A Dozen Years of Shellphish From DEFCON to the Cyber Grand Challenge

Antonio Bianchi

Jacopo Corbetta

Andrew Dutcher

antoniob@cs.ucsb.edu jacopo@cs.ucsb.edu dutcher@cs.ucsb.edu





Chaos Communication Congress / December 29th, 2015





- Who are we?
 - A team of security enthusiasts
 - do research in System Security
 - play Capture the Flag competitions
 - released a couple of tools

Shellphish



Started (in 2004) at: SecLab: University of California, Santa Barbara



Shellphish



- expanded to:
 - Northeastern
 University: Boston
 - Eurecom: France





Mini-primer: What's a CTF



- Security competition:
 - exploit a vulnerable service / website / device
 - reverse a binary

Ο.

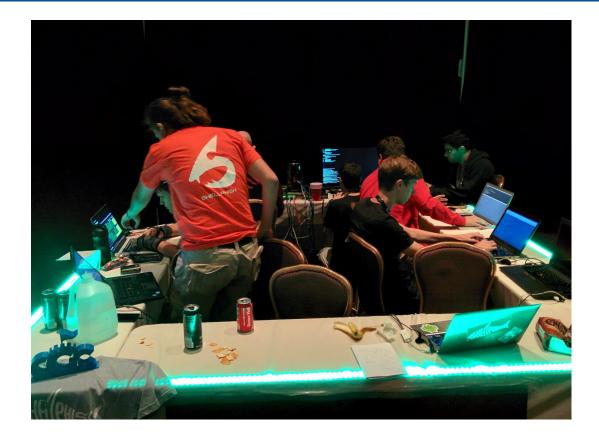
- Different formats
 - Jeopardy Attack-Defense
 - Online Live

с.

• Basic idea: find the secret, submit to organizers, ... profit

Shellphish









- We do not only play CTFs
- We also organize them!
 - UCSB iCTF
 - Attack-Defense format
 - every year, since 2002!
 - Try to innovate with a different style every year
 - Site: ictf.cs.ucsb.edu
 - Base: <u>github.com/ucsb-seclab/ictf-framework</u>
 - Vigna, et al., *"Ten years of ictf: The good, the bad, and the ugly."* 3GSE, 2014.

Why we're here









- DARPA Cyber Grand Challenge (CGC) The (almost-)Million Dollar Baby
- Our Cyber Reasoning System (CRS) Fancy term for auto-playing a CTF
- Automated Vulnerability Discovery
- Live example using **angr**
 - Open-source binary analysis framework
- Towards the Cyber Grand Challenge Finals (CFE)

Cyber Grand Challenge (CGC)



2004: DARPA Grand Challenge
 Autonomous vehicles



Cyber Grand Challenge (CGC)



2014: DARPA Cyber Grand Challenge
 Autonomous hacking!



Cyber Grand Challenge (CGC)

- Started in 2014
- Qualification event: June 3rd, 2015, online
 ~70 teams → 7 qualified teams
- Final event: August 4th, 2016 @ DEFCON (Las Vegas)
 Winning CRS will also play against humans!

cybergrandchallenge.com / cgc.darpa.mil







CGC Rules



- Attack-Defense CTF
- Solving security challenges → Developing a system that solves security challenges
- Develop a system that **automatically**
 - Exploit vulnerabilities in binaries
 - Patch binaries, removing the vulnerabilities
- No human intervention
- Think it's trivial? How would you play?
 And how would you organize it?

CGC – Rules



- Exploits
 - "getting a flag" (how? where?)
 - For the quals: exploit = **crash**
- Defend
 - o int main() { return 0; }
 - Functionality checks
 - o SIGSEGV => exit(0)
 - No easy "out-of-band" error handling
 - QEMU-style interpreter: interrupts => exit(0)
 Performance cost (CPU, memory, file size)

No "extra" complications

Is modeling the entire POSIX API a good use of team resources? What about the file systems? Or horrible things like interruptible syscalls?



CGC Qualification Event – Rules

- Basic idea:
 - Real(istic) programs

CGC Qualification Event – Rules



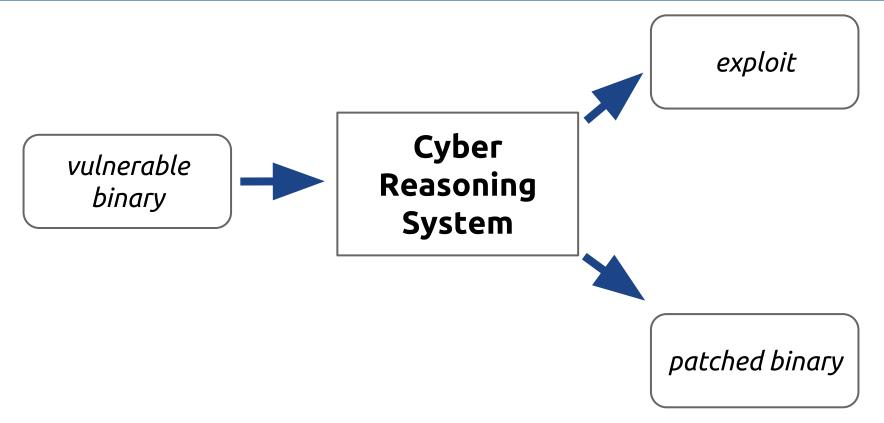
- Architecture: Intel x86, 32-bit
- OS: DECREE
 - Linux-like, but with 7 syscalls only
 - ∎ transmit / receive / fdwait (≈select)
 - allocate / deallocate (even executable!)
 - random
 - __terminate
 - no signals, threads, shared memory
- "Bring Your Own Defense" approach (and pay for it)
 Not even "the usual": stack is executable, no ASLR, ...





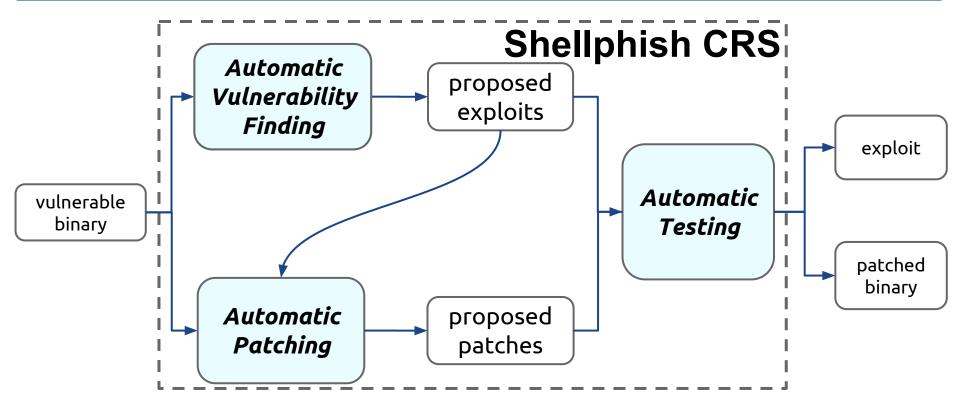
- DARPA Cyber Grand Challenge (CGC) The (almost-)Million Dollar Baby
- Our Cyber Reasoning System (CRS) Fancy term for auto-playing a CTF
- Automated Vulnerability Discovery
- Live example using angr Open-source binary analysis framework
- Towards the Cyber Grand Challenge Finals (CFE)





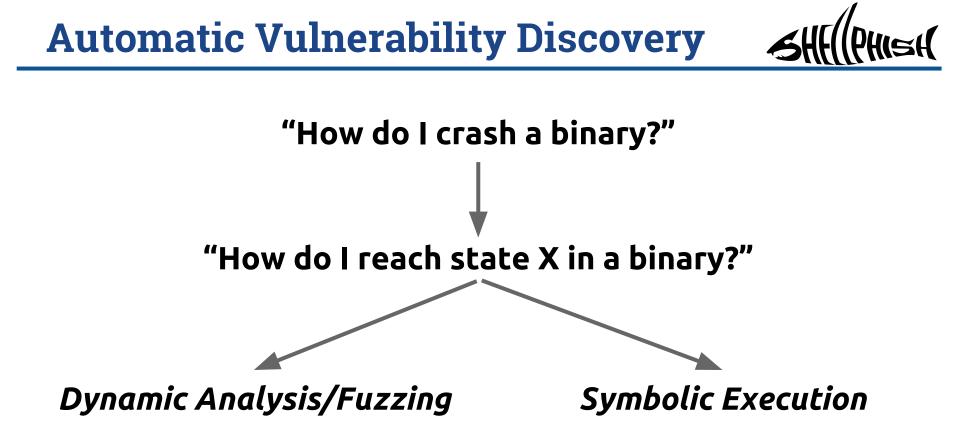
Shellphish CRS







- DARPA Cyber Grand Challenge (CGC) The (almost-)Million Dollar Baby
- Our Cyber Reasoning System (CRS) Fancy term for auto-playing a CTF
- Automated Vulnerability Discovery
- Live example using angr Open-source binary analysis framework
- Towards the Cyber Grand Challenge Finals (CFE)



Dynamic Analysis/Fuzzing



• How do I reach the state: "You win!" is printed?

```
x = int(input())
if x >= 10:
  if x < 100:
print "You win!"
  else:
     print "You lose!"
else:
  print "You lose!"
```

Dynamic Analysis/Fuzzing

• How do I reach the state: "You win!" is printed?





- How did we use Fuzzing for CGC?
- Coverage-guided fuzzing
 Looking for "crashing inputs"
 - Based on AFL <a>lcamtuf.coredump.cx/afl/
- In general, it cannot work in some cases
 e.g., "magic numbers", computations, ...

Dynamic Analysis/Fuzzing



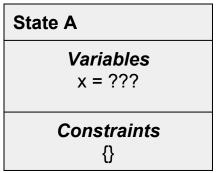
• How do I reach the state: "You win!" is printed?

```
x = int(input())
if x >= 10:
  if x^2 == 152399025:
     print "You win!"
  else:
     print "You lose!"
else:
  print "You lose!"
```



Interpret the binary code and replace user-input with symbolic variables

```
x = int(input())
if x >= 10:
  if x < 100:
     print "You win!"
  else:
     print "You lose!"
else:
  print "You lose!"
```





Variables

x = ???

Constraints

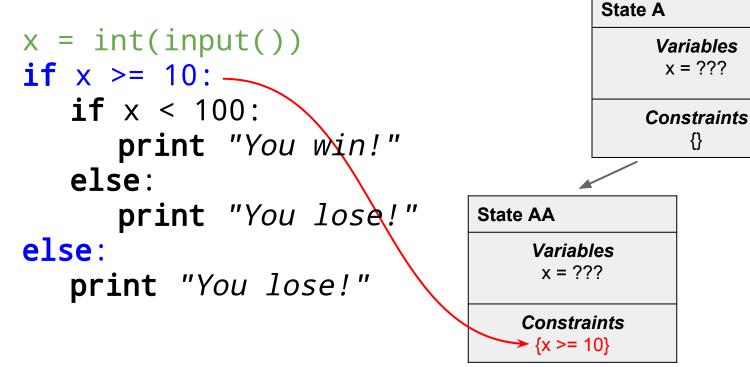
{}

Interpret the binary code and replace user-input with symbolic variables
 State A

x = int(input())**if** x >= 10: **if** x < 100: print "You win!" else: print "You lose!" else: print "You lose!"

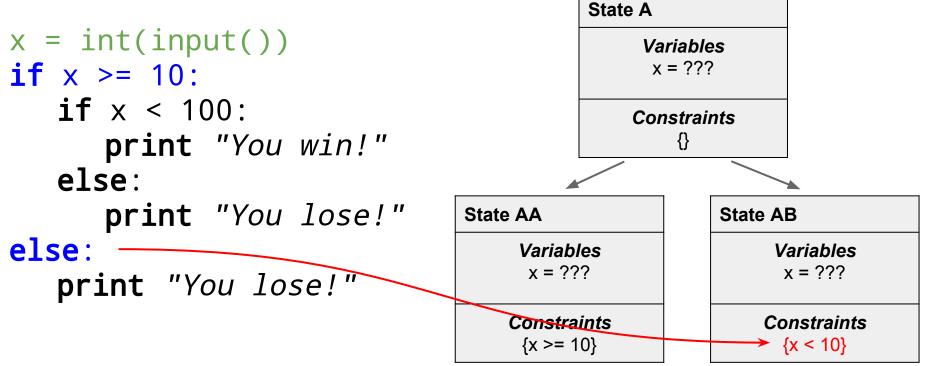


- SHELL PHISH
- Follow all feasible paths, tracking "constraints" on variables



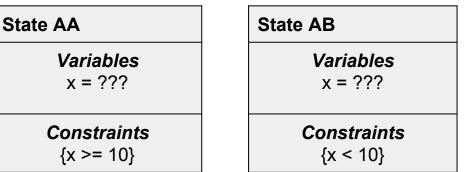


- SHE((PHISH
- Follow all feasible paths, tracking "constraints" on variables





- Follow all feasible paths, tracking "constraints" on variables
- x = int(input()) **if** x >= 10: **if** x < 100: print "You win!" else: print "You lose!" else: print "You lose!"



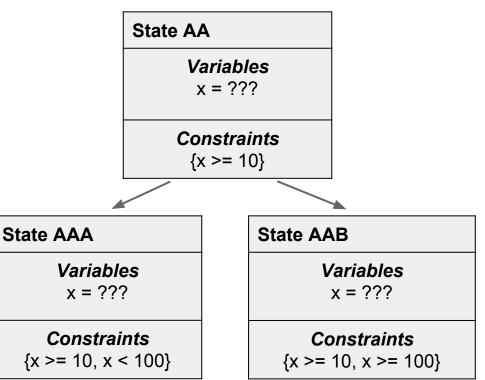


Follow all feasible paths, tracking "constraints" on variables

```
x = int(input())
if x >= 10:
  if x < 100:
     print "You win!"
  else:
     print "You lose!"
else:
  print "You lose!"
```

State AA	
Variables x = ???	
<i>Constraints</i> {x >= 10}	

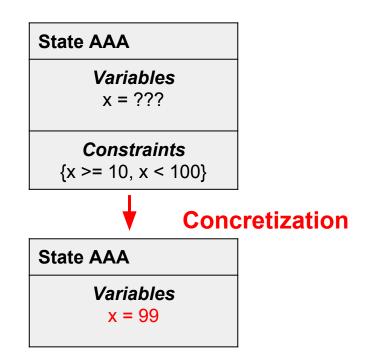
- SHE((PHIS
- Follow all feasible paths, tracking "constraints" on variables
- x = int(input()) **if** x >= 10: **if** x < 100: print "You win!" else: print "You lose!" else: print "You lose!"





• Concretize the constraints on the symbolic variables

```
x = int(input())
if x >= 10:
  if x < 100:
     print "You win!"
  else:
     print "You lose!"
else:
  print "You lose!"
```

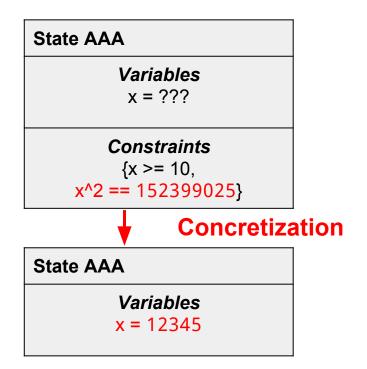


Symbolic Execution



• Concretize the constraints on the symbolic variables

```
x = int(input())
if x >= 10:
  if x^2 == 152399025:
     print "You win!"
  else:
     print "You lose!"
else:
  print "You lose!"
```



Symbolic Execution



- How did we use Symbolic Execution for CGC?
 We used the symbolic execution engine of **angr**, the binary analysis platform developed at UCSB
- Symbolically execute the binaries looking for
 - 1. Memory accesses outside allocated regions
 - 2. "Unconstrained" instruction pointer (e.g., controlled by user input)
 - eax = <user input>, jmp eax
- If either 1. or 2. is true
 → we found an input that will make the program crash

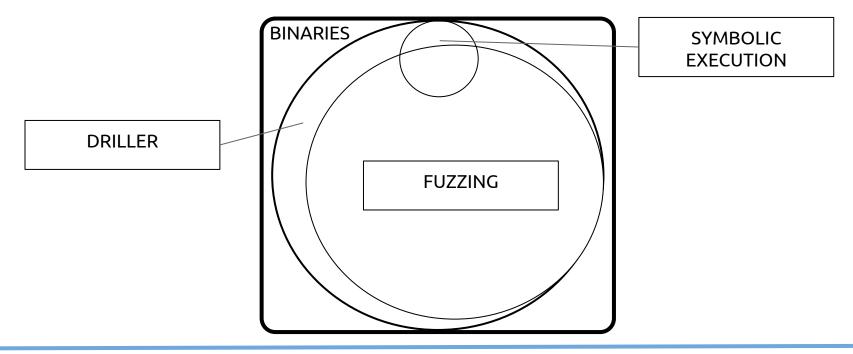


- Combining the two approaches
- *"Driller: Augmenting Fuzzing Through Selective Symbolic Execution"*
 - Network and Distributed System Security
 Symposium (NDSS), February 2016, San Diego

Future directions



• *"Driller: Augmenting Fuzzing Through Selective Symbolic Execution"*







- DARPA Cyber Grand Challenge (CGC) The (almost-)Million Dollar Baby
- Our Cyber Reasoning System (CRS) Fancy term for auto-playing a CTF
- Automated Vulnerability Discovery
- Live example using angr Open-source binary analysis framework
- Towards the Cyber Grand Challenge Finals (CFE)



- Binary analysis platform, developed at UCSB
 - Open source: <u>github.com/angr</u> (star it!)
- Architecture independent
 - x86 (ELF, CGC, PE), amd64, mips, mips64, arm, aarch64, ppc, ppc64









- Written in Python!
 - installable with one (two?) command!
 - mkvirtualenv angr

(optional, but don't complain if it's broken)

- pip install angr
- interactive shell (using IPython)
- it has an interactive GUI

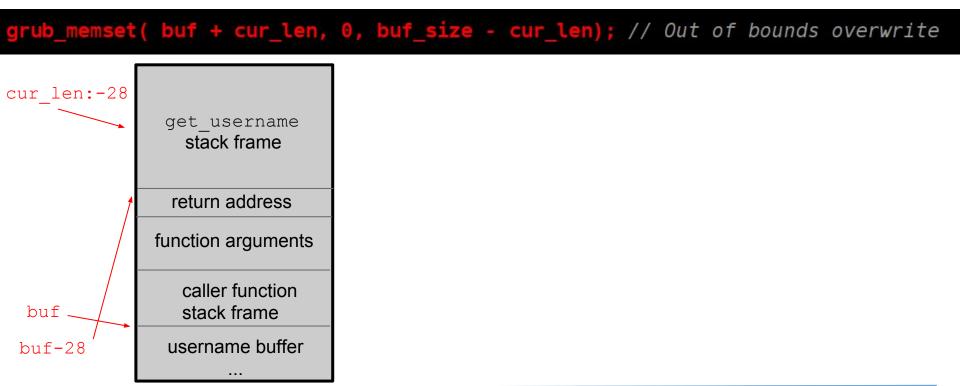


Grub: "Back to 28" vulnerability

- Pressing backspace 28 times on the grub username prompt can get you a rescue shell
- <u>http://hmarco.org/bugs/CVE-2015-8370-Grub2-</u> <u>authentication-bypass.html</u>

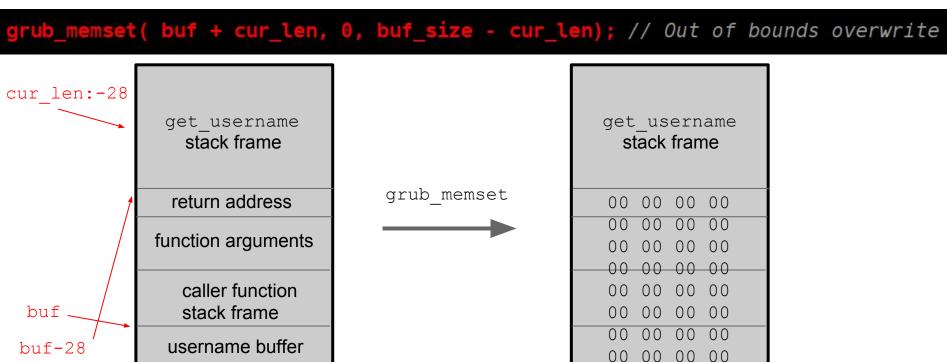


Grub: "Back to 28" vulnerability





Grub: "Back to 28" vulnerability





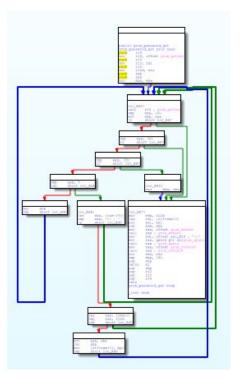
Grub: "Back to 28" vulnerability

- Somehow, jumping to 00:0000 is completely exploitable
- Way beyond the scope of this demo





Grub: "Back to 28" vulnerability



- The correct path to the exploit goes around this loop 28 times, each of which has to follow a specific path
- The universe will grow old and die before naive symbolic execution finds this bug
- Demonstration: this doesn't work, really!
- A technique (implemented by angr) called veritesting¹ solves this problem in *some cases* by merging states when their instruction pointers converge, but in *this case* the complexity generated is too much for the constraint solver

¹<u>http://users.ece.cmu.edu/~aavgerin/papers/veritesting-icse-2014.pdf</u>

- Symbolic execution is powerful
- Symbolic execution is stupid
- You are incredibly weak
- You are very clever

Use angr to unlock your true potential







Grub: "Back to 28" vulnerability

Manual examination of the state explosion tells you:

- Where the wasted computational power is going
- How to be more efficient

The naive approach is doing lots of weird things like entering letters and then deleting them again and again, or pressing the "home" key several times in a row, which don't produce any interesting new states to analyze.

You can fix this!





Grub: "Back to 28" vulnerability

Finding the bug





- DARPA Cyber Grand Challenge (CGC) The (almost-)Million Dollar Baby
- Our Cyber Reasoning System (CRS) Fancy term for auto-playing a CTF
- Automated Vulnerability Discovery
- Live example using angr Open-source binary analysis framework

• Towards the Cyber Grand Challenge Finals (CFE)



- 7 teams passed the qualification phase
- Shellphish is one of them! :-)
- We exploited 44 binaries out of 131
- Every qualified team received \$750,000 !



- Different setup
 - Round-based attack-defense CTF
 - Probably, zero human intervention allowed
 - Not even bug fixing?
 - Data about previous rounds is available:
 - submitted exploits/patched binaries performance
 - (anonymized) network traffic
 - patches from other teams
 - Stealing patched binaries/exploits?





- Exploits are more realistic:
 - Two types:
 - Crash at a specific location and set a specific register to a specific value
 - Leak data from a specific memory page
 - We'll need a more realistic exploit generator:

angr automatic ROP-chain builder!

• Every team can also deploy network-level filtering rules

- Every team has access to a cluster of:
 - **1280 cores**
 - 16 TB of RAM
 - 128 TB of storage







6 7		1 [] 2 [] 3 [] 4 [] 5 [] 7 []		21 [22 [23 [24 [25 [26 [27 [31 [32 [33 [34 [35 [36 [37 [
9 10	1 1 1 2 1725/257	7 [] 8 [] 9 [] 90 [] 931MB] 7931MB]		Load]] s: 21, averag	38 [39 [40 [4 thr, ge: 0.0 days,	00	г 0. З
PID	USER	PRI	NI	VIRT	RES	SHR		CP
0357	team	20	0	VIRT 22944	2048	1340	R	1
0357	team team	20 20	0	VIRT 22944 95080	2048 1784	1340 856	R	1
0357 0304 1246	team team root	20 20 20	0000	VIRT 22944 95080 176M	2048 1784 34592	1340 856 2764	RSS	100
0357 0304 1246 794	team team root syslog	20 20 20 20	000000000000000000000000000000000000000	VIRT 22944 95080 176M 253M	2048 1784 34592 2464	1340 856 2764 1072	RNNN	1000
0357 0304 1246 794 1178	team team root syslog root	20 20 20 20 20 20	000000	VIRT 22944 95080 176M 253M 23656	2048 1784 34592 2464 1036	1340 856 2764 1072 788	RNNNN	10000
0357 0304 1246 794 1178 1270	team team root syslog root root	20 20 20 20 20 20 20		VIRT 22944 95080 176M 253M 23656 13396	2048 1784 34592 2464 1036 752	1340 856 2764 1072 788 532	20 N N N N	100000
0357 0304 1246 794 1178 1270 831	team root syslog root root syslog	20 20 20 20 20 20 20 20	000000	VIRT 22944 95080 176M 253M 23656 13396 253M	2048 1784 34592 2464 1036 752 2464	1340 856 2764 1072 788	RNNNNN	1000000
0357 0304 1246 794 1178 1270 831	team team root syslog root root	20 20 20 20 20 20 20 20 20 20	000000000000000000000000000000000000000	VIRT 22944 95080 176M 253M 23656 13396 253M 253M	2048 1784 34592 2464 1036 752 2464 2464	1340 856 2764 1072 788 532 1072	RNNNNN	100000
0357 0304 1246 794 1178 1270 831	team root syslog root root syslog syslog root	20 20 20 20 20 20 20 20 20 20	000000000000000000000000000000000000000	VIRT 22944 95080 176M 253M 23656 13396 253M 253M 33328	2048 1784 34592 2464 1036 752 2464	1340 856 2764 1072 788 532 1072 1072	RNNNNNN	0000000
0357 0304 1246 794 1178 1270 831 833 1	team root syslog root root syslog syslog	20 20 20 20 20 20 20 20 20 20 20	000000000000000000000000000000000000000	VIRT 22944 95080 176M 253M 23656 13396 253M 253M	2048 1784 34592 2464 1036 752 2464 2464 2600	1340 856 2764 1072 788 532 1072 1072 1444	R. ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~	00000000
0357 0304 1246 794 1178 1270 831 833 1 694	team root syslog root root syslog syslog root root	20 20 20 20 20 20 20 20 20 20 20 20	000000000000000000000000000000000000000	VIRT 22944 95080 176M 253M 23656 13396 253M 253M 33328 20940	2048 1784 34592 2464 1036 752 2464 2464 2600 2256	1340 856 2764 1072 788 532 1072 1072 1072 1444 632	^R ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~	0000000000
0357 0304 1246 794 1178 1270 831 833 1 694 709	team root syslog root root syslog syslog root root root	20 20 20 20 20 20 20 20 20 20 20	000000000000000000000000000000000000000	VIRT 22944 95080 176M 253M 23656 13396 253M 253M 33328 20940 51180	2048 1784 34592 2464 1036 752 2464 2464 2600 2256 1552	1340 856 2764 1072 788 532 1072 1072 1072 1444 632 1012	๛๛๛๛๛๛๛๛	



| | Interrightspace (3 = 6602) Interig
 |

 | Image: second

 | Image:proj.2. -66/32 Image:proj.2. | Imardipupation Constraint The standard standard Standard standard Standard standard | | Transport Transport Transport Transport
 |
 | Inscriptops (1.6.4602) 1 1
 | tangopoil 0.000 1 0.000 0.000 0.000 1 0.000 0.000 0.000 0.000 1 0.000 0.000 0.000 0.000 0.000 1 0.000 0.000 0.000 0.000 0.000 0.000 1 0.000 | tear(purple): 4 - 4603 tear(purple): 4 |
 | Them@purple.22 -6102 Image: property and prope | Image: pre-22 - 6002 | |
--

--
--
--
---	---
--

--
--	--
--	--

133 core 3 core 3	10 10 10 2 20 10 20 20 10 20 50 10<
 | State State <th< td=""><td>11 11
 11 11 11 11 11 11 11<</td><td>All All All</td></th<> <td>121 2 mont 2 3 mont 2 4 mont 2</td> <td></td> <td>100 0 100</td> <td>John mark 2 1</td> <td>State State <th< td=""><td>Number 0<td>120 100</td></td></th<><td>$\begin{array}{c c c c c c c c c c c c c c c c c c c$</td><td>No. No. Solution Solution 1 2 1 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 2 1 2 2 1 2 2 1 2</td><td>State State <th< td=""><td>Start prime 0 1 10 0 10 1</td></th<></td></td> | 11 11 11 11 11
 11 11<
 | All | 121 2 mont 2 3 mont 2 4 mont 2 |
 | 100 0 100
 | John mark 2 1
 | State State <th< td=""><td>Number 0<td>120 100</td></td></th<> <td>$\begin{array}{c c c c c c c c c c c c c c c c c c c$</td> <td>No. No. Solution Solution 1 2 1 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 2 1 2 2 1 2 2 1 2</td> <td>State State <th< td=""><td>Start prime 0 1 10 0 10 1</td></th<></td> | Number 0 <td>120 100</td>
 | 120 100 | $ \begin{array}{c c c c c c c c c c c c c c c c c c c $ | No. No. Solution Solution 1 2 1 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 2 1 2 2 1 2 2 1 2
 2 | State State <th< td=""><td>Start prime 0 1 10 0 10 1</td></th<> | Start prime 0 1 10 0 10 1 |
| |

 |
 |

 | $ \begin{array}{cccccccccccccccccccccccccccccccccccc$ | | 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 |
 | 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 | | 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 | 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 | | | t d d d d d d d d d d d d d d d d d d d |
| | Non- Composition Composition <thcomposition< th=""> <thc< td=""><td></td><td>110 100<td>Bill (mail) Control Contro Control <thcontrol< th=""></thcontrol<></td><td>110 (1997) 20 (2007) 20 (</td><td>Dia Name Dia Name</td><td>Attention 0.0 # 2000 10000 1000 1000</td><td>Total mean Str. Total Med. Str. Str.<td>Dirit man So Titte and
so So Titte and
so So So</td><td>Ball Start 50 500 5</td><td>Hill: Start 2, 5 + 1000 Hill: Start 2, 5 + 1000 Hill:</td><td>100 100<td>121 max 250 100.4 40.0 120.4 100.4 120.4 100.4<</td><td>Life rest 20 Time rest 20 SUS rest 20 2000
 2000 2000</td><td>100 100</td></td></td></td></thc<></thcomposition<> |
 | 110 100 100 100 100 100 100 100 100 100 100 100 100 100 100
 100 100 <td>Bill (mail) Control Contro Control <thcontrol< th=""></thcontrol<></td> <td>110 (1997) 20 (2007) 20 (</td> <td>Dia Name Dia Name</td> <td>Attention 0.0 # 2000 10000 1000 1000</td> <td>Total mean Str. Total Med. Str. Str.<td>Dirit man So Titte and
so So Titte and
so So So</td><td>Ball Start 50 500 5</td><td>Hill: Start 2, 5 + 1000 Hill: Start 2, 5 + 1000 Hill:</td><td>100 100<td>121 max 250 100.4 40.0 120.4 100.4 120.4 100.4<</td><td>Life rest 20 Time rest 20 SUS rest 20 2000</td><td>100 100
 100 100</td></td></td> | Bill (mail) Control Contro Control <thcontrol< th=""></thcontrol<> | 110 (1997) 20 (2007) 20 (| Dia Name | Attention 0.0 # 2000 10000 1000 1000
 | Total mean Str. Total Med. Str. Str. <td>Dirit man So Titte and
so So Titte and
so So So</td> <td>Ball Start 50 500 5</td> <td>Hill: Start 2, 5 + 1000 Hill: Start 2, 5 + 1000 Hill:</td> <td>100 100<td>121 max 250 100.4 40.0 120.4 100.4 120.4 100.4<</td><td>Life rest 20 Time rest 20 SUS rest 20 2000</td><td>100 100
 100 100</td></td> | Dirit man So Titte and
so So Titte and
so So | Ball Start 50 500 5
 | Hill: Start 2, 5 + 1000 Hill: | 100 100 <td>121 max 250 100.4 40.0 120.4 100.4 120.4 100.4<</td> <td>Life rest 20 Time rest 20 SUS rest 20 2000</td> <td>100 100</td> | 121 max 250 100.4 40.0 120.4 100.4 120.4 100.4<
 | Life rest 20 Time rest 20 SUS rest 20 2000 | 100 100 |
| |

 |
 | 1 1 1 2 1

 | 1 1 0 1 1 0 0 1 1 0 0 1 1 0 1 0 1 0 1 0 | 1 | 1 1 1 1 1 1 1 1 1 2 1 2 1 1 2 1 1 2 1 1 1 1 1 1 1 1 2 1 2 1 1 2 1 1 1 2 1 1 1 2 1
 |
 | 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 | 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 1 4 4 0 0 4 0 2 1 0 2 1 0 2 1 0 2 1 0 2 1 0 0 1 0 1
 | 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 1 1 0 1 1 0 2 1 0
2 1 0 2 1 0 2 1 0 2 1 0 2 1 0 2 1 0 | | 1 1 1 0 1 1 2 2 2 1 2 1 2 1 1 1 2 2 1 2 1 1 2 1 1 2 1 1 2 1 1 2 1 1 2 1 1 2 1 1 2 1 1 2 1 1 2 1 1 2 1 1 2 1 1 2 1 1 2 1 1 2 1 1 2 1 1 2 1 1 1 2 1 | 1 1 1 2 1 |
| | 131 cpm 2 6 131 2500 231 231 131 cpm 2 2 231 232 331 1 141 cpm 2 2 231 122 735 1 147 cpm 2 2 231 122 735 1 147 cpm 2 2 231 122 735 1 147 cpm 2 2 231 122 735 1 </td <td>Lit Test <tht< td=""><td>122 press
123 press
124 press
125 press
125 press
126 press</td><td>Diff rest 2 1116 2010 1116 2010 1116 2010 <</td><td>115 State 2 2280 112 283 125 125 125 125 125 125 125 125 126 125 125 125 127 127 127 126 126 126 127</td><td>Hill was 30 2000 1000 200 1000 200 1000 200 1000</td><td>100 100<td>30.55 tem 2.5 0.105 1.10 1.05 0.10 1.05 0.10 0.15 0.10 0.15 0.10 0.10 0.15 0.10 <th0.10< th=""> 0.10 0.10 <t< td=""><td>Bit erst 30 1111 111 111 11</td><td>257 mat 3 100 2642 255 100 2642 255 100 256 100 250 100 250 100 250 100 250 100 250 100 250 100 250 100 250 100 250 100 250 100 250 100 250 100 250 100 250 100 250 100 250 100 250 100 250 100 250
 100 250 100 250<td>123 See 1 Tot 0 See 1 Tot 0 See 1 S</td><td>132 Test 20 100 200 100 200 100 200 100<td>10 100</td><td>Hill team 11 12 <th12< th=""> 12 12</th12<></td><td>219 Xmm 20 1000 1000 1000 210 Xmm 20 1000 1000 1000 116 Xmm 20 2000 1000 1000 116 Xmm 20 2000 1000 1000 1 600 20 2000 2000 1000 1 600 20 2000 2000 1000 1 600 20 2000 2000 1000 1 600 20 2000 2000 1000 1 600 20 2000 2000 1000 10 600 2000 2000 2000 2000 2000 10 600 2000 2000 2000 2000 2000 2000 2000 10 600 2000 2000 2000 2000 2000 2000 2000</td></td></td></t<></th0.10<></td></td></tht<></td> | Lit Test Test <tht< td=""><td>122 press
123 press
124 press
125 press
125 press
126 press</td><td>Diff rest 2 1116 2010 1116 2010 1116 2010 <</td><td>115 State 2 2280 112 283 125 125 125 125 125 125 125 125 126 125 125 125 127 127 127 126 126 126 127</td><td>Hill was 30 2000 1000 200 1000 200 1000 200 1000</td><td>100 100
100<td>30.55 tem 2.5 0.105 1.10 1.05 0.10 1.05 0.10 0.15 0.10 0.15 0.10 0.10 0.15 0.10 <th0.10< th=""> 0.10 0.10 <t< td=""><td>Bit erst 30 1111 111 111 11</td><td>257 mat 3 100 2642 255 100 2642 255 100 256 100 250<td>123 See 1 Tot 0 See 1 Tot 0 See 1 S</td><td>132 Test 20 100 200 100 200 100 200 100<td>10 100</td><td>Hill team 11 12 <th12< th=""> 12 12</th12<></td><td>219 Xmm 20 1000 1000 1000 210 Xmm 20 1000 1000 1000 116 Xmm 20 2000 1000 1000 116 Xmm 20 2000 1000 1000 1 600 20 2000 2000 1000 1 600 20 2000 2000 1000 1 600 20 2000 2000 1000 1 600 20 2000 2000 1000 1 600 20 2000 2000 1000 10 600 2000 2000 2000 2000 2000 10 600 2000 2000 2000 2000 2000 2000 2000 10 600 2000 2000 2000 2000 2000 2000 2000</td></td></td></t<></th0.10<></td></td></tht<> | 122 press
123 press
124 press
125 press
125 press
126 press

 | Diff rest 2 1116 2010 1116 2010 1116 2010 < | 115 State 2 2280 112 283 125 125 125 125 125 125 125 125 126 125 125 125 127 127 127 126 126 126 127 | Hill was 30 2000 1000 200 1000 200 1000 200 1000 | 100 100
 100 100 <td>30.55 tem 2.5 0.105 1.10 1.05 0.10 1.05 0.10 0.15 0.10 0.15 0.10 0.10 0.15 0.10 <th0.10< th=""> 0.10 0.10 <t< td=""><td>Bit erst 30 1111 111 111 11</td><td>257 mat 3 100 2642 255 100 2642 255 100 256 100 250<td>123 See 1 Tot 0 See 1 Tot 0 See 1 S</td><td>132 Test 20 100 200 100 200 100 200 100<td>10 100</td><td>Hill team 11 12 <th12< th=""> 12 12</th12<></td><td>219 Xmm 20 1000 1000 1000 210 Xmm 20 1000 1000 1000 116 Xmm 20 2000 1000 1000 116 Xmm 20 2000 1000 1000 1 600 20 2000 2000 1000 1 600 20 2000 2000 1000 1 600 20 2000 2000 1000 1 600 20 2000 2000 1000 1 600 20 2000 2000 1000 10 600 2000 2000 2000 2000 2000 10 600 2000 2000 2000 2000 2000 2000 2000 10 600 2000 2000 2000 2000 2000 2000 2000</td></td></td></t<></th0.10<></td> | 30.55 tem 2.5 0.105 1.10 1.05 0.10 1.05 0.10 0.15 0.10 0.15 0.10 0.10 0.15 0.10 <th0.10< th=""> 0.10 0.10 <t< td=""><td>Bit erst 30 111 111 111 111 111 111 111 111 111 111 111 111 111 111
 111 1111 111 111 11</td><td>257 mat 3 100 2642 255 100 2642 255 100 256 100 250<td>123 See 1 Tot 0 See 1 Tot 0 See 1 S</td><td>132 Test 20 100 200 100 200 100 200 100<td>10 100</td><td>Hill team 11 12 <th12< th=""> 12 12</th12<></td><td>219 Xmm 20 1000 1000 1000 210 Xmm 20 1000 1000 1000 116 Xmm 20 2000 1000 1000 116 Xmm 20 2000 1000 1000 1 600 20 2000 2000 1000 1 600 20 2000 2000 1000 1 600 20 2000 2000 1000 1 600 20 2000 2000 1000 1 600 20 2000 2000 1000 10 600 2000 2000 2000 2000 2000 10 600 2000 2000 2000 2000 2000 2000 2000 10 600 2000 2000 2000 2000 2000 2000 2000</td></td></td></t<></th0.10<> | Bit erst 30 1111 111 111 11
 | 257 mat 3 100 2642 255 100 2642 255 100 256 100 250 <td>123 See 1 Tot 0 See 1 Tot 0 See 1 S</td> <td>132 Test 20 100 200 100 200 100 200 100<td>10 100</td><td>Hill team 11 12 <th12< th=""> 12 12</th12<></td><td>219 Xmm 20 1000 1000 1000 210 Xmm 20 1000 1000 1000 116 Xmm 20 2000 1000 1000 116 Xmm 20 2000 1000 1000 1 600 20 2000 2000 1000 1 600 20 2000 2000 1000 1 600 20 2000
 2000 1000 1 600 20 2000 2000 1000 1 600 20 2000 2000 1000 10 600 2000 2000 2000 2000 2000 10 600 2000 2000 2000 2000 2000 2000 2000 10 600 2000 2000 2000 2000 2000 2000 2000</td></td> | 123 See 1 Tot 0 See 1 Tot 0 See 1 S | 132 Test 20 100 200 100 200 100 200 100 <td>10 100</td> <td>Hill team 11 12 <th12< th=""> 12 12</th12<></td> <td>219 Xmm 20 1000 1000 1000 210 Xmm 20 1000 1000 1000 116 Xmm 20 2000 1000 1000 116 Xmm 20 2000 1000 1000 1 600 20 2000 2000 1000 1 600 20 2000 2000 1000 1 600 20 2000 2000 1000 1 600 20 2000 2000 1000 1 600 20 2000 2000 1000 10 600 2000 2000 2000 2000 2000 10 600 2000 2000 2000 2000 2000 2000 2000 10 600 2000 2000 2000 2000 2000 2000 2000</td> | 10 100 | Hill team 11 12 <th12< th=""> 12 12</th12<>
 | 219 Xmm 20 1000 1000 1000 210 Xmm 20 1000 1000 1000 116 Xmm 20 2000 1000 1000 116 Xmm 20 2000 1000 1000 1 600 20 2000 2000 1000 1 600 20 2000 2000 1000 1 600 20 2000 2000 1000 1 600 20 2000 2000 1000 1 600 20 2000 2000 1000 10 600 2000 2000 2000 2000 2000 10 600 2000 2000 2000 2000 2000 2000 2000 10 600 2000 2000 2000 2000 2000 2000 2000 |
| |

 |
 |

 | | | 1 1 1 1 2 2 2 3 3 2 4 7 1 2 2 4 7 1
2 4 7 1 2 |
 |
 | |
 | |
 | | 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | |
| | 101 000 110 000 <td>Lot Total T</td> <td>100 100<td>Up: The set of the set of</td><td></td><td>$\begin{array}{c c c c c c c c c c c c c c c c c c c$</td><td>Lab res 21 110 2011 2113 211 2113<</td><td>$\begin{array}{cccccccccccccccccccccccccccccccccccc$</td><td>State Description Secretion Description Descripacription Description</td><td>$\begin{array}{c} \begin{array}{c} \begin{array}{c} \begin{array}{c} \\ \end{array} \\ \end{array} \\ \begin{array}{c} \end{array} \\ \begin{array}{c} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \end{array} \\ \end{array} \\ \end{array} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \end{array} \\ \end{array} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\$</td><td>Non-State 2 21200 1121 1233 1240</td><td>State Diff Table of the state Diff Table of the state Diff Diff</td><td>Implement Strephysics Strephysics</td><td>$\begin{array}{cccccccccccccccccccccccccccccccccccc$</td><td>160 70 7 100 200 700 160 700 70 100 200 700 117 700 200 200 200 200 200 117 700 200</td></td>
 | Lot Total T
 | 100 100 <td>Up: The set of the set of</td> <td></td> <td>$\begin{array}{c c c c c c c c c c c c c c c c c c c$</td> <td>Lab res 21 110 2011 2113 211 2113
2113 2113<</td> <td>$\begin{array}{cccccccccccccccccccccccccccccccccccc$</td> <td>State Description Secretion Description Descripacription Description</td> <td>$\begin{array}{c} \begin{array}{c} \begin{array}{c} \begin{array}{c} \\ \end{array} \\ \end{array} \\ \begin{array}{c} \end{array} \\ \begin{array}{c} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \end{array} \\ \end{array} \\ \end{array} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \end{array} \\ \end{array} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\$</td> <td>Non-State 2 21200 1121 1233 1240</td> <td>State Diff Table of the state Diff Table of the state Diff Diff</td> <td>Implement Strephysics Strephysics</td> <td>$\begin{array}{cccccccccccccccccccccccccccccccccccc$</td> <td>160 70 7 100 200 700 160 700 70 100 200 700 117 700 200 200 200 200 200 117 700 200</td> | Up: The set of |
 | $\begin{array}{c c c c c c c c c c c c c c c c c c c $ | Lab res 21 110 2011 2113 211 2113<
 | $\begin{array}{cccccccccccccccccccccccccccccccccccc$
 | State Description Secretion Description Descripacription Description | $\begin{array}{c} \begin{array}{c} \begin{array}{c} \begin{array}{c} \\ \end{array} \\ \end{array} \\ \begin{array}{c} \end{array} \\ \begin{array}{c} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\
\end{array} \\ \begin{array}{c} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \end{array} \\ \end{array} \\ \end{array} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \end{array} \\ \end{array} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ $ | Non-State 2 21200 1121 1233 1240 | State Diff Table of the state Diff Table of the state Diff
 | Implement Strephysics | $ \begin{array}{cccccccccccccccccccccccccccccccccccc$ | 160 70 7 100 200 700 160 700 70 100 200 700 117 700 200 200 200 200 200 117 700 200 |



- Finals will take place on August 2016
 DEFCON, Las Vegas
- Money prices!
 - First place: \$2'000'000
 - Second place: \$1'000'000
 - Third place: \$750'000
- The winning team will compete against human teams at DEFCON CTF Finals :-)

Shellphish CGC Team





I want more...



- **angr** hands-on workshop
 - Just after this talk
 - Hall 13 (first floor)
 - Bring your laptop!





Questions?

References:

CGC: <u>cybergrandchallenge.org</u> – <u>cgc.darpa.mil</u>

DARPA CGC presentation (DEFCON 2015): youtu.be/gnyCbU7jGYA

angr: <u>angr.io</u> – <u>github.com/angr</u>

emails: antoniob@cs.ucsb.edu – jacopo@cs.ucsb.edu – dutcher@cs.ucsb.edu