Angry Hacking

How angr pwned CTFs and the CGC
Motivation 6 mins
Fundamentals of angr 3 mins
   Pure awesomeness
Live demos 20 mins
   Symbolic execution
   Static analysis
   Emulation
angr applications 10 mins
   Rop gadget finder
   Binary diffing
   Cyber Grand Challenge
Open source! 3 minutes
   http://angr.io
   Credits
Why angr?
2005 Hex-Rays was founded
2007 Hex-Rays Decompiler 1.0
2009 Hex-Rays IDA 5.5
2011 Hex-Rays IDA 6.1
2013 Hex-Rays IDA 6.4
2015 ???
How standards proliferate:
(see: A/C chargers, character encodings, instant messaging, etc)

Situation:
There are 14 competing standards.

14?! Ridiculous! We need to develop one universal standard that covers everyone’s use cases.

Yeah!

(Soon:

Situation:
There are 15 competing standards.)
Fundamentals of angr
• iPython-accessible
• powerful analyses
• versatile
• well-encapsulated
• open and expandable
• architecture "independent"
%quickref -> Quick reference.
help    -> Python's own help system.
object? -> Details about 'object', use 'object??' for extra details.

In [1]: import angr

[an gr.init] | INFO: Largescale module not available

e. Clone from git if needed.

In [2]: p = angr.Project('/bin/echo')

[cle.generic] | WARNING: Unknown reloc type: 37

In [3]: p.
p.arch      p.filename      p.loader
p.entry     p.hook          p.set_sim_procedure
p.factory   p.is_hooked    p.unhook

In [3]: p.factory.
p.factory.analyses p.factory.path
p.factory.blank_state p.factory.path_group
p.factory.block     p.factory.sim_block
p.factory.entry_state p.factory.sim_run
p.factory.full_init_state p.factorysurveyors

In [3]: p.factory.
Binary Loader

Static Analysis Routines

Symbolic Execution Engine

Control-Flow Graph

Data-Flow Analysis

Value-Set Analysis
ARE YOU READY FOR THE ANGRY POWER?
Victim binary
Symbolic execution
"How do I trigger path X or condition Y?"

- **Dynamic analysis**
  - Based on concrete inputs to application.
- **(Concrete) static analysis**
  - "You can't"/"You might be able to"
  - Based on various static techniques.

We need something slightly different.
"How do I trigger path X or condition Y?"

1. Interpret the application.
2. Track "constraints" on variables.
3. When the required condition is triggered, "concretize" to obtain a possible input.
Constraint solving:

- Conversion from set of constraints to set of concrete values that satisfy them.
- NP-complete, in general.
x = int(input())
if x >= 10:
    if x < 100:
        print "Two!"
    else:
        print "Lots!"
else:
    print "One!"
```python
x = int(input())
if x >= 10:
    if x < 100:
        print "Two!"
    else:
        print "Lots!"
else:
    print "One!"
```

State A

Variables

x = ???

Constraints

------
```python
x = int(input())
if x >= 10:
    if x < 100:
        print "Two!"
    else:
        print "Lots!"
else:
    print "One!"
```
x = int(input())
if x >= 10:
    if x < 100:
        print "Two!"
    else:
        print "Lots!"
else:
    print "One!"

State AA
Variables
x = ???
Constraints
x < 10

State AB
Variables
x = ???
Constraints
x >= 10
x = int(input())
if x >= 10:
    if x < 100:
        print "Two!"
    else:
        print "Lots!"
else:
    print "One!"
```python
x = int(input())
if x >= 10:
    if x < 100:
        print "Two!"
    else:
        print "Lots!"
else:
    print "One!"
```
Static analysis
<table>
<thead>
<tr>
<th>Variable recovery</th>
<th>Range recovery</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wrapped-interval analysis</td>
<td>Value-set analysis</td>
</tr>
</tbody>
</table>

Abstract interpretation
What a value-set looks like

```
{
    ( global, (4[0x601000, 0x602000], 32) ),
    ( stack_0x400957, (8[-0xc, -0x4], 32) )
}
```
angr applications
ROP gadget finder
Binary differencing
Cyber Grand Challenge
Shellphish CRS

PCAP

CB

Autonomous processing

Autonomous vulnerability scanning

Test cases

Proposed POVs

Autonomous service resiliency

Proposed RBs
Vulnerability Discovery via SymExec

Program

Symbolic inputs

Symbolic execution engine

Security policy checker

POVs
KEEP CALM
ITS TIME FOR A DEMO!
Open Source
Major contributors:
- Zardus - Yan Shoshitaishvili
- Fish - Ruoyu Wang
- kereoz - Christophe Hauser
- rhelmot - Andrew Dutcher
- nezorg - John Grosen
- salls - Chris Salls

Special thanks to:
- our professors
- DARPA VET Project
- DARPA Cyber Grand Challenge
Open angr!

→ http://angr.io
→ https://github.com/angr
→ angr@lists.cs.ucsb.edu

Pull requests, issues, questions, etc super-welcome! Let's bring on the next generation of binary analysis!
Draft and backups
• motivation (keep it quick) - 6 mins
  ○ "In the beginning, there was IDA. However, as the field of binary security advanced, there is now … still IDA?"
  ○ We need something more!
  ○ There are a few solutions, but they all suffer from lacking one of: cross-platform, open, active, usable.
• angr fundamentals - 3 mins
  ○ power (state-of-the-art)
  ○ ease of use (abstraction)
  ○ expandable, cross-platform, blah blah
• main components - 20 minutes
  ○ introduce a demo: some combination of a crackme and a pwnable
  ○ symbolic execution (slides + demo)
    ■ the demo should get us past the crackme portion using angr's symbolic execution
  ○ VSA (slides + demo)
    ■ the demo should allow us to identify an overflow to pwn
  ○ dynamic execution (slides + demo)
    ■ we'll demo a shellcode that's used to exploit the overflow
• angr applications - 10 minutes
  ○ rop gadget finder (demo)
  ○ binary differ
  ○ Cyber Grand Challenge
• open source! - 3 minutes
  ○ http://angr.io