

Using SMT solvers for binary analysis and exploitation

A primer on SMT, SMT solvers, Z3 & angr

Carl Svensson

August 29, 2018

Nixucon 2018

About me

- Carl Svensson, 27
- MSc in Computer Science, KTH
- Head of Security, KRY/LIVI
- 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 satisfied?

$$x + 13 = 37$$



$$x + y + 13 = 37 - z$$

$$x - 2 \cdot y + 10 = 10 \cdot z$$

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



SMT: Example 3

$$\begin{aligned}
 & |b(\tau, z, a, b)| \leq 2 \\
 & \varphi(\bar{v}_1 t) \varphi(\bar{v}_2 t) = \varphi(\sqrt{\bar{v}_1^2 + \bar{v}_2^2} t) \\
 & \rho(\omega) = \frac{\sum_{k=1}^n \rho_k^* \log_2 \frac{1}{\rho_k}}{\sum_{k=1}^n \rho_k^*} \quad \text{if } \bar{v}_k = \lambda; \text{ c.f.} \quad \eta_1 = \sum_{k=1}^n a_k \bar{v}_k \quad \log \varphi(u) = -\frac{\sigma^2 u^2}{2} \quad i^2 = -1; j^2 = -1; i \bar{v}_k = -1 \\
 & y = \varphi(x) = \frac{1}{\sqrt{2\pi}} \int_0^{\infty} e^{-\frac{t^2}{2}} dt \quad S(\alpha, \tau) = \frac{2}{\pi} \int_0^{\pi} \frac{\sin \alpha t}{t} dt \quad P(\eta_0 < x) = F(x) \\
 & W_k = \binom{n}{k} p^k (1-p)^{n-k} \quad P(\eta < y | \xi = x) = \sup_{\gamma < y, \gamma \geq 0} P(\eta < y | \xi = x) \\
 & S_n = A_n U \pi A_n \quad |A_n| = \frac{n!}{2} \left| \int_{|x| > A} f(x) \log_2 \frac{1}{f(x)} dx \right| < \varepsilon \quad g^{-1} \cdot g = e \quad \gamma = \left[\frac{2a}{\sqrt{n}} \left(\frac{\eta_1}{\sqrt{n}} + \frac{\eta_2 - \eta_1}{\sqrt{n}} \right) \right] \quad f(t|y) = \frac{2e^{\frac{y^2}{2}}}{\sqrt{2\pi}} \int_0^{\infty} \frac{e^{-\frac{u^2}{2}} du}{(1 - \frac{y^2}{u^2})^{\frac{3}{2}}} \\
 & \int_{-\infty}^{\infty} d\mu_k(x) \geq \frac{1}{2} \sum_{n \rightarrow \infty} e^{-\frac{n^2}{2}} = H(k) \quad \prod_{k \leq b} \bigcup_{i=1}^{n-1} M_i; \bigcap_{n=0}^{\infty} X_n \quad f_n(t) = \frac{2^{-n} e^{-t} e^{-2t}}{(n-1)!} \quad H_r(x) = \frac{Gr(x)}{1 + Gr(x)} \quad U_{n-1}^+ = (2n) - (2n)_{n-1} \\
 & \int_{-u}^u f_{n-1}(t) = \int_{-u}^u f_n(u) f_1(t-u) du = \frac{2n+1}{n!} e^{-2t} \quad \lim_{t \rightarrow 0} (Gr) = 0 \quad \lim_{n \rightarrow \infty} \frac{\gamma_n}{n} = P_e \quad R = \int_{-\infty}^{\infty} \varphi(t) dt \quad U_{n-1}^+ = (2n) - (2n)_{n-1} \\
 & \log \varphi(t) = i \gamma t - c |t| \left[1 + i \beta \frac{t}{|t|} \omega(t, n) \right] \quad \beta(v) = \sum_{k=1}^n \varphi^*(b_k v) \quad \lim_{n \rightarrow \infty} P \left(\frac{\sum_{j=1}^n (a_j - \bar{a}_j) \log \frac{1}{q}}{\sqrt{\frac{1-q}{q}}} \right) C_n(\alpha) \geq \frac{n!}{\prod_{k=1}^n n_k(\omega)!} \quad \frac{1}{m} \varphi(t) = \varphi\left(c \left(\frac{n}{m}\right) t\right) \\
 & \int_{-\infty}^{\infty} e^{-\frac{u^2}{2}} du = F(x) \left(\frac{1}{\sqrt{2\pi}} \right)^{-1} \quad |\Psi_S(t)| = \left| \int_{-\infty}^{\infty} e^{itx} dF(x) \right| \leq \int_{-\infty}^{\infty} e^{-ux} dF(x) = \varphi_S(u) \quad g^{-1} N_g = \{g^{-1} n_g | n \in N\} \quad Q = F^{-1}(c_p) \quad q_n(\alpha) = \sum_{j=1}^n P_j^* \quad P(\Pi_2 = \\
 & \prod_m = \prod_r \prod_{m-r} \quad \lim_{n \rightarrow \infty} \frac{1}{n} \ln \left(\frac{x}{1/n} \right) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} \quad P_n(b_k) = \frac{C_n}{P_n} \quad P \left(\lim_{n \rightarrow \infty} \sup \frac{\ln n}{2n \log \log n} \leq 1 \right) = 1 \quad (q, H) = 1 - \sqrt{1 - e^{2H}} \\
 & f: X \rightarrow X \cap W \quad Q(A) = \int_A \chi(x) dP \quad \ell'(x) = -\log_2 \left(\frac{\sum_{k=1}^n \rho_k^* \log_2 \frac{1}{\rho_k}}{\sum_{k=1}^n \rho_k^*} - \left(\frac{\sum_{k=1}^n \rho_k^* \log_2 \frac{1}{\rho_k}}{\sum_{k=1}^n \rho_k^*} \right)^2 \right) \quad f(g(u_i)) = f \left(\sum_{j=1}^{\dim V_k} a_{ji} v_j \right) = \sum_{j=1}^{\dim V_k} a_{ji} \sum_{k=1}^{\dim V_k} b_{kj} w_k \quad \frac{(2\ell)}{2\ell_k} \approx \frac{1}{\sqrt{1/\ell_k}} \\
 & q \left(e^{-x} \sqrt{\frac{1-q}{nq}} - 1 \right) = x \sqrt{\frac{q(1-q)}{n}} + o\left(\frac{1}{\sqrt{n}}\right) \quad \prod_{k=1}^r \left[g_k \left(\frac{t}{\sqrt{1/\ell_k}} \right) \right]^{N_k \alpha_k} = e^{-\frac{t^2}{2}} \quad P_{j,k}^{(m)} = \sum_{c=0}^m P_j^{(m)} P_{k,c}^{(m-r)} \quad \frac{1}{2\pi} \int_{-\infty}^{\infty} \operatorname{Re} \left\{ \varphi(t) \frac{e^{it\alpha} - e^{-it\beta}}{it} \right\} dt \quad P(|\omega|) \leq \frac{C_q}{\log N} \\
 & \liminf_{N \rightarrow \infty} \int_{-\infty}^{\infty} f_N(x)^\alpha dx \geq \int_{-\infty}^{\infty} f(x)^\alpha dx \quad \lim_{N \rightarrow \infty} \int_{-1}^1 f_N(x) \log_2 \frac{1}{f_N(x)} dx = \int_{-1}^1 f(x) \log_2 \frac{1}{f(x)} dx \quad P(\omega, \nu) \leq \frac{C_q}{\log N} \\
 & D^2(J_n) \leq \frac{K}{n} + 2K \left(\frac{1}{2} \sum_{k=1}^n R(\ell_k) \right) \quad \det(M') = \det(M) + \det(M'') = \det(M) \quad h(xy) = \frac{1}{2\pi} \left[\sqrt{2} e^{-\frac{x^2}{2}} - e^{-x^2} \right] \quad |M(\varepsilon_n, \varepsilon_m)| \leq C_2 \sqrt{\frac{n}{m-n}} \\
 & \det(M') = \det(M) + \det(M'') = \det(M) \quad h(xy) = \frac{1}{2\pi} \left[\sqrt{2} e^{-\frac{x^2}{2}} - e^{-x^2} \right] \quad |M(\varepsilon_n, \varepsilon_m)| \leq C_2 \sqrt{\frac{n}{m-n}}
 \end{aligned}$$



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



Using Z3 in Python

```
1  from z3 import *
2
3  x = Real('x')
4  y = Real('y')
5
6  s = Solver()
7  s.add(x + y > 5)
8  s.add(x > 1, y > 1)
9
10 print(s.check())
11 print(s.model())
12
13 """
14 > python z3_example.py
15 sat
16 [y = 3/2, x = 4]
17 """
18
```

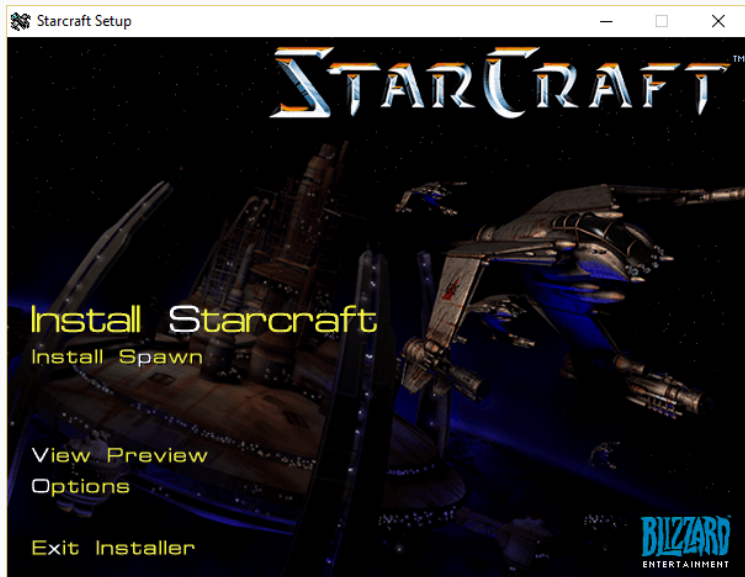
Throwback Thursday: Starcraft

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

```
45 506 It is not necessary to install DirectX on Windows NT version 4.0 or greater.  
46 DirectX is built into Windows NT.  
47 507 A DLL required to install DirectX is missing or corrupt.  
48 DirectX installation aborted.  
49 600 Invalid CD-Key  
50 601 You entered an invalid CD-Key. Please check to ensure that  
51 you have entered the CD-Key as it appears on the CD-case.  
52 602 You entered an invalid CD-Key. The CD-Key you entered was too short.  
53 Please check to ensure that you have entered all 13 digits of your CD-Key.  
54 603 Invalid Name  
55 604 You must enter a name to continue with installation.  
56 605 Please enter a name that is less than 127 characters long.  
57 606 Please enter a name that does not contain quotes (").
```

Getting to the core: Decompilation

```
current = serial[i];  
( current < '0' || current > '9' )
```

```
LoadResourceString3(600, 601,  
return 0;
```

```
sum += 2 * sum ^ (current - '0'  
;
```

```
i < 12 );  
serial[12] == sum % 10 + '0'
```

```
serial )
```

```
if ( !strlenA(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'
```

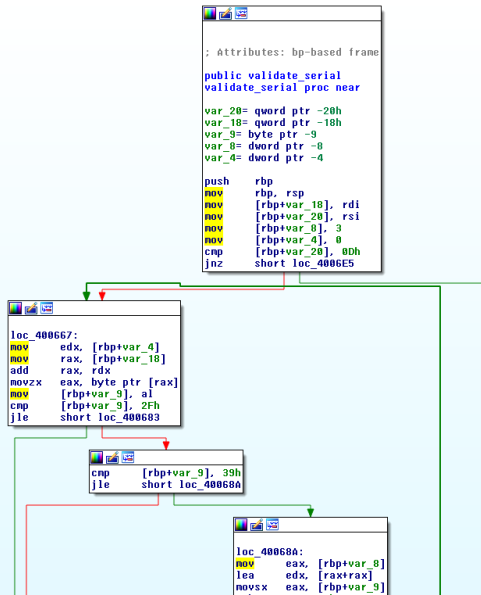
```
{
```

```
result = 1;
```

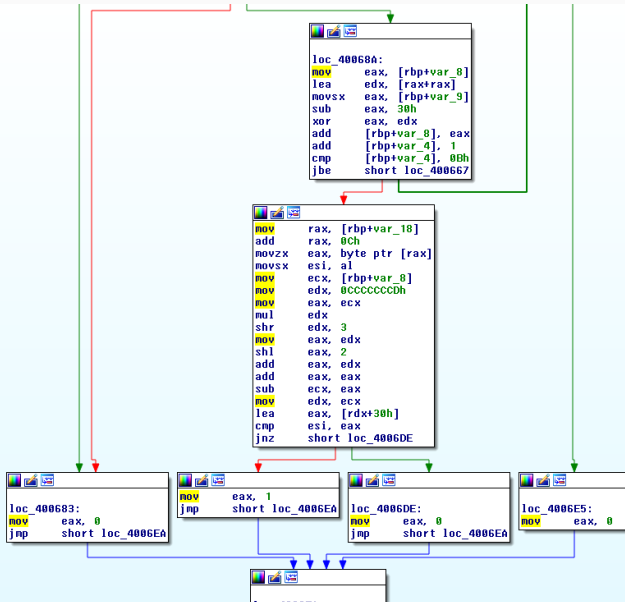
Pseudocode-A

```
1 int __cdecl validate_serial(LPCSTR serial, HWND hWnd)  
2 {  
3     int result; // eax@2  
4     unsigned int sum; // eax@5  
5     unsigned int i; // edx@5  
6     CHAR current; // cl@6  
7  
8     if ( serial )  
9     {  
10        if ( !strlenA(serial) == 13 )  
11        {  
12            sum = 3;  
13            i = 0;  
14            do  
15            {  
16                current = serial[i];  
17                if ( current < '0' || current > '9' )  
18                {  
19                    LoadResourceString3(600, 601, hWnd);  
20                    return 0;  
21                }  
22                sum += 2 * sum ^ (current - '0');  
23                ++i;  
24            }  
25            while ( i < 12 );  
26            if ( serial[12] == sum % 10 + '0' )  
27            {  
28                result = 1;  
29            }  
30            else  
31            {  
32                LoadResourceString3(600, 601, hWnd);  
33                result = 0;  
34            }  
35        }  
36        else  
37        {  
38            LoadResourceString3(600, 602, hWnd);  
39            result = 0;  
40        }  
41        else  
42        {  
43            sub_41A4D9(87u);  
44            result = 0;  
45        }  
46    }  
47    return result;  
48 }
```

Getting to the core: Call graph



Getting to the core: Call graph



Getting to the core: Decompilation

```
current = serial[i];  
( current < '0' || current > '9' )
```

```
LoadResourceString3(600, 601,  
return 0;
```

```
sum += 2 * sum ^ (current - '0'  
;
```

```
while ( i < 12 );  
if ( serial[12] == sum % 10 + '0' )
```

```
return serial )
```

```
if ( !strlenA(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;
```

Pseudocode-A

```
1 int __cdecl validate_serial(LPCSTR serial, HWND hWnd)  
2 {  
3     int result; // eax@2  
4     unsigned int sum; // eax@5  
5     unsigned int i; // edx@5  
6     CHAR current; // cl@6  
7  
8     if ( serial )  
9     {  
10        if ( !strlenA(serial) == 13 )  
11        {  
12            sum = 3;  
13            i = 0;  
14            do  
15            {  
16                current = serial[i];  
17                if ( current < '0' || current > '9' )  
18                {  
19                    LoadResourceString3(600, 601, hWnd);  
20                    return 0;  
21                }  
22                sum += 2 * sum ^ (current - '0');  
23                ++i;  
24            }  
25            while ( i < 12 );  
26            if ( serial[12] == sum % 10 + '0' )  
27            {  
28                result = 1;  
29            }  
30            else  
31            {  
32                LoadResourceString3(600, 601, hWnd);  
33                result = 0;  
34            }  
35        }  
36        else  
37        {  
38            LoadResourceString3(600, 602, hWnd);  
39            result = 0;  
40        }  
41        else  
42        {  
43            sub_41A4D9(87u);  
44            result = 0;  
45        }  
46    }  
47    return result;  
48 }
```

Z3: Formulating formulas

```
< > solve.py x
1  from z3 import *
2
3  s = Solver()
4
5  # Serial is 13 digits
6  serial = [BitVec('c%d' % i, 32) for i in range(13)]
7  for c in serial:
8      s.add(c >= 0)
9      s.add(c < 10)
10
11 # Partial sum
12 partials = [3]
13 for i in range(len(serial)-1):
14     p = BitVec('p%d' % i, 32)
15     s.add(p == partials[-1] + ((2*partials[-1]) ^ (serial[i])))
16     partials.append(p)
```

Z3: Formulating formulas

```
17
18 # Final check
19 s.add(serial[-1] == (partials[-1] % 10))
20
21 # Print model
22 if s.check() == sat:
23     m = s.model()
24     res = map(lambda s: m[s].as_long(), serial)
25     res = map(lambda n: chr(n+ord('0')), res)
26     print(''.join(res))
27
```

- Symbols vs. concrete values
- Pro: Explore "all" paths
- Con: Exponential complexity

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

```
< > validate.c x
1  int __cdecl validate_serial(LPCSTR serial, HWND hWnd)
2  {
3      int result; // eax@2
4      unsigned int v3; // eax@5
5      unsigned int v4; // edx@5
6      CHAR v5; // cl@6
7
8      if ( serial )
9      {
10         if ( lstrlenA(serial) == 13 )
11         {
12             v3 = 3;
13             v4 = 0;
14             do
15             {
16                 v5 = serial[v4];
17                 if ( v5 < '0' || v5 > '9' )
18                 {
19                     LoadResourceString(600, 601, hWnd);
20                     return 0;
21                 }
22                 v3 += 2 * v3 ^ (v5 - '0');
23                 ++v4;
24             }
25             while ( v4 < 12 );
26             if ( serial[12] == v3 % 10 + '0' )
27             {
28                 result = 1;
29             }
30             else
31             {
32                 LoadResourceString(600, 601, hWnd);
33                 result = 0;
34             }
35         }
36         else
37         {
38             LoadResourceString(600, 602, hWnd);
39             result = 0;
40         }
41     }
42     else
43     {
44         sub_41A4D9(0x57u);
45         result = 0;
46     }
47     return result;
48 }
```

Angr management: Minimizing the code

```
< > validate_clean.c x
1  #include <stdio.h>
2  #include <string.h>
3
4  int validate_serial(char *serial, size_t len)
5  {
6      int result;
7      unsigned int sum = 3;
8      unsigned int i = 0;
9      char current;
10
11     if ( len == 13 )
12     {
13         do
14         {
15             current = serial[i];
16             if ( current < '0' || current > '9' )
17             {
18                 return 0;
19             }
20             sum += 2 * sum ^ (current - '0');
21             ++i;
22         }
23         while ( i < 12 );
24
25         ++i;
26     }
27     while ( i < 12 );
28     if ( serial[12] == sum % 10 + '0' )
29     {
30         return 1;
31     }
32     else
33     {
34         return 0;
35     }
36 }
37
38 int main(int argc, char **argv) {
39     char serial[1024];
40     scanf("%s", serial);
41     printf("Serial: %s\nValid: %d\n", serial, validate_serial(serial, strlen(serial)));
42
43     return 0;
44 }
```


Angr management: Writing the explorer

```
< > solve_angr.py x
1  #!/usr/bin/python
2
3  import angr
4
5  def main():
6      p = angr.Project('./validator2', load_options={"auto_load_libs": False})
7      pg = p.factory.path_group()
8
9      pg.explore(find=(0x4006d7,), avoid=(0x400683,0x4006de,0x4006e5,))
10
11     found = pg.found[0]
12     return found.state.posix.dumps(0).split('\0')[0]
13
14 if __name__ == '__main__':
15     print(main())
16
```

Can we use even less effort?

- Extracting code is cumbersome
- Can't we use the code in place?
- "Call" directly into validator
- Symbolic argument
- Patch away irrelevant parts



Full fury: Writing the explorer

```
1 int __cdecl validate_serial(LPCSTR lpString, HWND hWnd)
2 {
3     int result; // eax
4     unsigned int sum; // eax
5     unsigned int i; // edx
6     CHAR cur; // cl
7
8     if ( lpString )
9     {
10         if ( lstrlenA(lpString) == 13 )
11         {
12             sum = 3;
13             i = 0;
14             do
15             {
16                 cur = lpString[i];
17                 if ( cur < 48 || cur > 57 )
18                 {
19                     LoadResourceString(600, 601, hWnd);
20                     return 0;
21                 }
22                 sum += (cur - 48) ^ 2 * sum;
23                 ++i;
24             }
25             while ( i < 0xC );
26             if ( lpString[12] == sum % 0xA + 48 )
27             {
28                 result = 1;
29             }
30             else
31             {
32                 LoadResourceString(600, 601, hWnd);
33                 result = 0;
34             }
35         }
36         else
37         {
38             LoadResourceString(600, 602, hWnd);
39             result = 0;
40         }
41     }
42     else
43     {
44         SetError(0x57u);
45         result = 0;
46     }
47     return result;
48 }
```

Full fury: Writing the explorer

```
1  #!/usr/bin/env python
2  import angr
3  import claripy
4
5  """
6  > file INSTALL.EXE
7  INSTALL.EXE: PE32 executable (GUI) Intel 80386, for MS Windows
8  > sha256sum INSTALL.EXE
9  ba155a30ca11a57a2ea917cb4f25715f79bee7397ebb16db4816e7725395e58d  INSTALL.EXE
10 """
11
12 ADDR_VALIDATE = 0x0040F8A0
13 ADDR_VALIDATE_OK = 0x0040F93F
14 ADDR_VALIDATE_BAD = [
15     0x0040F8B5,
16     0x0040F8DD,
17     0x0040F937,
18     0x0040F95B,
19 ]
20
21 ADDR_RESOURCE_STRING = 0x00402DD0
22 ADDR_SET_ERROR = 0x0041696A
23
24 # This takes a long time
25 loader = angr.cle.Loader("INSTALL.EXE", auto_load_libs=False, use_system_libs=False)
26 print('Binary loaded')
```

Full fury: Writing the explorer

```
28 proj = angr.Project(loader)
29
30 # Skip irrelevant functions
31 stub_func = angr.SIM_PROCEDURES['stubs']['ReturnUnconstrained']
32 proj.hook(ADDR_RESOURCE_STRING, stub_func())
33 proj.hook(ADDR_SET_ERROR, stub_func())
34
35 # Setup the symbolic input
36 serial_size = 32
37 sym_serial = claripy.BVS("sym_serial", serial_size * 8)
38
39 # HWND pointer can be NULL as all uses are hooked
40 state = proj.factory.call_state(ADDR_VALIDATE, sym_serial, 0)
41 simgr = proj.factory.simulation_manager(state)
42 simgr.explore(find=ADDR_VALIDATE_OK, avoid=ADDR_VALIDATE_BAD)
43
44 found = simgr.found[0] # A state that reached the find condition from explore
45 val_serial = found.solver.eval(sym_serial, cast_to=str) # Return a concrete string value for the sym arg to reach this state
46 val_serial = val_serial.strip('\x00') # Cleanup
47
48 print('Serial key: %s' % val_serial)
49
```

What about exploitation?

- IP control
- Satisfy condition

- Find execution path
- Constrain execution
- Satisfy condition

Example from Security Fest CTF

- Function pointer lookup
- Index OOB
- Hook messy function

angr exploitation example

```
1 void __fastcall __noreturn main(__int64 argc, char **argv, char **envp)
2 {
3     void (__fastcall *func_ptr)(); // rdx
4     int choice; // [rsp+0h] [rbp-10h]
5
6     setvbuf(stdin, 0LL, 2, 0LL);
7     setvbuf(stdout, 0LL, 2, 0LL);
8     alarm(0x3Cu);
9     print_welcome();
10
11     while ( 1 )
12     {
13         choice = get_choice();
14         if ( choice == -1 )
15         {
16             printf("\x1B[31;1merror:\x1B[0m not a number: %s\n", nptr);
17         }
18         else
19         {
20             memset(nptr, 0, endptr - nptr);
21             func_ptr = func_table[abs(choice) % 7];
22             ++endptr;
23             func_ptr();
24         }
25         print_menu();
26     }
27 }
```

angr exploitation example

```
1  #!/usr/bin/env python
2  import angr
3
4  BASE_ADDR = 0x400000
5  def pie_addr(addr):
6      return BASE_ADDR + addr
7
8  def hook_nop(state):
9      state.regs.rax = 0
10
11  # Setup project and patch call
12  proj = angr.Project('dist/bowrain_581bbadaafd23051a25ccb4adc80b670', load_options={'auto_load_libs': False})
13  proj.hook(pie_addr(0xFAD), hook_nop, length = 5) # The memset call does nothing of importance and messes up angr (why?)
14
```

angr exploitation example

```
0000000000000F46 ;  
0000000000000F46  
0000000000000F46 loc_F46: ; CODE XREF: main+8C+j  
0000000000000F46  
0000000000000F49      mov     eax, [rbp+choice]  
0000000000000F4C      sar     eax, 1Fh  
0000000000000F4C      mov     ecx, eax  
0000000000000F4E      xor     ecx, [rbp+choice]  
0000000000000F51      sub     ecx, eax  
0000000000000F53      mov     edx, 92492493h  
0000000000000F58      mov     eax, ecx  
0000000000000F5A      imul    edx  
0000000000000F5C      lea     eax, [rdx+rcx]  
0000000000000F5F      sar     eax, 2  
0000000000000F62      mov     edx, eax  
0000000000000F64      mov     eax, ecx  
0000000000000F66      sar     eax, 1Fh  
0000000000000F69      sub     edx, eax  
0000000000000F6B      mov     eax, edx  
0000000000000F6D      mov     [rbp+var_C], eax  
0000000000000F70      mov     edx, [rbp+var_C]  
0000000000000F73      mov     eax, edx  
0000000000000F75      shl     eax, 3  
0000000000000F78      sub     eax, edx  
0000000000000F7A      sub     ecx, eax  
0000000000000F7C      mov     eax, ecx  
0000000000000F7E      mov     [rbp+var_C], eax  
0000000000000F81      lea     rax, endptr  
0000000000000F88      mov     rax, [rax]  
0000000000000F8B      mov     rdx, rax  
0000000000000F8E      lea     rax, nptr ; "0"  
0000000000000F95      sub     rdx, rax  
0000000000000F98      mov     rax, rdx  
0000000000000F9B      mov     rdx, rax ; n  
0000000000000F9E      mov     esi, 0 ; c  
0000000000000FA3      lea     rax, nptr ; "0"  
0000000000000FAA      mov     rdi, rax ; s  
0000000000000FAD      call    memset  
0000000000000FB2      lea     rax, func_table  
0000000000000FB9      mov     edx, [rbp+var_C]  
0000000000000FBC      movsxd  rdx, edx  
0000000000000FBF      mov     rdx, (func_table - 2030A0h)[rax+rdx*8]  
0000000000000FC3      lea     rax, endptr  
0000000000000FCA      mov     rax, [rax]  
0000000000000FCD      lea     rcx, [rax+1]  
0000000000000FD1      lea     rax, endptr  
0000000000000FD8      mov     [rax], rcx  
0000000000000FDB      lea     rax, endptr  
0000000000000FE2      mov     rax, [rax]  
0000000000000FE5      mov     rdi, rax  
0000000000000FE8      mov     eax, 0  
0000000000000FED      call    rdx
```

angr exploitation example

```
14
15  # Create state and setup symbolic variable
16  state = proj.factory.blank_state(addr=pie_addr(0x000F46))
17  ADDR_CHOICE = state.regs.rbp - 0x10
18  state.mem[ADDR_CHOICE:].dword = state.solver.BVS('choice', 32)
19
20  # Find funcptr lookup instruction
21  sm = proj.factory.simgr(state)
22  sm.explore(find=pie_addr(0xFBFB))
23
24  # Pick out the state and add constraint
25  find_state = sm.found[0].state
26  find_state.solver.add(find_state.solver.Or(find_state.regs.rdx < 0, find_state.regs.rdx > 7))
27
28  # Extract and display results
29  choice = find_state.regs.rbp - 0x10
30  print('Choice: %0d' % find_state.solver.eval(find_state.mem[choice:].int64_t.resolved, cast_to=int))
31  print('RDX: %08x' % find_state.solver.eval(find_state.regs.rdx, cast_to=int))
32
```

angr exploitation example

```
> python exploit_angr.py  
Choice: 2147483648  
RDX: ffffffffef  
  
> ./bowrain_581bbadaafd23051a25ccb4adc80b670  
...  
: 2147483648  
[1] 17059 segmentation fault (core dumped)
```

Even deobfuscation?!

- Make code hard to read
 - for humans
 - for computers
- Control flow flattening
- Packer
- Dropper
- VM
- Dead code

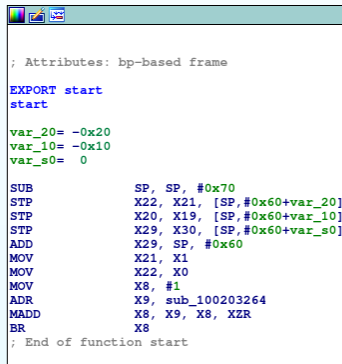
Deobfuscation in general

- Undo the mess
- Hard problem

Deobfuscation of dead code with angr

- Prove that dead code is dead
- Prove uniqueness of value

Example: indirect jmp deobfuscator



```
; Attributes: bp-based frame

EXPORT start
start

var_20= -0x20
var_10= -0x10
var_s0= 0

SUB             SP, SP, #0x70
STP             X22, X21, [SP,#0x60+var_20]
STP             X20, X19, [SP,#0x60+var_10]
STP             X29, X30, [SP,#0x60+var_s0]
ADD             X29, SP, #0x60
MOV             X21, X1
MOV             X22, X0
MOV             X8, #1
ADR             X9, sub_100203264
MADD            X8, X9, X8, XZR
BR              X8

; End of function start
```

Example from mobile app

- Find "jmp reg"
- Search callgraph backwards
- Search forward
- Simplify expression
- Replace code

Example: indirect jmp deobfuscator

```
19 def try_get_reg_value(proj, node, addr, reg):
20     state = proj.factory.blank_state(addr=node.addr)
21     simgr = proj.factory.simgr(state)
22
23     # Find call location and eval target
24     simgr.explore(find=addr)
25     if len(simgr.found) == 0:
26         print("Unconstrained")
27         return False
28     s = simgr.found[0]
29     target_addr = s.solver.eval_upto(getattr(s.regs, reg), 10)
30     if len(target_addr) ≤ 1:
31         print('Jump addr: %016x' % target_addr[0])
32         return target_addr[0]
33     else:
34         print('Non-unique addr: %016x' % target_addr[0])
35         return False
36
```

Example: indirect jmp deobfuscator

```
61 def get_reg_value(proj, cfg, addr):
62     current_function = get_target_function(cfg, addr)
63     current_node = cfg.get_any_node(addr, anyaddr=True)
64
65     reg = get_block_call_operand(current_node.block)
66     if not reg:
67         print('ERROR: Does not end with br')
68
69     while True:
70         target_addr = try_get_reg_value(proj, current_node, addr, reg)
71         if target_addr:
72             return reg, target_addr
73         current_node = bfs_back_to_function(current_node, current_function)
74         if not current_node:
75             return reg, False
76
```

Example: indirect jmp deobfuscator

```
; Attributes: bp-based frame

EXPORT start
start

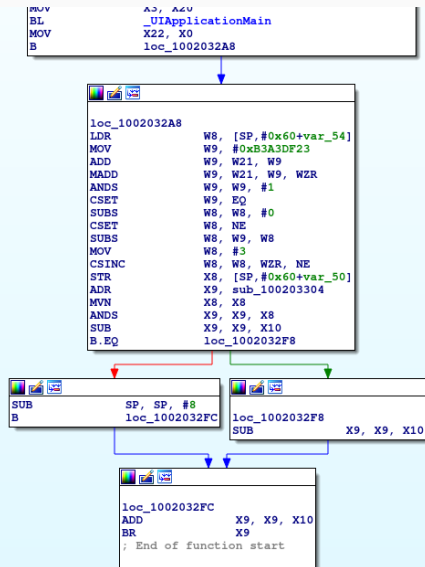
var_54= -0x54
var_50= -0x50
var_20= -0x20
var_10= -0x10
var_s0= 0

SUB         SP, SP, #0x70
STP         X22, X21, [SP, #0x60+var_20]
STP         X20, X19, [SP, #0x60+var_10]
STP         X29, X30, [SP, #0x60+var_s0]
ADD         X29, SP, #0x60
MOV         X21, X1
MOV         X22, X0
MOV         X8, #1
ADR         X9, loc_100203264
MADD        X8, X9, X8, XZR
B           loc_100203264
```

↓

```
loc_100203264
BL          _objc_autoreleasePoolPush
MOV         X19, X0
ADRP        X8, #classRef_____42@PAGE
LDR         X0, [X8, #classRef_____42@PAGEOFF] ; id
ADRP        X8, #selRef_class@PAGE
LDR         X1, [X8, #selRef_class@PAGEOFF] ; SEL
BL          _objc_msgSend
BL          _NSStringFromClass
BL          _objc_retainAutoreleasedReturnValue
MOV         X20, X0
MOV         X0, X22
MOV         X1, X21
MOV         X2, #0
MOV         X3, X20
```

Example: indirect jmp deobfuscator



Thanks for listening!