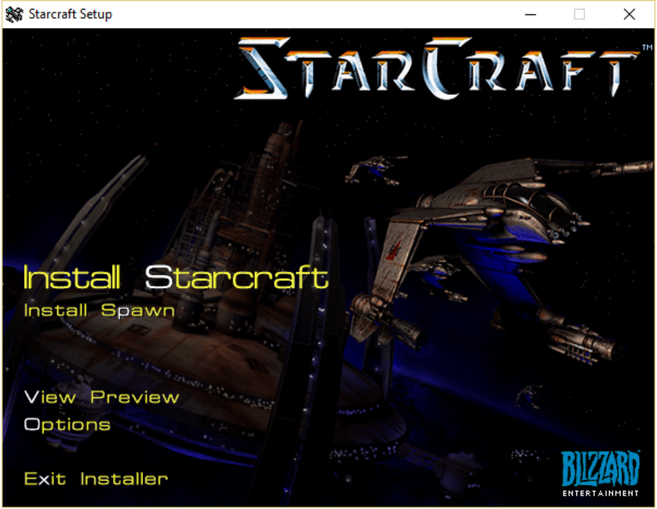
- Can we automate? Yes!
- Microsoft Research
- Z3 Theorem Prover
  - General purpose
  - Own language
  - Bindings for several languages
  - Open source & cross platform

Throwback Thursday: Starcraft

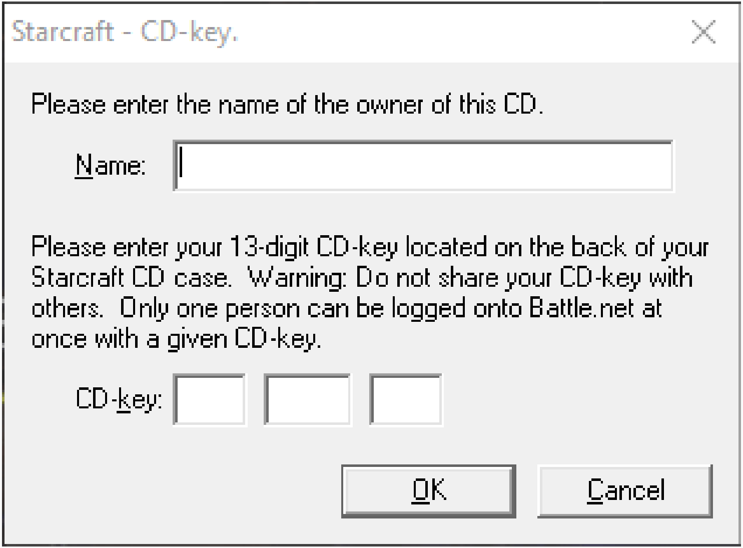
Throwback Thursday: Starcraft

- Commercial software
- Released in 1998
  - Simple protections
  - Good starting point
- Requires a serial key
- Can we create our own?

Getting to the core: Installer



Getting to the core: Serial key input



Starcraft - CD-key.

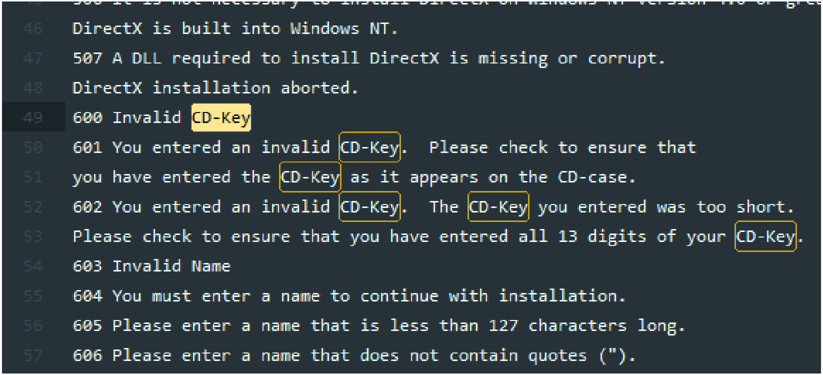
Please enter the name of the owner of this CD.

Name:

Please enter your 13-digit CD-key located on the back of your Starcraft CD case. Warning: Do not share your CD-key with others. Only one person can be logged onto Battle.net at once with a given CD-key.

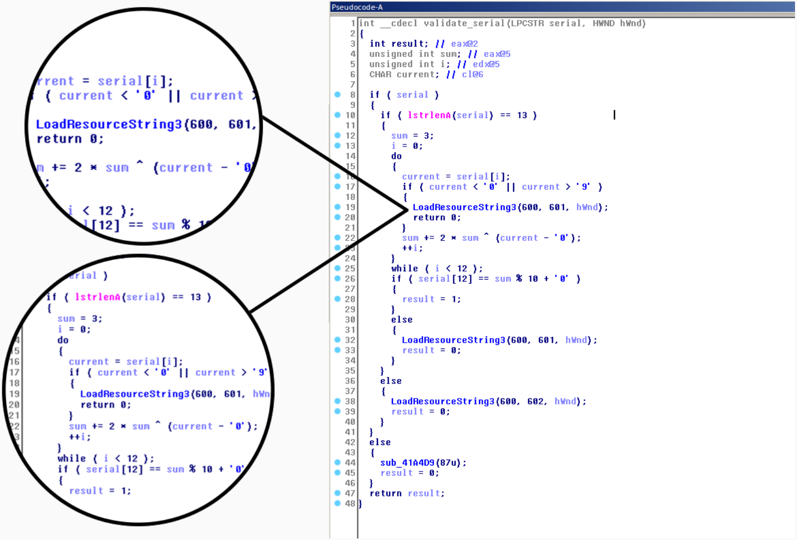
CD-key:

Getting to the core: Resource strings



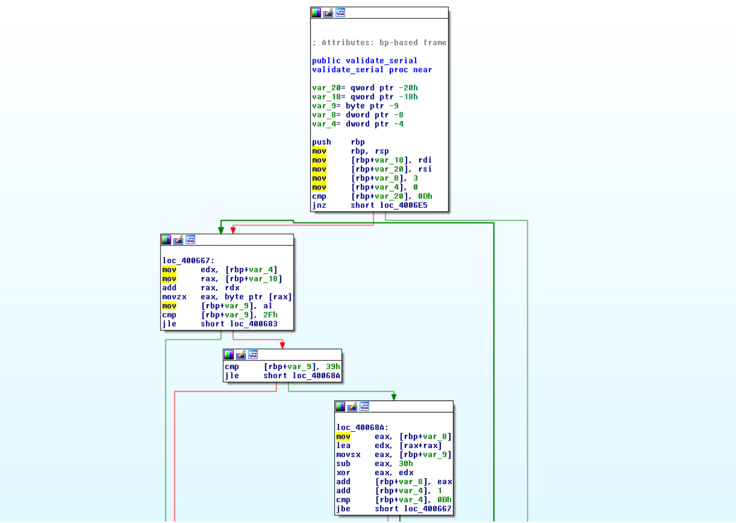
```
506 It is not necessary to install DirectX on Windows NT version 4.0 or grea
DirectX is built into Windows NT.
507 A DLL required to install DirectX is missing or corrupt.
DirectX installation aborted.
600 Invalid CD-Key
601 You entered an invalid CD-Key.  Please check to ensure that
you have entered the CD-Key as it appears on the CD-case.
602 You entered an invalid CD-Key.  The CD-Key you entered was too short.
Please check to ensure that you have entered all 13 digits of your CD-Key.
603 Invalid Name
604 You must enter a name to continue with installation.
605 Please enter a name that is less than 127 characters long.
606 Please enter a name that does not contain quotes (").
```

Getting to the core: Decompilation



```
int __cdecl validate_serial(LPCSTR serial, HWND hWnd)
{
  int result; // eax@2
  unsigned int sum; // eax@5
  unsigned int i; // edx@5
  CHAR current; // cl@6

  if ( serial )
  {
    if ( lstrlenA(serial) == 13 )
    {
      sum = 3;
      i = 0;
      do
      {
        current = serial[i];
        if ( current < '0' || current > '9' )
        {
          LoadResourceString3(600, 601, hWnd);
          return 0;
        }
        sum += 2 * sum ^ (current - '0');
        ++i;
      }
      while ( i < 12 );
      if ( serial[12] == sum % 10 + '0' )
      {
        result = 1;
      }
      else
      {
        LoadResourceString3(600, 601, hWnd);
        result = 0;
      }
    }
    else
    {
      LoadResourceString3(600, 602, hWnd);
      result = 0;
    }
  }
  else
  {
    sub_41A4D9(87u);
    result = 0;
  }
  return result;
}
```

Getting to the core: Call graph



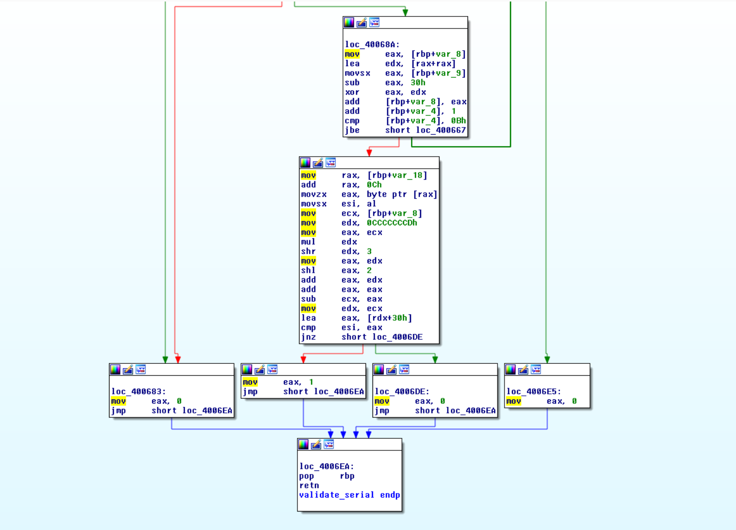
```
; Attributes: bp-based frame

public validate_serial
validate_serial proc near

var_20= qword ptr -20h
var_18= qword ptr -18h
var_9= byte ptr -9
var_8= dword ptr -8
var_4= dword ptr -4

push    rbp
mov     rbp, rsp
mov     [rbp+var_18], rdi
mov     [rbp+var_20], rsi
mov     [rbp+var_8], 3
mov     [rbp+var_4], 0
cmp     [rbp+var_20], 0Dh
jnz     short loc_4006E5
```

```
loc_400667:
mov     edx, [rbp+var_4]
mov     rax, [rbp+var_18]
add     rax, rdx
movzx   eax, byte ptr [rax]
mov     [rbp+var_9], al
cmp     [rbp+var_9], 2Fh
jle     short loc_400683
```

```
cmp     [rbp+var_9], 39h
jle     short loc_40068A
```

```
loc_40068A:
mov     eax, [rbp+var_8]
lea     edx, [rax+rax]
movsx   eax, [rbp+var_9]
sub     eax, 30h
xor     eax, edx
add     [rbp+var_8], eax
add     [rbp+var_4], 1
cmp     [rbp+var_4], 0Bh
jbe     short loc_400667
```

Getting to the core: Call graph



```
loc_40068A:
mov     eax, [rbp+var_8]
lea     edx, [rax+rax]
movsx   eax, [rbp+var_9]
sub     eax, 30h
xor     eax, edx
add     [rbp+var_8], eax
add     [rbp+var_4], 1
cmp     [rbp+var_4], 0Bh
jbe     short loc_400667
```

```
mov     rax, [rbp+var_18]
add     rax, 0Ch
movzx   eax, byte ptr [rax]
movsx   esi, al
mov     ecx, [rbp+var_8]
mov     edx, 0CCCCCCCDh
mov     eax, ecx
mul     edx
shr     edx, 3
mov     eax, edx
shl     eax, 2
add     eax, edx
add     eax, eax
sub     ecx, eax
mov     edx, ecx
lea     eax, [rdx+30h]
cmp     esi, eax
jnz     short loc_4006DE
```

```
loc_400683:
mov     eax, 0
jmp     short loc_4006EA
```

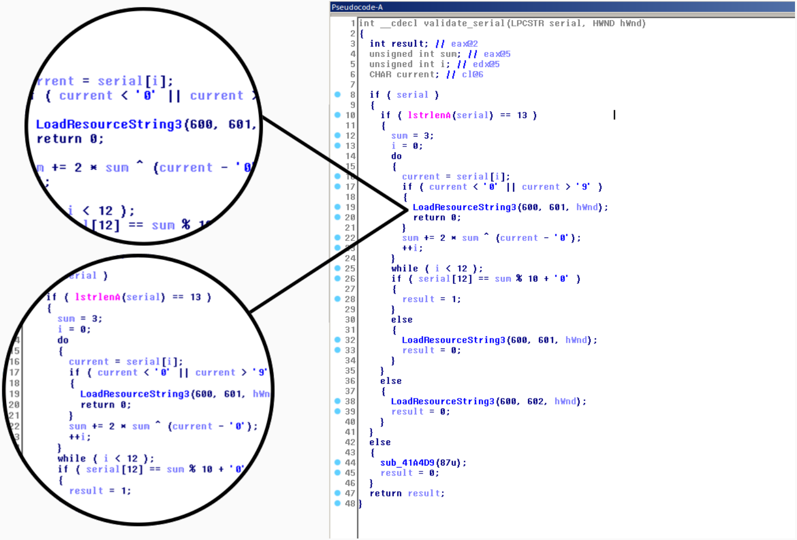
```
mov     eax, 1
jmp     short loc_4006EA
```

```
loc_4006DE:
mov     eax, 0
jmp     short loc_4006EA
```

```
loc_4006E5:
mov     eax, 0
```

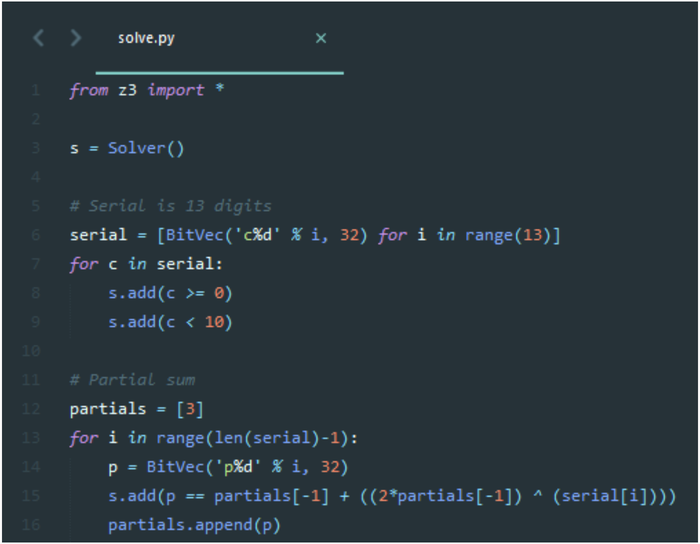
```
loc_4006EA:
pop     rbp
retn
validate_serial endp
```

Getting to the core: Decompilation



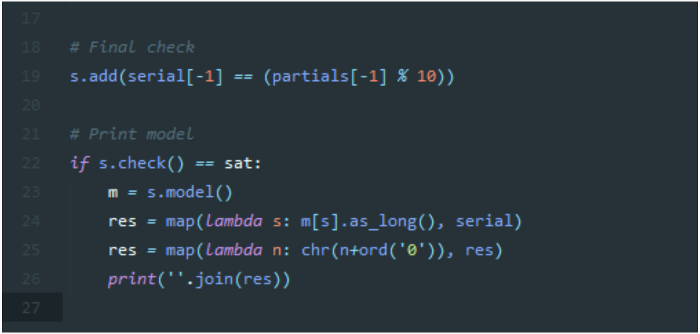
```
int __cdecl validate_serial(LPCSTR serial, HWND hWnd)
{
  int result; // eax@2
  unsigned int sum; // eax@5
  unsigned int i; // edx@5
  CHAR current; // cl@6

  if ( serial )
  {
    if ( lstrlenA(serial) == 13 )
    {
      sum = 3;
      i = 0;
      do
      {
        current = serial[i];
        if ( current < '0' || current > '9' )
        {
          LoadResourceString3(600, 601, hWnd);
          return 0;
        }
        sum += 2 * sum ^ (current - '0');
        ++i;
      }
      while ( i < 12 );
      if ( serial[12] == sum % 10 + '0' )
      {
        result = 1;
      }
      else
      {
        LoadResourceString3(600, 601, hWnd);
        result = 0;
      }
    }
    else
    {
      LoadResourceString3(600, 602, hWnd);
      result = 0;
    }
  }
  else
  {
    sub_41A4D9(87u);
    result = 0;
  }
  return result;
}
```

Z3: Formulating formulas



```python
from z3 import *

s = Solver()

# Serial is 13 digits
serial = [BitVec('c%d' % i, 32) for i in range(13)]
for c in serial:
    s.add(c >= 0)
    s.add(c < 10)

# Partial sum
partials = [3]
for i in range(len(serial)-1):
    p = BitVec('p%d' % i, 32)
    s.add(p == partials[-1] + ((2*partials[-1]) ^ (serial[i])))
    partials.append(p)
```

Z3: Formulating formulas

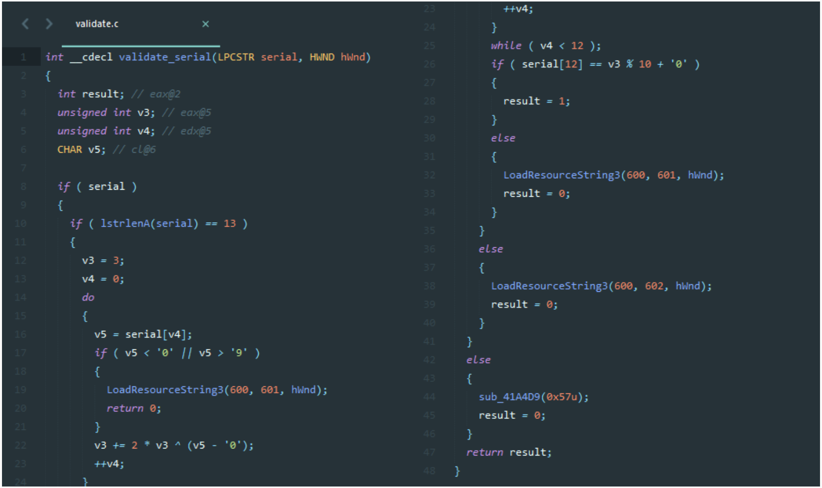


```
# Final check
s.add(serial[-1] == (partials[-1] % 10))

# Print model
if s.check() == sat:
    m = s.model()
    res = map(lambda s: m[s].as_long(), serial)
    res = map(lambda n: chr(n+ord('0')), res)
    print(''.join(res))
```

Once again, with fee... angr

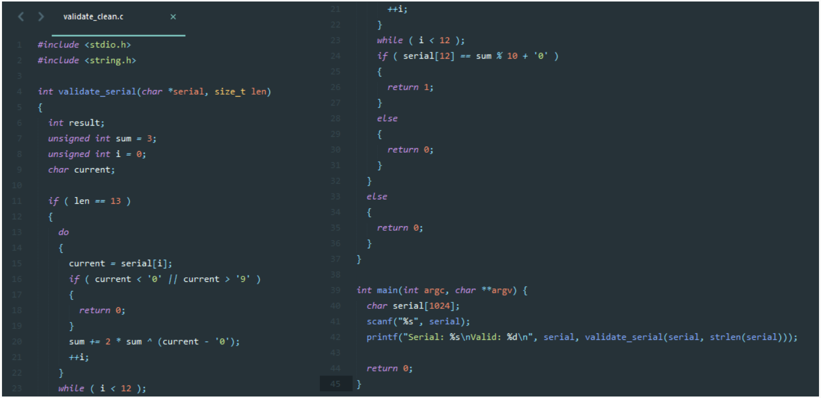
- ”python framework for analyzing binaries”
- ”both static and dynamic symbolic (concolic)”
- Computer Security Lab at UC Santa Barbara
- Uses Z3 internally

Angr management: Extracting the code



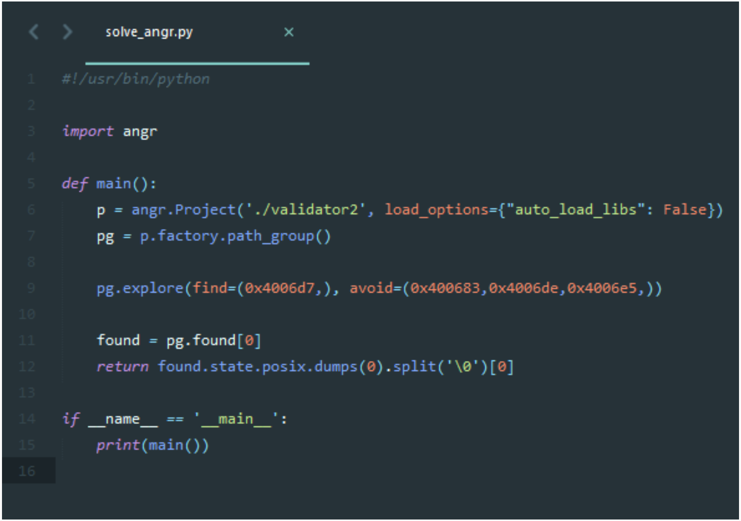
```c
int __cdecl validate_serial(LPCSTR serial, HWND hWnd)
{
  int result; // eax@2
  unsigned int v3; // eax@5
  unsigned int v4; // edx@5
  CHAR v5; // cl@6

  if ( serial )
  {
    if ( lstrlenA(serial) == 13 )
    {
      v3 = 3;
      v4 = 0;
      do
      {
        v5 = serial[v4];
        if ( v5 < '0' || v5 > '9' )
        {
          LoadResourceString3(600, 601, hWnd);
          return 0;
        }
        v3 += 2 * v3 ^ (v5 - '0');
        ++v4;
      }
      while ( v4 < 12 );
      if ( serial[12] == v3 % 10 + '0' )
      {
        result = 1;
      }
      else
      {
        LoadResourceString3(600, 601, hWnd);
        result = 0;
      }
    }
    else
    {
      LoadResourceString3(600, 602, hWnd);
      result = 0;
    }
  }
  else
  {
    sub_41A4D9(0x57u);
    result = 0;
  }
  return result;
}
```

ANGR MANAGEMENT: MINIMIZING THE CODE



```c
#include <stdio.h>
#include <string.h>

int validate_serial(char *serial, size_t len)
{
  int result;
  unsigned int sum = 3;
  unsigned int i = 0;
  char current;

  if ( len == 13 )
  {
    do
    {
      current = serial[i];
      if ( current < '0' || current > '9' )
      {
        return 0;
      }
      sum += 2 * sum ^ (current - '0');
      ++i;
    }
    while ( i < 12 );
    if ( serial[12] == sum % 10 + '0' )
    {
      return 1;
    }
    else
    {
      return 0;
    }
  }
  else
  {
    return 0;
  }
}

int main(int argc, char **argv) {
  char serial[1024];
  scanf("%s", serial);
  printf("Serial: %s\nValid: %d\n", serial, validate_serial(serial, strlen(serial)));

  return 0;
}
```

Angr management: Writing the explorer



```python
#!/usr/bin/python

import angr

def main():
    p = angr.Project('./validator2', load_options={"auto_load_libs": False})
    pg = p.factory.path_group()

    pg.explore(find=(0x4006d7,), avoid=(0x400683,0x4006de,0x4006e5,))

    found = pg.found[0]
    return found.state.posix.dumps(0).split('\0')[0]

if __name__ == '__main__':
    print(main())
```

Thanks for listening!