angr

List

angr

Description

- Angr은 Python기반의 바이너리 분석 프레임워크입니다.
- Angr은 정적, 동적으로 Symbolic(Concolic) 분석을 사용합니다.
- Angr은 다음과 같은 과정을 통해 분석합니다.
  - Angr은 VEX IR, SimuVex를 이용해 분석할 코드의 IR 생성과 실행을 통해 분석을 진행합니다.
    - PyVEX(VEX IR)를 이용해 분석할 대상 바이너리의 코드를 IR로 변환합니다.
    - 변환된 IR은 SimuVex를 이용해 실행 및 분석을 진행합니다.

> Github
>
> - https://github.com/angr/pyvex
> - https://github.com/angr/simuvex

API

API	Description

Angr	• **해당 API는 다음과 같은 기능을 지원합니다.**
	  • 디스 어셈블리 및 중간 표현 리프팅
	  • 프로그램 계측
	  • 상징적 실행
	  • 제어 흐름 분석
	  • 데이터 종속성 분석
	  • 가치 집합 분석 (VSA)

claripy	• Solver Engine
	  • 제약 조건을 해결하는 기능을 제공합니다.
	  • 사용방법은 z3와 유사합니다.

cle	• Binary Loader
	  • 분석 대상 바이너리에서 사용되는 라이브러리 정보들을 출력
	  • 프로세스 메모리의 추상화

pyvex	• Binary Translator • 바이너리 코드를 VEX 중간 레졸루션 (IR)으로 변환하는 인터페이스를 제공합니다
archinfo	• Arch Information Repository • 아키텍처 관련 정보를 포함하는 클래스의 모음입니다.

Github

- https://github.com/angr/angr
- https://github.com/angr/claripy
- https://github.com/angr/cle
- https://github.com/angr/pyvex
- https://github.com/angr/archinfo

Install

Command

```
lazenca0x0@ubuntu:~$ sudo apt-get install python-dev libffi-dev build-essential virtualenvwrapper

...  ...

(angr) lazenca0x0@ubuntu:~$ deactivate
lazenca0x0@ubuntu:~/Documents/angr$ workon angr
(angr) lazenca0x0@ubuntu:~/Documents/angr$
```

Execute and Exit

```
lazenca0x0@ubuntu:~/Documents/angr$ workon angr
(angr) lazenca0x0@ubuntu:~$ deactivate
lazenca0x0@ubuntu:~/Documents/angr$
```

- https://docs.angr.io/INSTALL.html

Example

Defcamp CTF Quals 2015 - r100

File

- r100

Source code

- **해당 코드는 다음과 같은 기능을 합니다.**
  - fgets() 함수를 이용해 사용자로 부터 값을 입력 받습니다.
    - 입력 받는 값의 최대 길이는 255입니다.
  - 입력받은 값은 sub_4006FD() 함수로 전달 되며, 해당 함수의 return 값에 의해 다음과 같은 문장이 출력됩니다.
    - 참 일 경우 : "Incorrect password!"
    - 거짓 일 경우 : "Nice!"

main()

```
signed __int64 __fastcall main(__int64 a1, char **a2, char **a3)
{
  signed __int64 result; // rax@3
  __int64 v4; // rcx@6
  char s; // [rsp+0h] [rbp-110h]@1
  __int64 v6; // [rsp+108h] [rbp-8h]@1

  v6 = *MK_FP(__FS__, 40LL);
  printf("Enter the password: ", a2, a3);
  if ( fgets(&s, 255, stdin) )
  {
    if ( (unsigned int)sub_4006FD(&s, 255LL) )
    {
      puts("Incorrect password!");
      result = 1LL;
    }
    else
    {
      puts("Nice!");
      result = 0LL;
    }
  }
  else
  {
    result = 0LL;
  }
  v4 = *MK_FP(__FS__, 40LL) ^ v6;
  return result;
}
```

Find address of function

- **다음과 같이 angr에서 사용될 주소를 찾습니다.**
    - angr의 Symbolic execution을 이용해 Password를 찾기 위해 다음과 같은 주소 값이 필요합니다.
    - 0x400864 영역의 jmp 명령어에 의해 다음과 같이 이동합니다.
        - Password가 틀릴 경우 이동하는 주소 : 0x400855
        - Password가 정확할 경우 이동하는 주소 : 0x400844

disassemble main

```
gdb-peda$ x/36i 0x4007E8
   0x4007e8:        push   rbp
   0x4007e9:        mov    rbp,rsp
   0x4007ec:        sub    rsp,0x110
   0x4007f3:        mov    rax,QWORD PTR fs:0x28
   0x4007fc:        mov    QWORD PTR [rbp-0x8],rax
   0x400800:        xor    eax,eax
   0x400802:        mov    edi,0x400937
   0x400807:        mov    eax,0x0
   0x40080c:        call   0x4005c0 <printf@plt>
   0x400811:        mov    rdx,QWORD PTR [rip+0x200850]        # 0x601068 <stdin>
   0x400818:        lea    rax,[rbp-0x110]
   0x40081f:        mov    esi,0xff
   0x400824:        mov    rdi,rax
   0x400827:        call   0x4005e0 <fgets@plt>
   0x40082c:        test   rax,rax
   0x40082f:        je     0x400866
   0x400831:        lea    rax,[rbp-0x110]
   0x400838:        mov    rdi,rax
   0x40083b:        call   0x4006fd
   0x400840:        test   eax,eax
   0x400842:        jne    0x400855
   0x400844:        mov    edi,0x40094c
   0x400849:        call   0x4005a0 <puts@plt>
   0x40084e:        mov    eax,0x0
   0x400853:        jmp    0x40086b
   0x400855:        mov    edi,0x400952
   0x40085a:        call   0x4005a0 <puts@plt>
   0x40085f:        mov    eax,0x1
   0x400864:        jmp    0x40086b
   0x400866:        mov    eax,0x0
   0x40086b:        mov    rcx,QWORD PTR [rbp-0x8]
   0x40086f:        xor    rcx,QWORD PTR fs:0x28
   0x400878:        je     0x40087f
   0x40087a:        call   0x4005b0 <__stack_chk_fail@plt>
   0x40087f:        leave
   0x400880:        ret
gdb-peda$
```

Source code

- **다음과 같이 코드를 작성 할 수 있습니다.**
  - "r100" 바이너리를 분석하기 위해 우선 angr.Project() API를 이용해 Project를 생성합니다.
  - path_group() API를 이용해 새로운 Path group를 생성합니다.
  - explore() API 이용해 찾을 경로와 피해야 할 경로를 설정합니다.
    - 찾을 경로 : find=0x400844
    - 피해야 할 경로 : avoid=0x400855
  - explore() API를 이용해 0x400844 영역으로 가기 위한 값을 확인합니다.

poc.py

```
import os
import angr

project = angr.Project("r100", auto_load_libs=False)
path_group = project.factory.path_group()
path_group.use_technique(angr.exploration_techniques.DFS())
avoid_addr = [0x400855]
find_addr = 0x400844

path_group.explore(find=find_addr, avoid=avoid_addr)
print path_group.found[0]
print path_group.found[0].state.posix.dumps(0)
```

- 다음과 같이 주소가 아닌 문자로 찾을 경로를 설정할 수 도 있습니다.

poc2.py

```
import os
import angr

project = angr.Project("defcamp_quals_2015_r100", auto_load_libs=False)
path_group = project.factory.path_group()
path_group.explore(find=lambda path: 'Nice!' in path.state.posix.dumps(1))
print path_group.found[0].state.posix.dumps(0)
```

- project.factory.path_group() : http://angr.io/api-doc/angr.html#angr.path_group.PathGroup
- path_group.explore() : http://angr.io/api-doc/angr.html#angr.path_group.PathGroup.explore

Result

- 다음과 같이 스크립트를 이용해 password값을 확인 할 수 있습니다.

Command

```
(angr) lazenca0x0@ubuntu:~/Documents/angr$ python symbolicECE.py
WARNING | 2017-09-06 23:27:40,112 | claripy | Claripy is setting the recursion limit to 15000. If Python
segfaults, I am sorry.
Code_Talkers

(angr) lazenca0x0@ubuntu:~/Documents/angr$ ./r100
Enter the password: Code_Talkers
Nice!
(angr) lazenca0x0@ubuntu:~/Documents/angr$
```

angr with it

- **다음과 같은 Tool들을 angr과 같이 사용할 수 있습니다.**

List

Tool		github
angrop	자동으로 gadget을 찾고 chain을 생성합니다.	https://github.com/salls/angrop
Patcherex	자동으로 바이너리 패치를 작성하는데 사용됩니다.	https://github.com/shellphish/patcherex
rex	자동으로 Exploit을 생성하는데 사용됩니다.	https://github.com/shellphish/rex
Driller	angr을 이용해 AFL의 성능을 향상 시켰습니다.	https://github.com/shellphish/driller

Relation site

- https://angr.io
- http://angr.io/api-doc/
- https://github.com/angr/angr
- https://github.com/angr/angr-doc
- https://github.com/axt/angr-utils