Automatic Binary Exploitation and Patching using Mechanical [Shell]Phish

Antonio Bianchi antoniob@cs.ucsb.edu

University of California, Santa Barbara (UCSB)

HITCON Pacific
December 2nd, 2016

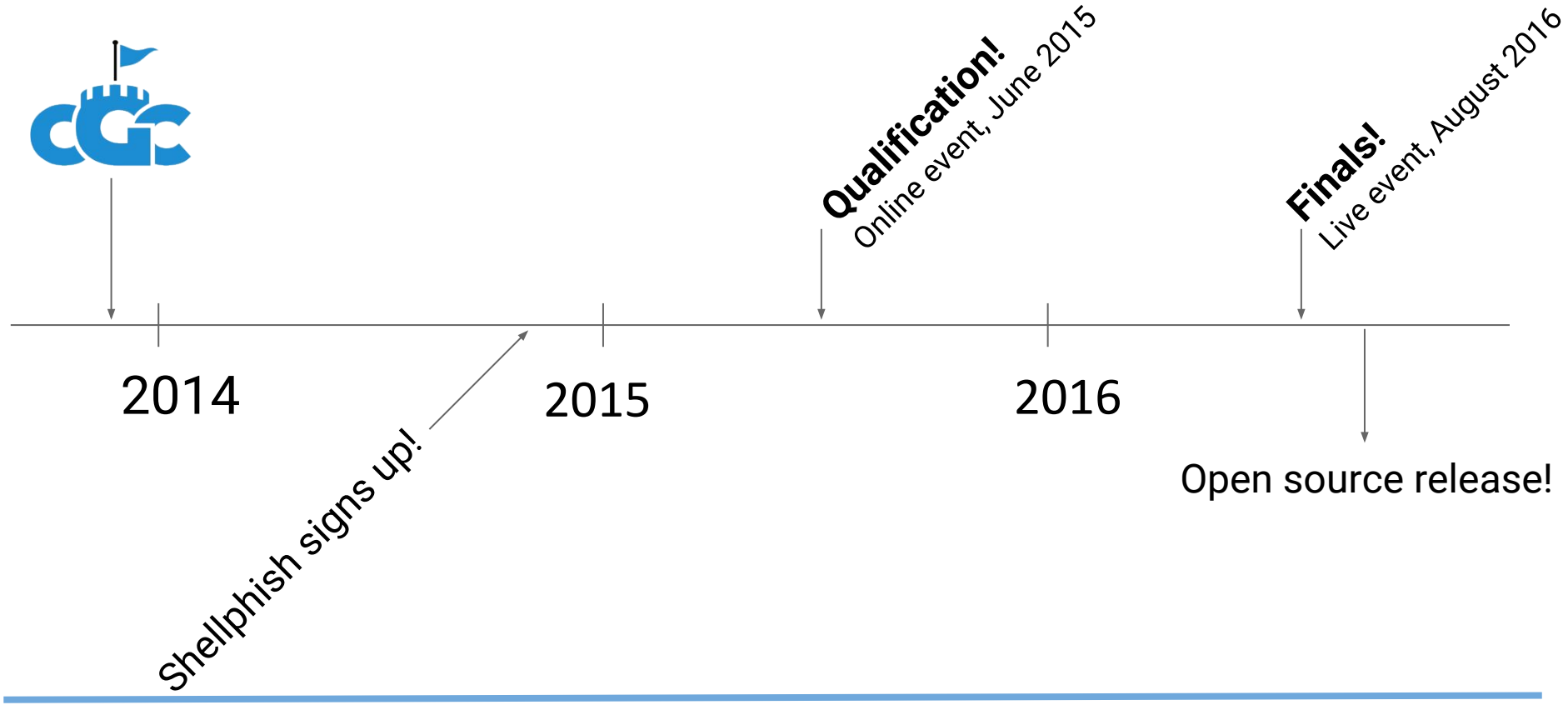
Shellphish

- A team of security enthusiasts
  - Do research in system security
  - Play security competitions (CTF)
  - Mainly students from University of California, Santa Barbara
  - More info:
    - “A Dozen Years of Shellphish” https://youtu.be/APY2SsBde1U

CGC

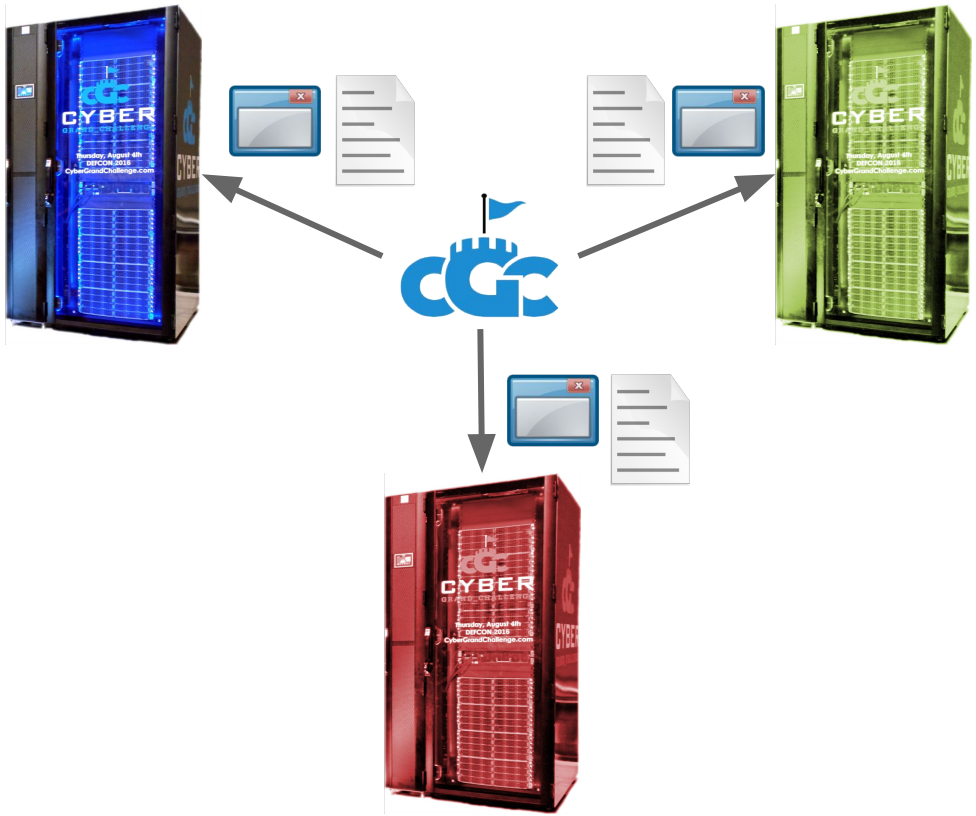
- A fully automated CTF competition
- Organized by DARPA
- No Human intervention

CGC – Timeline



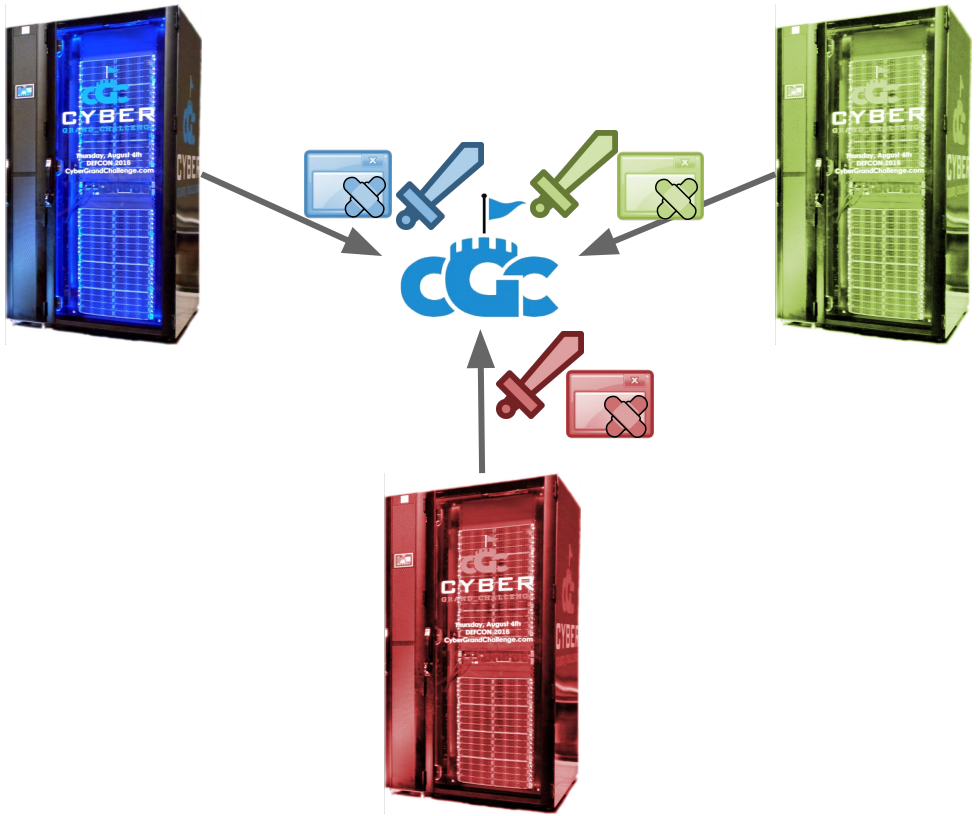

CYBER
GRAND CHALLENGE
DARPA
CYBER
DARPA
CYBER
DARPA
CYBER
DARPA
CYBER
SCORING
INTEGRITY
MECHANICAL PHISH
MAYHEM
SECURE
SECURE

CGC – How the game worked



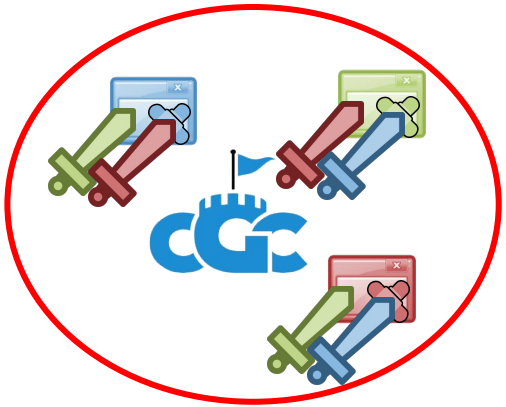
- Round-based game
- Organizers' servers provide:
  - Binaries
    - Linux-like, Intel x86, limited syscalls
    - Console (stdin/stdout)
    - Compiled C programs → no source code
    - Contain one or more vulnerabilities
  - Network traffic
    - Collected during previous rounds

CGC – How the game worked



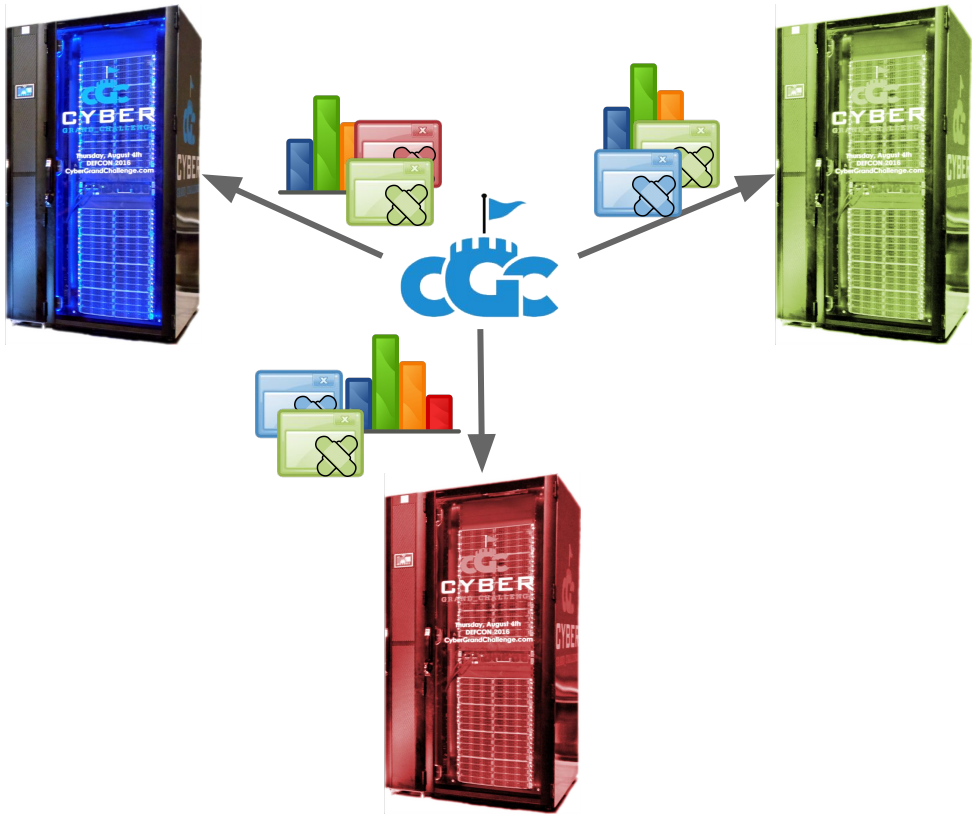
- Teams provide
  - Patched binaries
  - Attacks

CGC – How the game worked



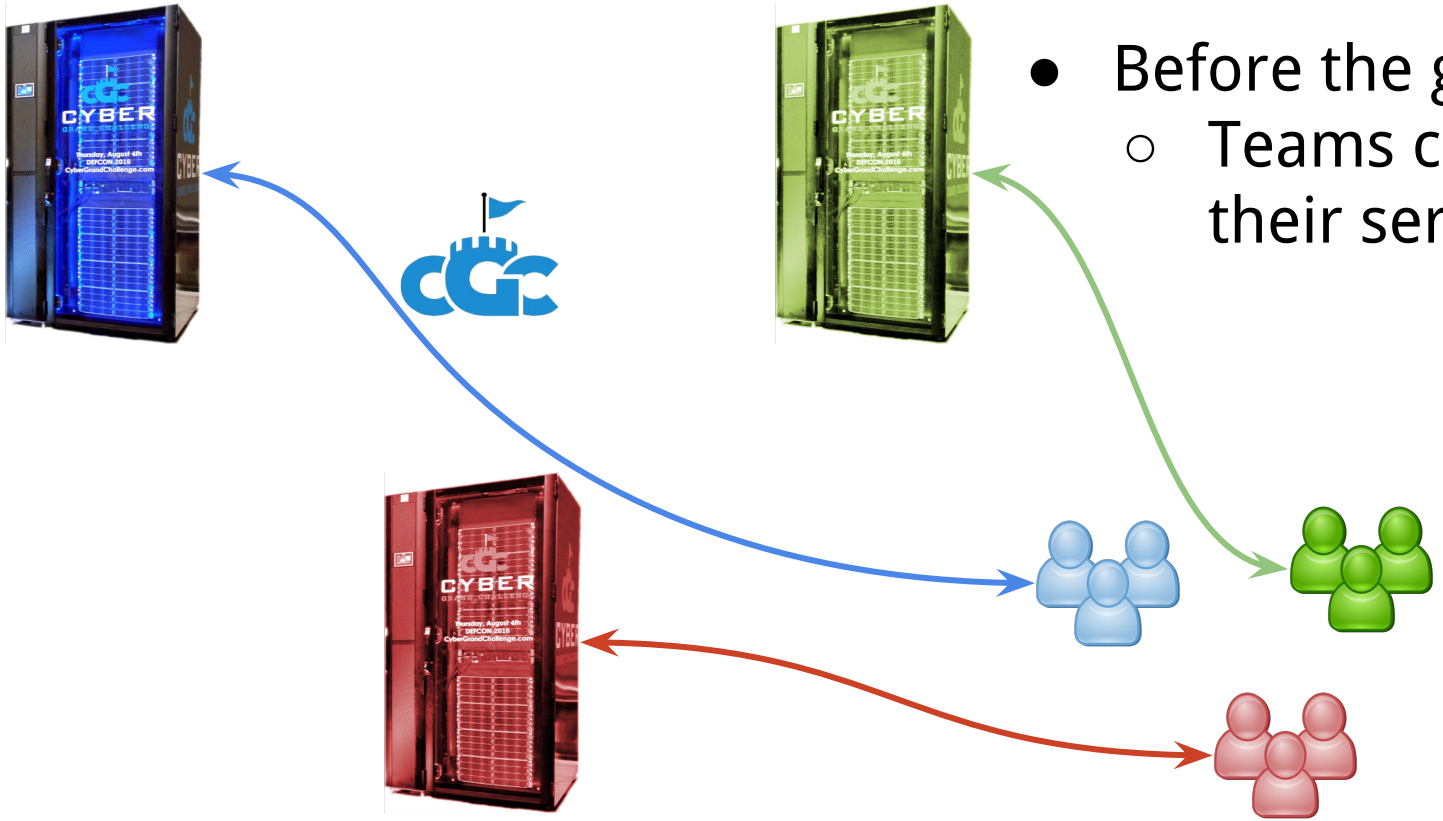
- Organizers's servers evaluate
  - Attacks
    vs.
    Patched binary

CGC – How the game worked



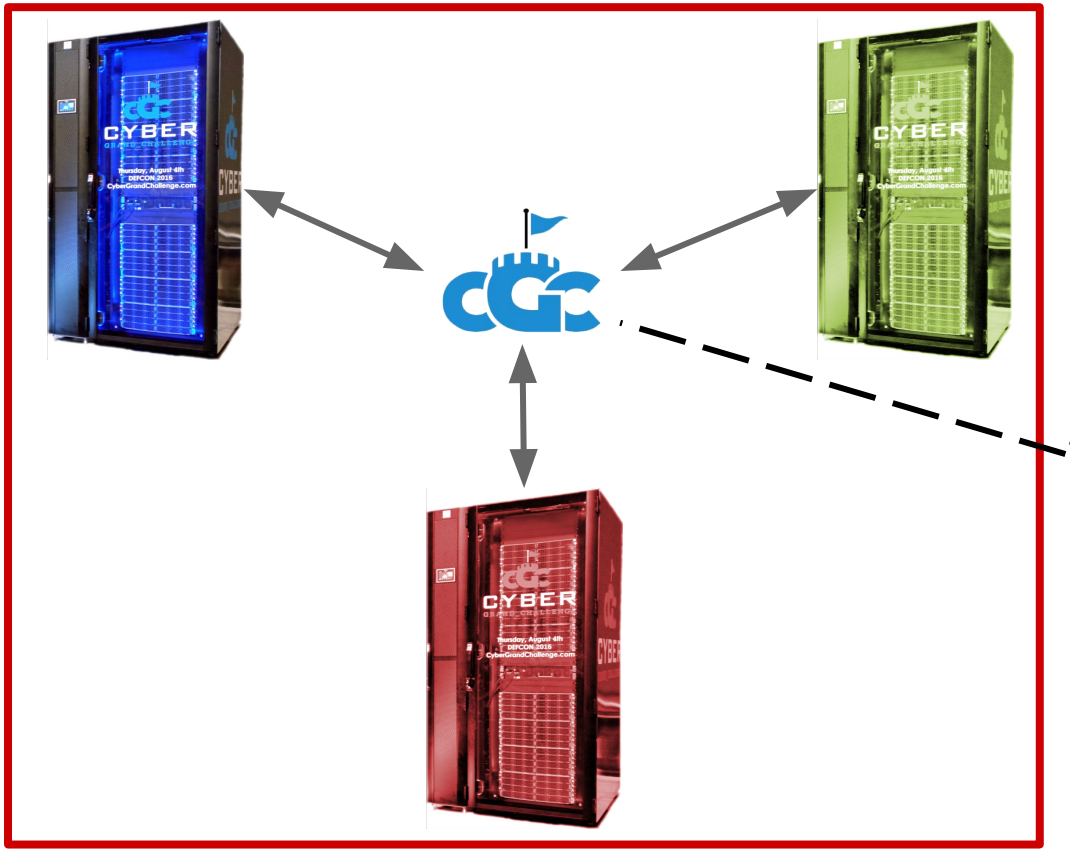
- Organizers' servers send back
  - Scores
  - Patched binaries from adversarial teams

CGC – How the game worked



- Before the game
  - Teams can set up their servers

CGC – How the game worked



- During the game (10h)
  - "Certified air gap"
  - Scores are the only data exiting


MIND THE
AIR GAP
DATA OUT
POWER
DARPA

How to play?

Automatic Binary Exploitation

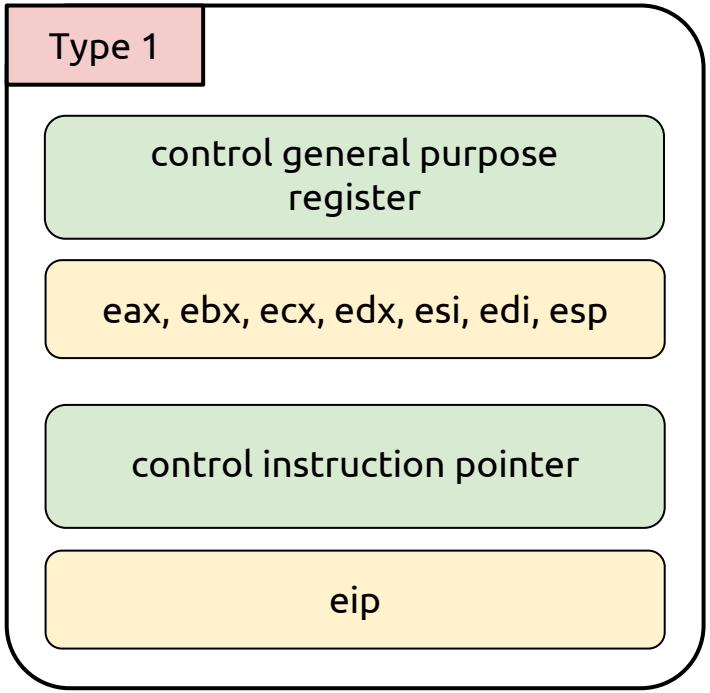
Automatic Binary Patching

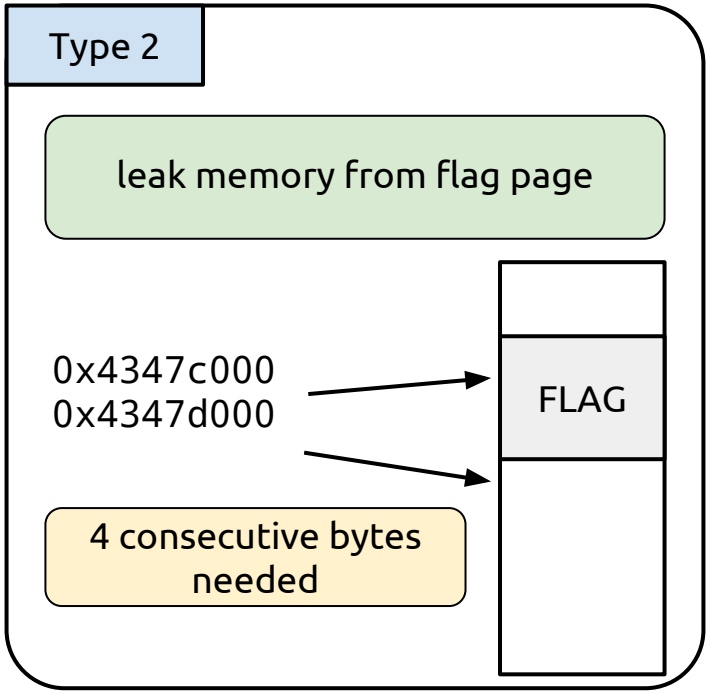
Infrastructure

How to play?

Automatic Binary Exploitation

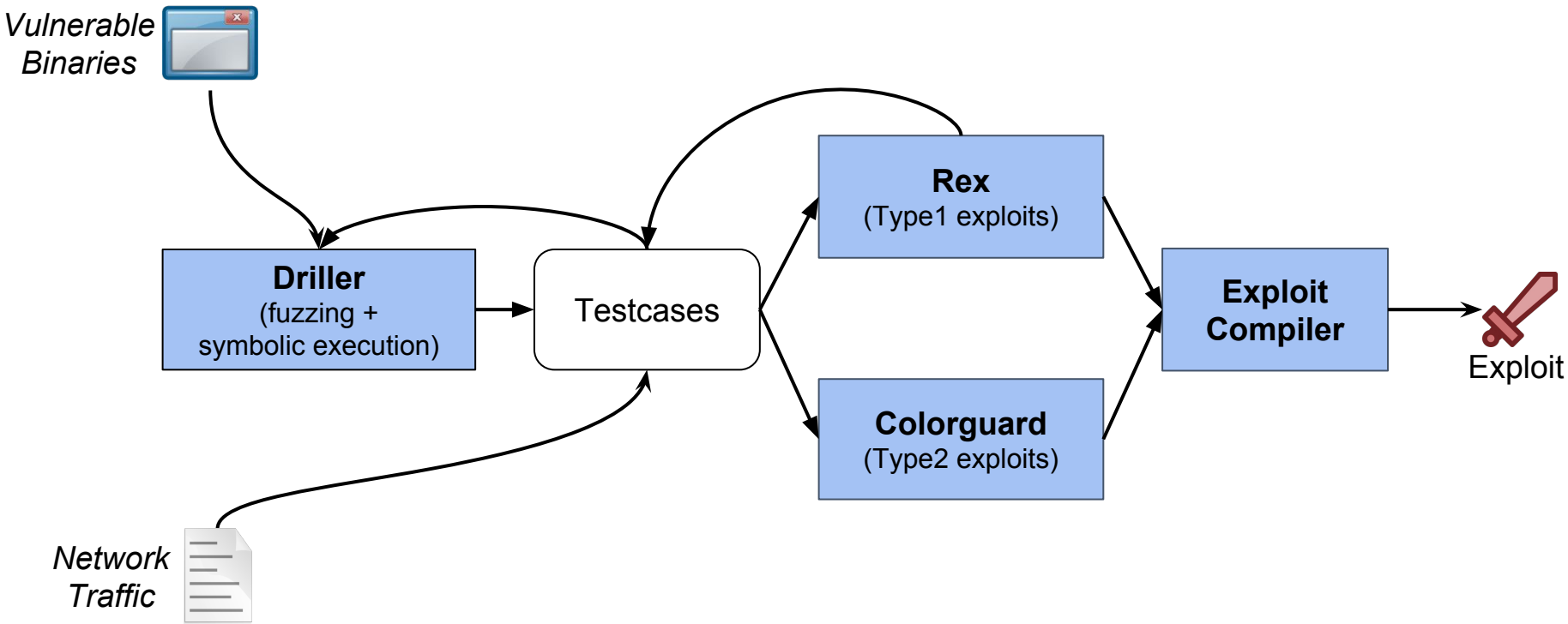
Two types of exploits





- Arbitrary code execution
- Information leak
  (Heartbleed, ASLR base address leak, ...)

Exploitation pipeline (simplified)



Driller

- Execute “the most” of the program →
  Find good inputs to the binary

```
v1 = user_input1()
v2 = user_input2()

if(v1 < 10){
  if (v1 == 3){
    foo()
  }else if(v1 == 7){
    bar()
 }
}else{
 if((v1^2 - 19087925*v1)==57263784){
   function_pointer = v2 + 300
   function_pointer()
 }
}
```

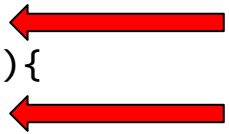
Try many different inputs:
“1”, “2”, “3”, “4”, “5”, “7”, “8”, …

Driller

- Execute “the most” of the program → Find good inputs to the binary

```
v1 = user_input1()
v2 = user_input2()

if(v1 < 10){
  if (v1 == 3){
    foo()
  }else if(v1 == 7){
    bar()
 }
}else{
 if((v1^2 - 19087925*v1)==57263784){
   function_pointer = v2 + 300
   function_pointer()
 }
}
```



“3” and “7” are “good” testcases: they reach new code locations

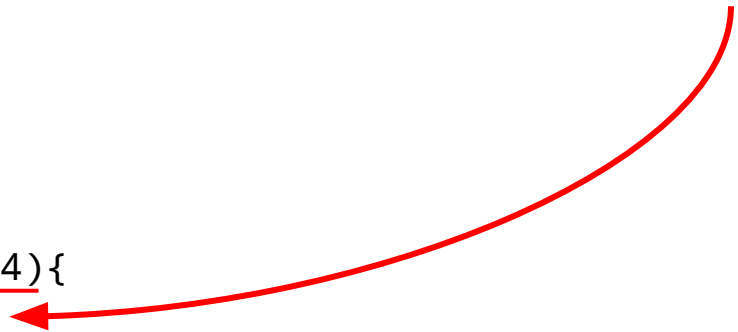
Driller

- Execute “the most” of the program → Find good inputs to the binary

```
v1 = user_input1()
v2 = user_input2()

if(v1 < 10){
  if (v1 == 3){
    foo()
  }else if(v1 == 7){
    bar()
 }
}else{
 if((v1^2 - 19087925*v1)==57263784){
   function_pointer = v2 + 300
   function_pointer()
 }
}
```

This is hard to reach randomly

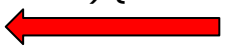


Driller

- Execute “the most” of the program → Find good inputs to the binary

```
v1 = user_input1()
v2 = user_input2()

if(v1 < 10){
  if (v1 == 3){
    foo()
  }else if(v1 == 7){
    bar()
 }
}else{
 if((v1^2 - 19087925*v1)==57263784){
   function_pointer = v2 + 300
   function_pointer()
 }
}
```



We can use “symbolic tracing”

Driller

- Execute “the most” of the program → Find good inputs to the binary

```
v1 = user_input1()
v2 = user_input2()

if(v1 < 10){
  if (v1 == 3){
    foo()
  }else if(v1 == 7){
    bar()
 }
}else{
 if((v1^2 - 19087925*v1)==57263784){
   function_pointer = v2 + 300
   function_pointer()
 }
}
```

Target

We can use “symbolic tracing”

Contraints:

- v1 = user_input1()

Driller

- Execute “the most” of the program →
  Find good inputs to the binary

```
v1 = user_input1()
v2 = user_input2()

if(v1 < 10){
  if (v1 == 3){
    foo()
  }else if(v1 == 7){
    bar()
 }
}else{
 if((v1^2 - 19087925*v1)==57263784){
   function_pointer = v2 + 300
   function_pointer()
 }
}
```

Target

We can use “symbolic tracing”

Contraints:

- v1 = user_input1
- not (v1 < 10)

Driller

- Execute “the most” of the program → Find good inputs to the binary

```
v1 = user_input1()
v2 = user_input2()

if(v1 < 10){
  if (v1 == 3){
    foo()
  }else if(v1 == 7){
    bar()
 }
}else{
 if((v1^2 - 19087925*v1)==57263784){
   function_pointer = v2 + 300
   function_pointer()
 }
}
```

Target

We can use “symbolic tracing”

Contraints:

- v1 = user_input1()
- not (v1 < 10)
- v1^2 - 19087925*v1 == 57263784

Driller

- Execute “the most” of the program → Find good inputs to the binary

```
v1 = user_input1()
v2 = user_input2()

if(v1 < 10){
  if (v1 == 3){
    foo()
  }else if(v1 == 7){
    bar()
 }
}else{
 if((v1^2 - 19087925*v1)==57263784){
   function_pointer = v2 + 300
   function_pointer()
 }
}
```

Target

We can use “symbolic tracing”

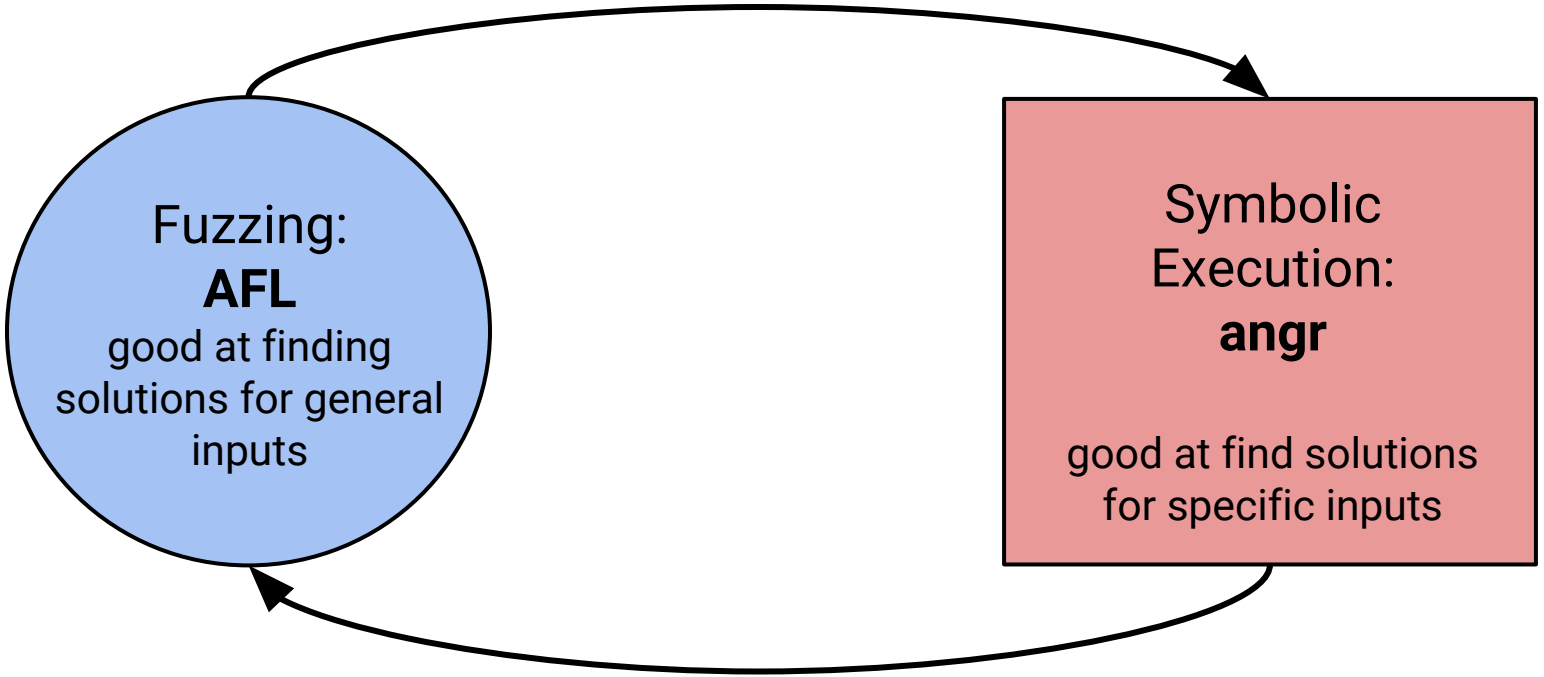
Contraints:

- v1 = user_input1()
- not (v1 < 10)
- v1^2 - 19087925*v1 == 57263784

Solve:

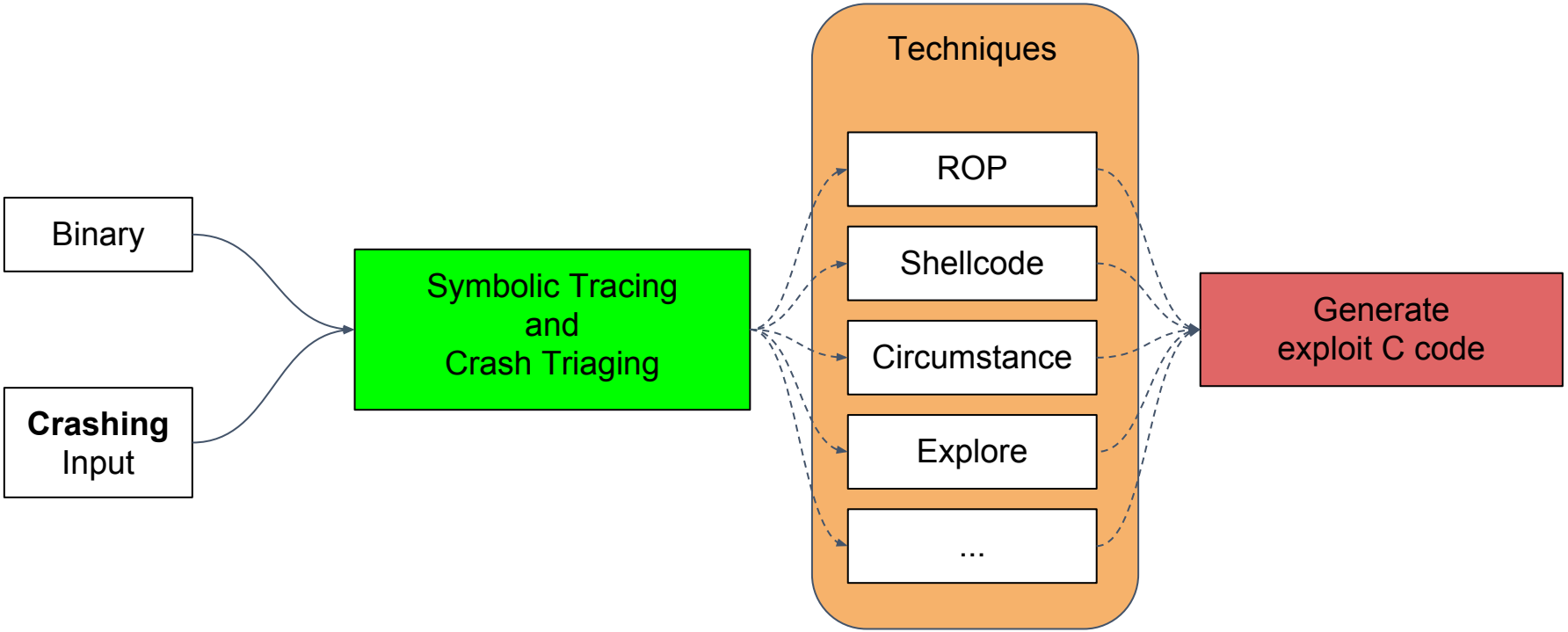
- user_input1 = 0x1234238

Driller – AFL + angr

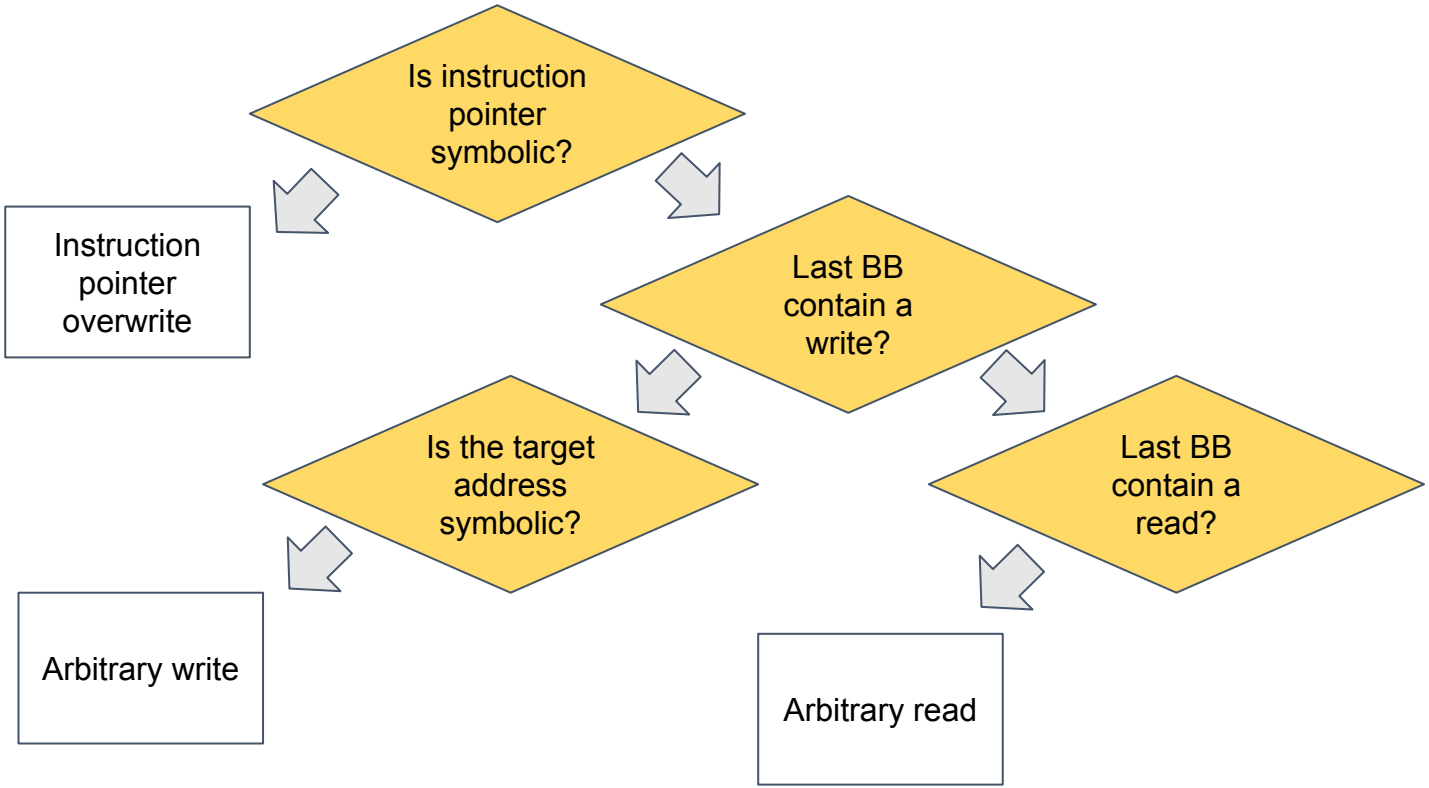


Driller: Augmenting fuzzing through selective symbolic execution.
N. Stephens, J. Grosen, C. Salls, A. Dutcher, R. Wang, J. Corbetta, Y. Shoshitaishvili, C. Kruegel, and G. Vigna.
at NDSS 2016

Rex


Techniques
ROP
Shellcode
Circumstance
Explore
...
Binary
Crashing Input
Symbolic Tracing and Crash Triaging
Generate exploit C code

Rex – Crash Triaging



Rex – Symbolic Tracing

- Understand “how” to control the crash

```
v1 = user_input1()
v2 = user_input2()

if(v1 > 10){
  if (v1 == 3){
    foo()
  }else if(v1 == 7){
    bar()
 }
}else{
 if((v1^2 - 19087925*v1)==57263784){
   function_pointer = v2 + 300
   function_pointer()
 }
}
```

If this instruction is reached the program crashes

Rex – Symbolic Tracing

- Understand “how” to control the crash

```
v1 = user_input1()
v2 = user_input2()

if(v1 > 10){
  if (v1 == 3){
    foo()
  }else if(v1 == 7){
    bar()
 }
}else{
 if((v1^2 - 19087925*v1)==57263784){
   function_pointer = v2 + 300
   function_pointer()
 }
}
```

Using symbolic tracing, we know:

instruction_pointer =
function_pointer =
v2 + 300 =
user_input2 + 300

Rex – Symbolic Tracing

- Understand “how” to control the crash

```
v1 = user_input1()
v2 = user_input2()

if(v1 > 10){
  if (v1 == 3){
    foo()
  }else if(v1 == 7){
    bar()
 }
}else{
 if((v1^2 - 19087925*v1)==57263784){
   function_pointer = v2 + 300
   function_pointer()
 }
}
```

Using symbolic tracing, we know:

instruction_pointer = user_input2 + 300

Therefore:

- By controlling the user input we control the instruction pointer

Rex – Symbolic Tracing

- Understand “how” to control the crash

```
v1 = user_input1()
v2 = user_input2()

if(v1 > 10){
  if (v1 == 3){
    foo()
  }else if(v1 == 7){
    bar()
 }
}else{
 if((v1^2 - 19087925*v1)==57263784){
   function_pointer = v2 + 300
   function_pointer()
 }
}
```

Using symbolic tracing, we know:

instruction_pointer = user_input2 + 300

Therefore:

- By controlling the user input we control the instruction pointer
- If we want:

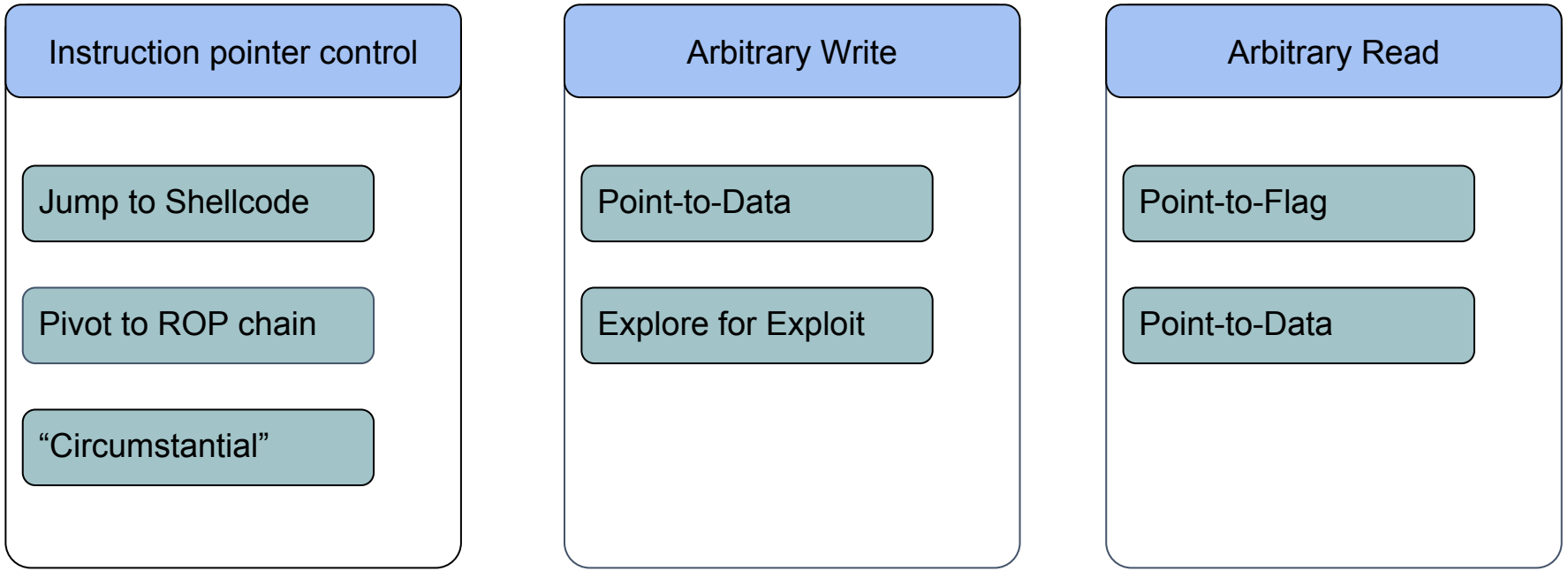
  instruction_pointer = X

  we have to set:

  user_input2 = X - 300

Rex – Techniques

- Crashing input → Exploit



Rex – Technique: Jump to Shellcode

- We want to place shellcode in `buffer` and jump to it

```
v1 = user_input1()
buffer = base64_decode(user_input2())

//…

function_pointer = v1 + 300
function_pointer()
```

Rex – Technique: Jump to Shellcode

- We want to place shellcode in `buffer` and jump to it

```
v1 = user_input1()
buffer = base64_decode(user_input2())

//…

function_pointer = v1 + 300
function_pointer()
```

Using symbolic tracing, we know:

instruction_pointer = user_input1 + 300

buffer = base64_decode(user_input2)

Rex – Technique: Jump to Shellcode

- We want to place shellcode in `buffer` and jump to it

```
v1 = user_input1()
buffer = base64_decode(user_input2())

//…

function_pointer = v1 + 300
function_pointer()
```

Using symbolic tracing, we know:

instruction_pointer = user_input1 + 300
buffer = base64_decode(user_input2)

We want:

instruction_pointer = &(buffer)
buffer = shellcode

Rex – Technique: Jump to Shellcode

- We want to place shellcode in `buffer` and jump to it

```
v1 = user_input1()
buffer = base64_decode(user_input2())

//…

function_pointer = v1 + 300
function_pointer()
```

Using symbolic tracing, we know:

instruction_pointer = user_input1 + 300
buffer = base64_decode(user_input2)

We want:

instruction_pointer = &(buffer)
buffer = shellcode

Therefore:

user_input1 = &(buffer) - 300
user_input2 = base64_encode(shellcode)

Colorguard

- Memory-leak (Type 2) exploits
- Use symbolic tracing
- Analyze all inputs

Colorguard

- Memory-leak (Type 2) exploits are also generated using symbolic tracing

```
v1 = user_input1()

//…

printed_value = array[v1]
print(printed_value)
```

Using symbolic tracing, we know:

printed_value = *(&array + v1)

v1 = user_input1

We want:

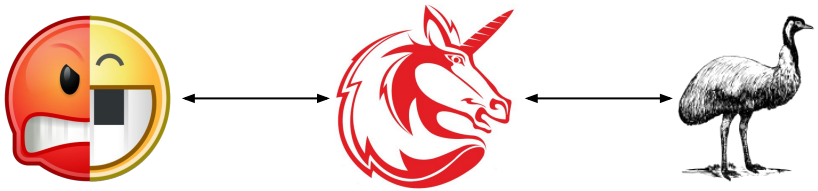
printed_value = flag_page[0]

Therefore:

user_input1 = (&flag_page) - (&array)

Colorguard – Unicorn Engine

- Every testcase can potentially leak the flag page
- Full symbolic tracing of every testcase is too slow
- angr + unicorn engine (QEMU wrapper)
  - Execute “most” of the code in QEMU



angr **unicorn engine** **QEMU**

3,000,000 times slower than a real CPU

2~5 times slower than a real CPU

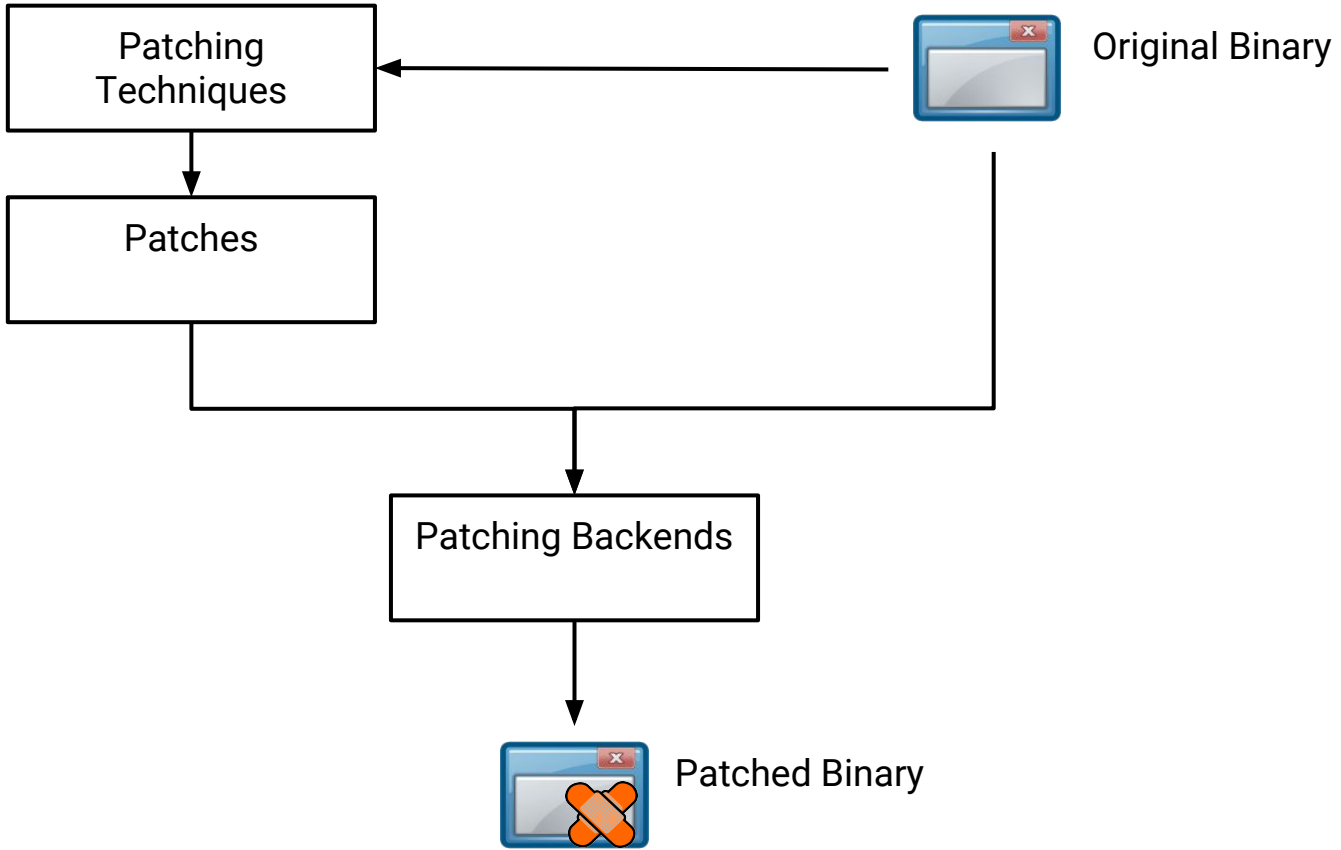
How to play?

Automatic Binary Patching

Automatic Patching

- Prevent binary form being exploit
- Preserve binary functionality
- Preserve binary performance
  - speed
  - memory usage
  - disk space
- Prevent analysis from other teams

Patcherex


Patching Techniques
Original Binary
Patches
Patching Backends
Patched Binary

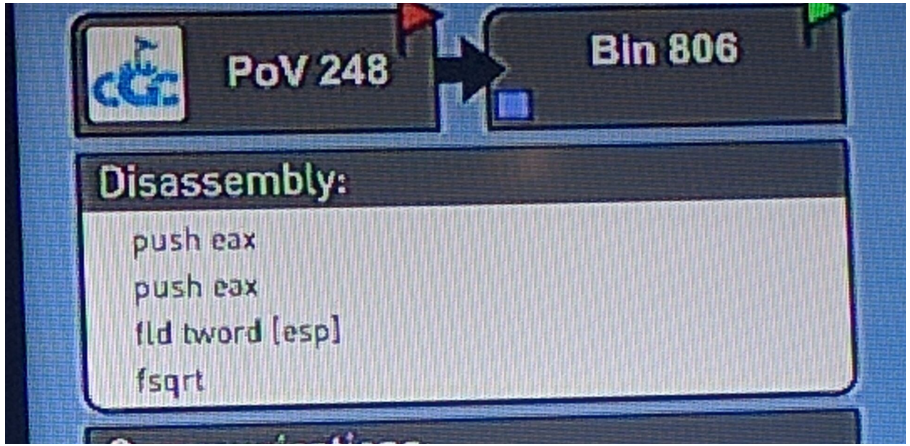
Patcherex

- Defensive Techniques
  - Return pointer encryption
  - Protect indirect calls/jmps
  - Extended Malloc allocations
  - Randomly shift the stack (ASLR)
  - ...

Patcherex

- Adversarial Techniques
  - Detect QEMU

```
mov eax, 0x1
push eax
push eax
push eax
fld TBYTE PTR [esp]
fsqrt
```



  - Backdoor
  - ...

Patcherex

- Making the original binary faster →

  Our patches can be slower!

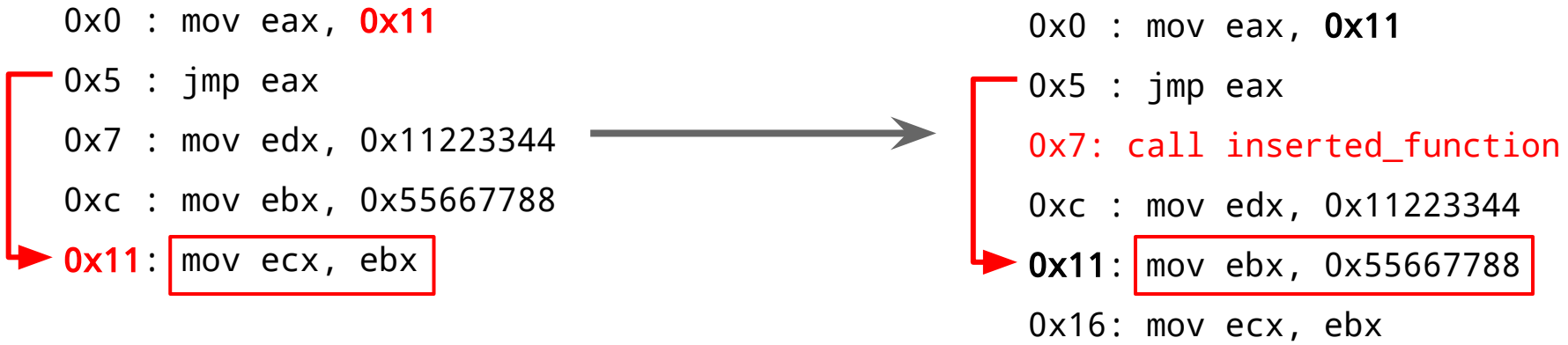
- Optimization Techniques:
  - Constant Propagation
  - Dead Assignment Elimination
  - ...

Patcherex – Backends

- Patching Backends
  - Inject code/data in an existing binary
  - No source code
  - No symbols

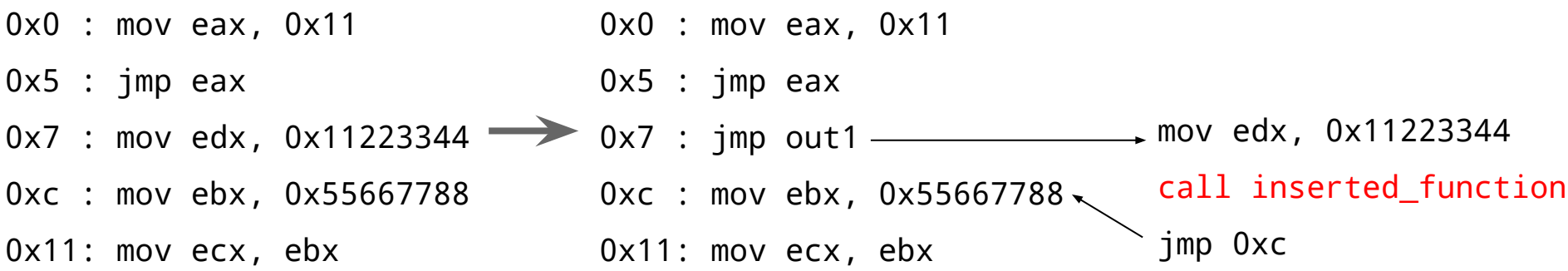
Patcherex – Backends

- How to inject code without breaking functionality?



Patcherex – Backends

- Detour Backend
  - Try to add code without moving the original one
  - Not always possible
  - Slow (requires a lot of additional jmp instructions)

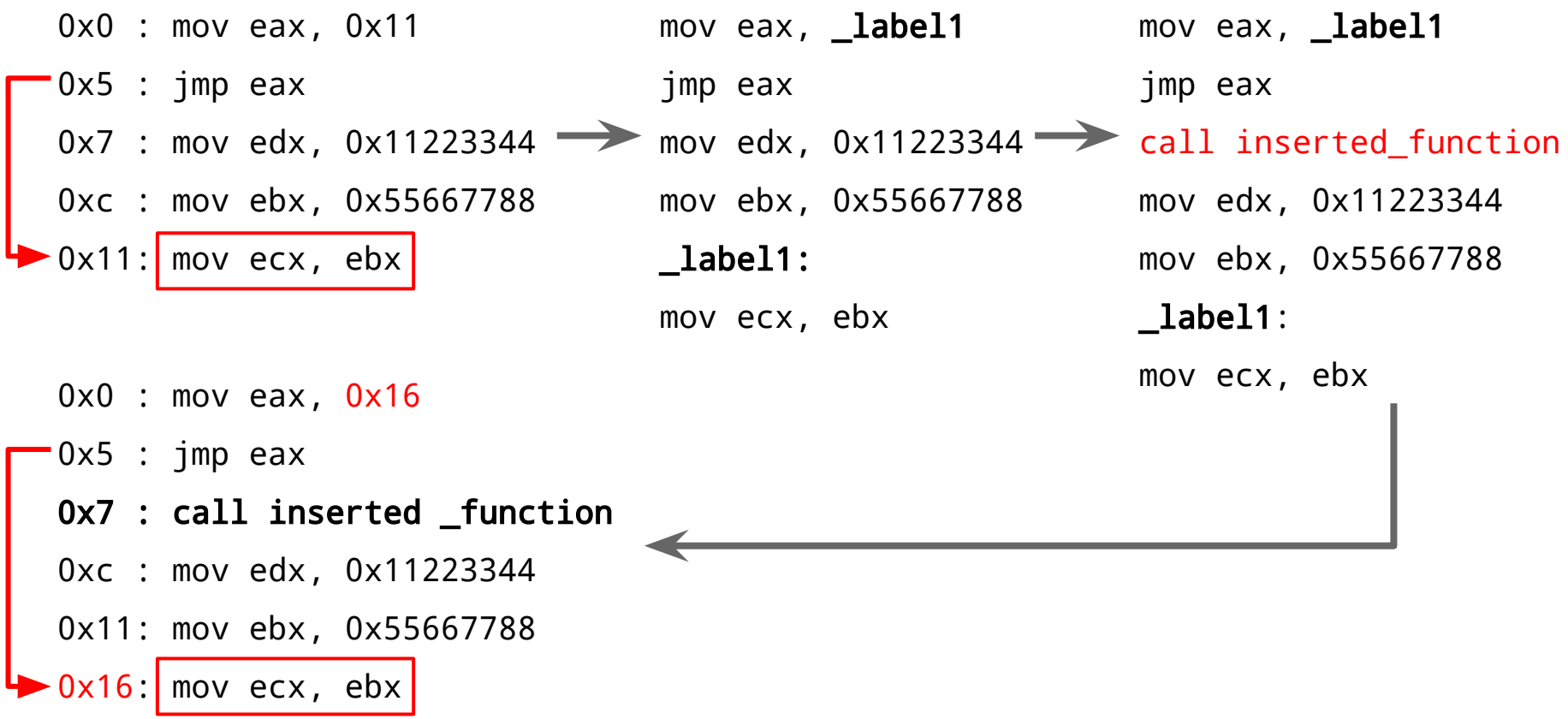


Patcherex – Backends

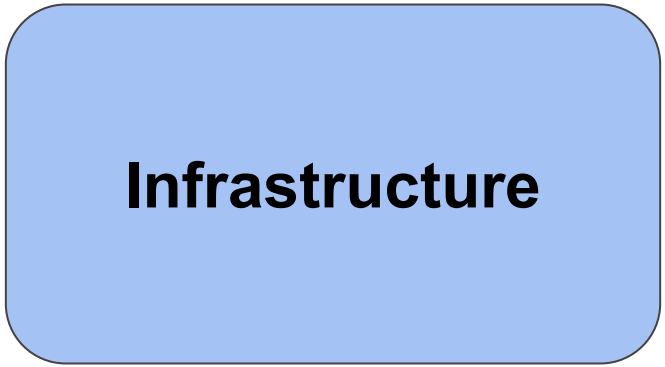
- Reassemler Backend
  - Recover original “program symbols”
  - More efficient code
  - (Slightly) less reliable

Ramblr: Making Reassembly Great Again.
R. Wang, Y. Shoshitaishvili, A. Bianchi, A. Machiry, J. Grosen, P. Grosen, C. Kruegel, G. Vigna
In NDSS 2017

Patcherex – Backends



How to play?



Infrastructure

- Our code had to run for 10 hours on:

  64 servers, 16TB of RAM, 2560 cores
- No human intervention →

  No possibility of failure!
- Extremely hard to test the full system
  - A lot of test cases
  - Testing after *every* single git push

Infrastructure

- Separate and (mostly) independent *tasks*

- Every task run in a separate container
  - Docker
- Tasks are distributed “transparently” among servers
  - Kubernetes

What Happened?

Results

- Exploitation
  - 2442 Exploits generated
  - 1709 Exploits for 14/82 challenges with 100% Reliability
  - Longest exploit: 3791 lines of C code
  - crackaddr: 517 lines of C code
  - **Shellphish exploited the most binaries!**
- Defense
  - Only 12/82 services were compromised
  - Second best team in terms of defense points

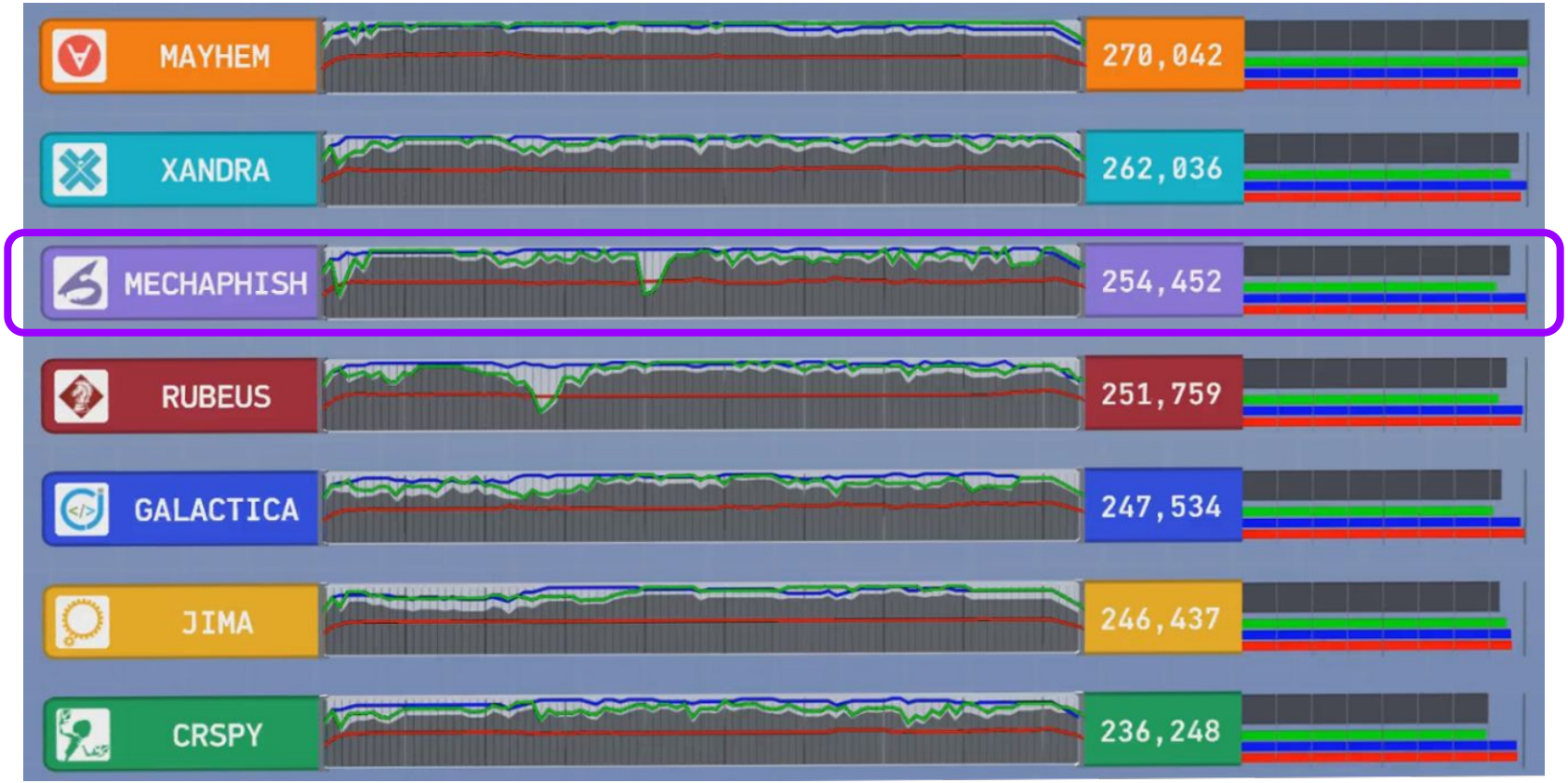
Third Place! ::\OwO/::

- **Third Place!**

- Happiness!

- First among University-only teams
- First among unfunded teams

Results


MAYHEM
270,042
XANDRA
262,036
MECHAPHISH
254,452
RUBEUS
251,759
GALACTICA
247,534
JIMA
246,437
CRSPY
236,248

$$$

750,000 $ +

750,000 $ =

1,500,000 $

What went wrong

- Our strategy was not ideal ~~ 囧
  - Patch everything!
  - Score penalty
  - Only 20/82 binaries were exploited in total

Open source release

- Open source all the code!

Open source release

- About 100,000 lines of Python code
- github.com/shellphish
  - Core, independent components: REX, Patcherex, ...
- github.com/mechaphish
  - Infrastructure, utilities, and documentation
- github.com/angr
  - Binary analysis framework, symbolic execution, ...

Standing on the shoulders of giants

AFL

QEMU
open source processor emulator

python

kubernetes

Unicorn
Engine

Z3

ubuntu

pypy

peewee

PostgreSQL

VEX

Capstone
Engine

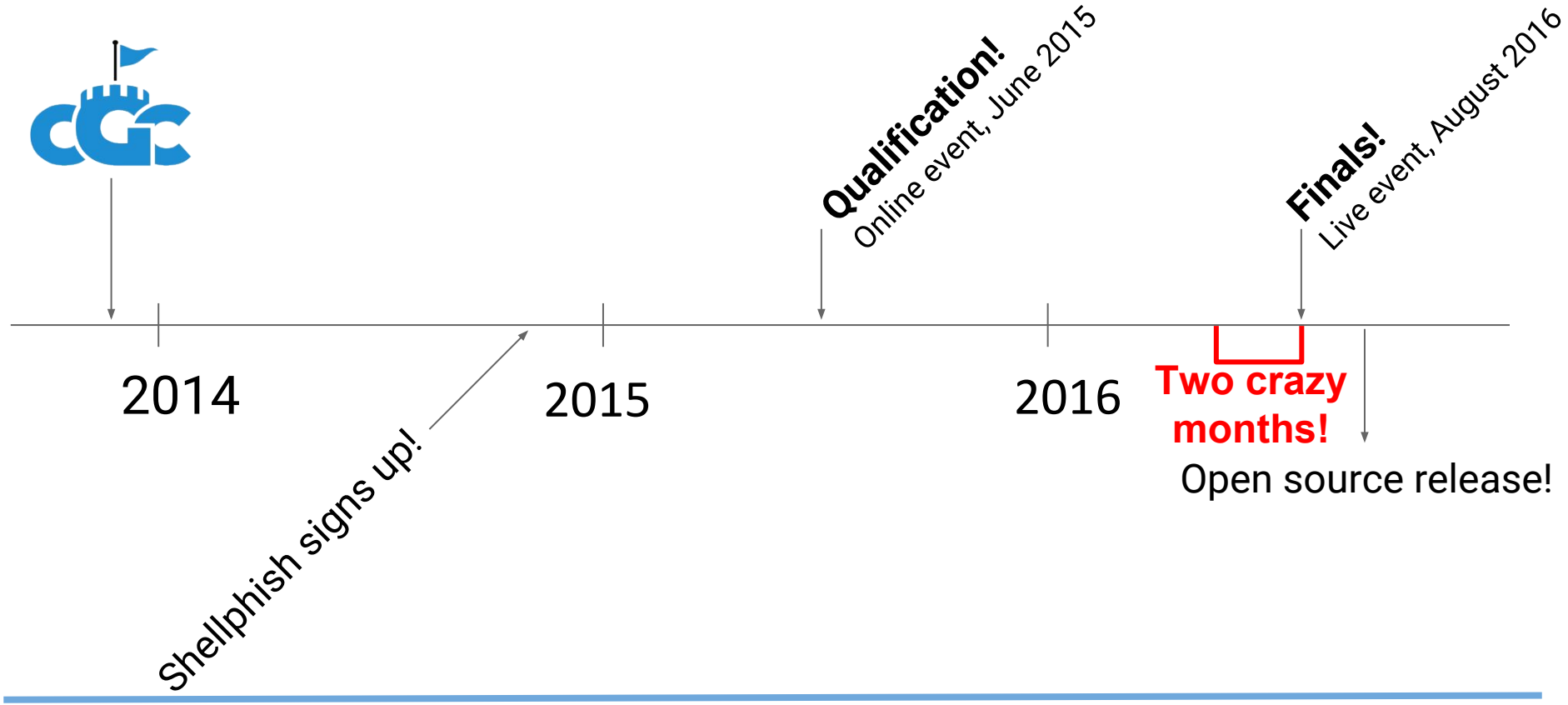
docker

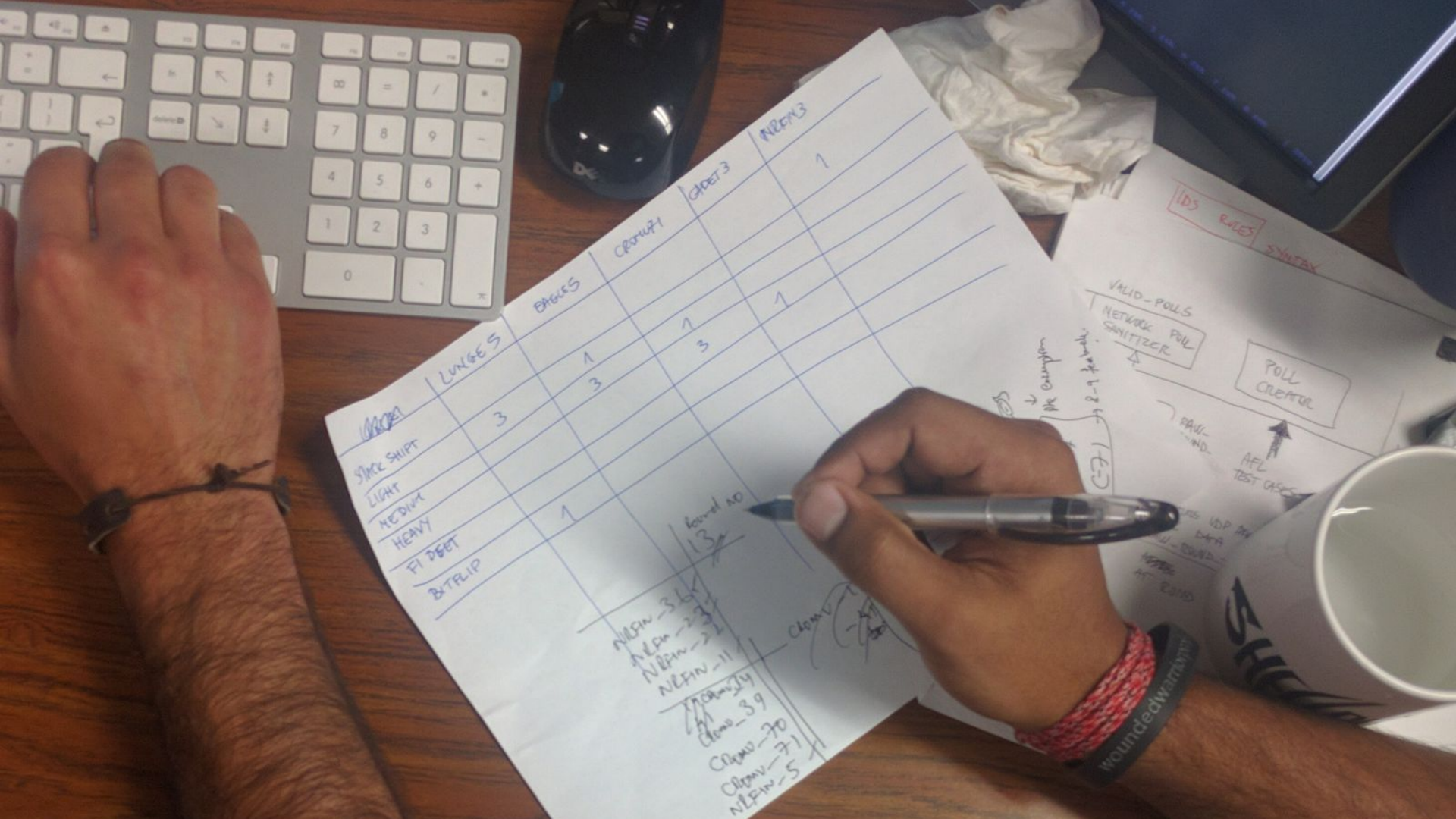
Future Directions

- Human-assisted automatic exploitation and defense

- You can contribute
  - Port code to non-CGC architecture
  - Are you a student?
    Looking for an internship?
    Master thesis?
    Wanting to do a PhD?
    Want a free Shellphish Tshirt/sticker?

CGC – Timeline



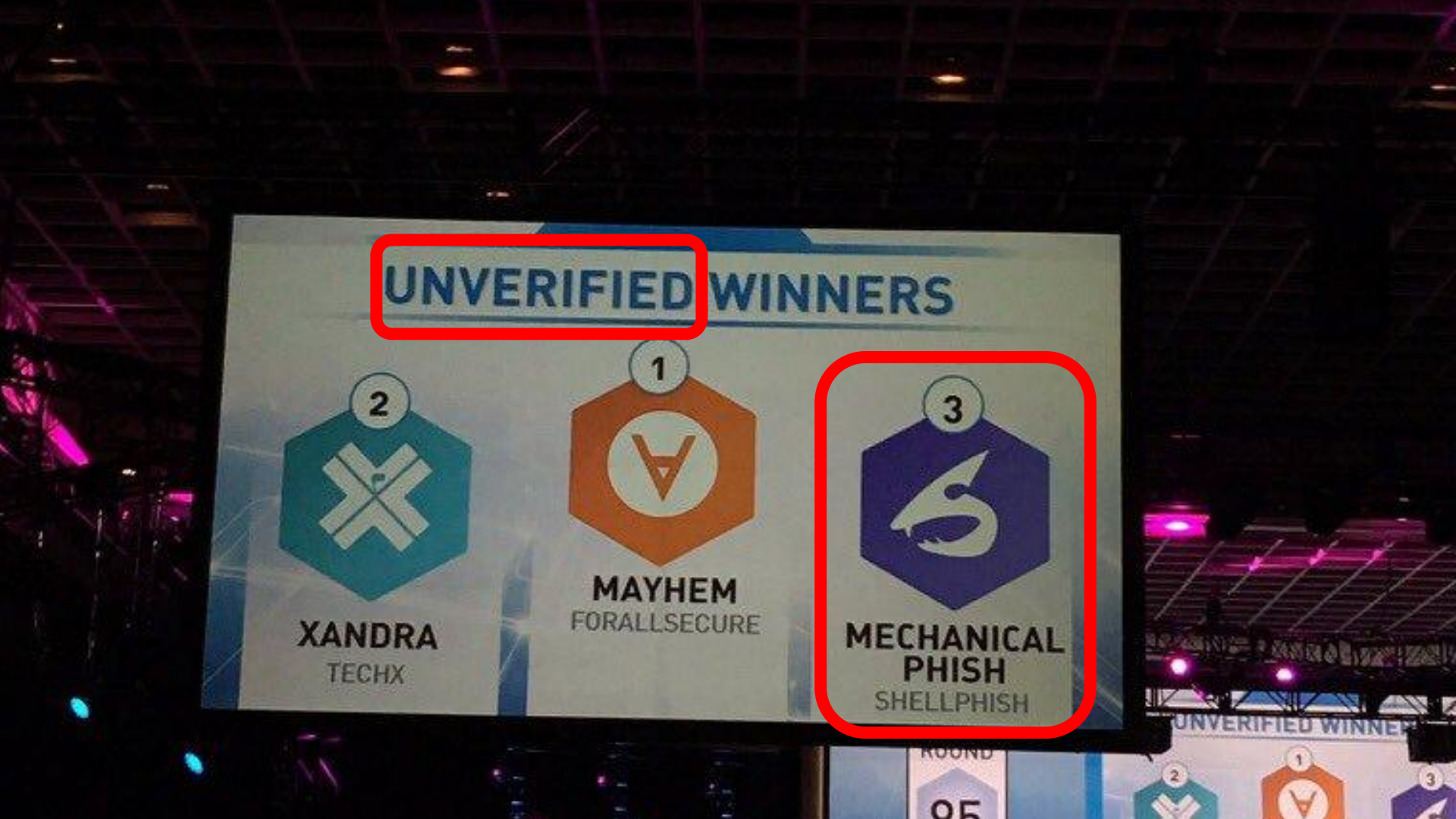

LUNGES
EAGLES
CRUNCH
GADET3
NEFIN3
STACK SHIFT
LIGHT
MEDIUM
HEAVY
FI DBET
BITFLIP
3
1
3
1
3
1
1
1
VALID_POLLS
NETWORK POLL SANITIZER
POLL CREATOR
AFL TEST CASES
wounded warrior


BALLYS


BALLYS

```
i meister cao@stegmutt .) git log --format="format:%C(auto)%ci %h %s" --since="2016-07-26 16:01 -07:00" --until="2016-08-03 15:00 -07:00"
2016-08-03 12:42:26 -0700 bfec79f Merge branch 'fix/colorguard-only-trace-those-untraced' into 'master'
2016-08-03 12:41:30 -0700 f90c995 Log failed pod deletion
2016-08-03 12:41:23 -0700 6f0ac2e Delete failed pods
2016-08-03 12:35:05 -0700 1290f67 Only trace testcases which have been untraced by colorguard
2016-08-03 08:02:29 -0700 ecbe399 create the list in parallel
2016-08-03 06:32:11 -0700 fce13f8 Select only crash.id for colorguard
2016-08-03 06:27:04 -0700 58cc1f7 Fix colorguard and driller creators
2016-08-03 06:22:08 -0700 169b96d Set creator time limit to 15
2016-08-03 05:05:50 -0700 983d261 Use minimum of 2 seconds as a minimum rate for staggering
2016-08-03 04:56:37 -0700 f042428 Fix number of pods needed
2016-08-03 04:55:23 -0700 d582e92 Use runtime to determine jobs to stagger
2016-08-03 04:26:07 -0700 0a90221 Do not kill jobs unnecessarily
2016-08-03 03:34:58 -0700 eb82518 Fix job_ids_to_kill for staggered scheduling
2016-08-03 02:20:23 -0700 c1e8e3e Merge branch 'feature/staggered-priority' into 'master'
2016-08-03 02:11:15 -0700 3fba706 Use set for jobs_to_ignore
2016-08-03 02:03:45 -0700 b76594c Staggered pod creation
2016-08-03 02:01:16 -0700 5eb57fd Merge branch 'fix/pov_fuzzing_devshm' into 'master'
2016-08-03 01:57:55 -0700 a60f7ee up memory for using dev shm
2016-08-02 21:24:38 -0700 44a8c76 Merge branch 'fix/rex-has-time-limit' into 'master'
2016-08-02 01:53:49 -0700 fc72f0b Add time limit of 30 minutes to Rex jobs
2016-08-01 19:21:31 -0700 3f35df3 Merge branch 'fix/patcherex_priority' into 'master'
2016-08-01 19:07:03 -0700 35fa7a6 lower patcherex priority to 200
2016-08-01 15:23:33 -0700 edddb09 Merge branch 'fix/same-notion-everywhere' into 'master'
2016-08-01 15:11:00 -0700 1f34b81 Fix some formatting
2016-08-01 11:19:00 -0700 2ddd699 Make povfuzzer1,2/rex _normalize_sort like colorguard
2016-08-01 11:16:30 -0700 38ca610 Fix import order povfuzzer2
2016-08-01 04:21:46 -0700 9a42566 Merge branch 'fix/fuzzer2' into 'master'
2016-08-01 04:21:05 -0700 697650b fix the payload for fuzzer2
2016-08-01 03:03:10 -0700 b4a9461 Merge branch 'fix/revert-the-revert-so-we-can-test-network-dude-please' into 'master'
2016-08-01 02:57:12 -0700 b70883b Revert "Revert "Merge branch 'feat/showmap-chunky' into 'master'""
2016-08-01 02:19:20 -0700 cd5db97 Merge branch 'revert-507644f8' into 'master'
2016-08-01 02:18:27 -0700 4f5d710 Revert "Merge branch 'feat/showmap-chunky' into 'master'"
2016-08-01 02:12:25 -0700 507644f Merge branch 'feat/showmap-chunky' into 'master'
2016-08-01 02:12:12 -0700 20287d3 Fix formatting
2016-08-01 01:39:43 -0700 9850ded Fix _cpu2float bug
2016-08-01 01:30:33 -0700 2a45a2d ShowmapSync is created on raw round traffics, not rounds
2016-08-01 01:02:25 -0700 74eb387 Merge branch 'fix/limit-crashes-sceduling' into 'master'
2016-07-31 23:08:58 -0700 019bf2b Implement a FEED_LIMIT on creators
2016-07-31 22:52:11 -0700 ab48ff0 Merge branch 'feat/fuzz_others' into 'master'
2016-07-31 19:05:40 -0700 514da33 only select crash id to avoid slowdowns with huge crashes and lots of them
2016-07-31 18:59:48 -0700 e76c108 Merge branch 'fix/colorguard-priority-sorting' into 'master'
2016-07-31 18:53:43 -0700 480687f Fix, remove line overwriting priority set by _normalize_sort
2016-07-31 18:24:13 -0700 017ef73 Make _normalize_sort calls more clear
2016-07-31 17:52:08 -0700 2bbb6e9 Reorder SQL query and formatting
2016-07-31 17:36:11 -0700 3beeb30 schedule pov_fuzzers on opponents
2016-07-31 15:00:07 -0700 2cdf5dc Use sorting for colorguard priorities to avoid conflicts between CSes
2016-07-31 14:40:27 -0700 2f90ab7 Merge branch 'fix/still-schedule-colorguard-if-circumstantial-exists' into 'master'
2016-07-30 23:58:24 -0700 b873261 Proper CI stages
2016-07-30 23:49:12 -0700 13ba3c5 Fix .gitlab-ci indentation
2016-07-30 23:48:57 -0700 f0722d3 Automatically deploy on push to master
2016-07-30 23:15:08 -0700 1b3cd68 Use docker-builder for deploy
2016-07-30 23:09:58 -0700 3c54c70 Enable manual deploy to CGC nodes
2016-07-30 18:48:03 -0700 f2dffb7 Merge branch 'fix/possible-attribute-error-in-colorguard' into 'master'
2016-07-30 18:28:01 -0700 a3b1d8e Still schedule ColorGuard (with lower priority) if a circumstantial type2 exists
2016-07-30 18:25:56 -0700 8334771 Handle the unfortunate case where no resources are set on pod
2016-07-30 18:25:13 -0700 7839ec3 Add missing var to .env
2016-07-30 18:19:59 -0700 30939ed Whitespace
2016-07-30 17:59:55 -0700 6ca8a13 Fix, use crash id instead of old test id
2016-07-30 17:36:54 -0700 d55dc46 Merge branch 'feat/colorguard-on-crashes' into 'master'
2016-07-30 16:56:33 -0700 2f9d1f7 Add comment describing the rationale for the priority value
2016-07-30 14:52:48 -0700 de2d944 Colorguard now creates lower priority jobs for crashes
2016-07-29 15:13:20 -0700 5786e4e Use requests to count resources, not limits
2016-07-29 15:08:50 -0700 992bdcd Merge branch 'feat/less-resources-for-pov-tester' into 'master'
2016-07-29 13:42:35 -0700 2e747f9 Merge branch 'feat/better-pov-tester-logging' into 'master'
2016-07-29 12:46:09 -0700 d8d6e50 PovTesterJob now only requests four cores
2016-07-29 11:47:49 -0700 7356d55 Add better logging for the PovTesterCreator
2016-07-28 05:57:26 -0700 1dc1100 Fix overprovisioning
2016-07-28 05:48:10 -0700 3046051 Merge branch 'fix/threading-overprovision' into 'master'
2016-07-28 05:46:58 -0700 c6e8e6c Overprovision and thread out Kube API
2016-07-28 04:33:34 -0700 81d0d8f Delete succeeded pods
2016-07-28 01:18:28 -0700 83f9f1a Merge branch 'wip/balls-to-the-wall' into 'master'
2016-07-28 01:16:34 -0700 407a41d disabling more
2016-07-28 01:16:34 -0700 6e2b840 disable patch testing
2016-07-28 00:30:18 -0700 7d119dd Merge branch 'fix/slow_pov_test_creator' into 'master'
2016-07-28 00:19:02 -0700 f119f66 fix the slow pov test creator
2016-07-27 23:05:38 -0700 6460ced Merge branch 'feat/afl-slightly-higher-priority' into 'master'
2016-07-27 23:04:13 -0700 308d854 Merge branch 'feature/always-force-afl-jobs' into 'master'
2016-07-27 22:39:32 -0700 ec3fb30 AFL jobs should have slightly higher priorities over other jobs
2016-07-27 22:03:24 -0700 727983a Always force AFLJob by ignoring completed_at
2016-07-27 19:53:28 -0700 4ad04dc Merge branch 'fix/dont-run-colorguard-on-multicbs' into 'master'
2016-07-27 19:40:03 -0700 db7238c ColorGuard should not be scheduled on MultiCBs
2016-07-27 14:54:29 -0700 0b26f7d Merge branch 'fix/showmap-sync-creator-race-condition' into 'master'
2016-07-27 14:46:55 -0700 7dbd1f3 Remove join, use where on RawRoundPoll.round
2016-07-27 14:46:03 -0700 91163f7 Debug message rephrase
2016-07-27 02:04:58 -0700 ae817a5 Make pylint happy
2016-07-27 02:04:35 -0700 0e0c85a Add missing import Job
2016-07-27 02:04:17 -0700 4ddd7bc Remove unused variable multi_cbn
2016-07-27 02:03:44 -0700 5f05354 Remove final newline
2016-07-27 02:03:25 -0700 421bf1e Remove unused imports
2016-07-27 02:02:55 -0700 2bdcdd0 Remove IDSCreator
2016-07-27 00:02:55 -0700 5ebc5bf Fix a race condition in the creation of ShowmapSync jobs.
2016-07-26 22:29:48 -0700 e11c219 Bump to version 1.0.1
2016-07-26 22:29:24 -0700 2e43204 Merge branch 'fix/prev-round-none' into 'master'
2016-07-26 22:28:10 -0700 ca2fcea Fix comparison for prev_round
2016-07-26 17:12:33 -0700 d8cf85f Merge branch 'fix/bump-rex-upper-memory-limit' into 'master'
2016-07-26 16:19:56 -0700 7184e0e Bump up Rex's upper memory limit to 25G
2016-07-26 16:02:17 -0700 5619314 Bump to version 1.0.0
i meister cao@stegmutt .) |                                                                  git meister master ::
```

```
i farnsworth cao@stegmutt .) git log --format="format:%C(auto)%ci %h %s" --since="2016-07-26 16:01 -07:00" --until="2016-08-03 15:00 -07:00"
2016-08-02 05:40:47 -0700 2faf708 Merge branch 'fix/peewee-compat' into 'master'
2016-08-02 05:36:37 -0700 91ba65a Max most_recent query in ExploitSubmissionCable more friendly
2016-08-02 00:55:42 -0700 7457643 Merge branch 'fix/exploit-submission-uniqueness' into 'master'
2016-08-02 00:50:46 -0700 2f15b09 Add cable_exists method
2016-08-02 00:31:05 -0700 2df683d Improve test case for ExploitSubmissionCable
2016-08-02 00:15:56 -0700 6005937 Add test for most_recent on ExploitSubmissionCable
2016-08-02 00:09:20 -0700 133246d Round is also an element of uniqueness for ExploitSubmissionCables
2016-08-01 23:48:20 -0700 19a2b99 ExploitSubmissionCable has round foreign cable and uniqueness for CS and Team
2016-08-01 19:12:12 -0700 421bde5 Merge branch 'fix/do_not_restart_patcherex' into 'master'
2016-08-01 18:55:15 -0700 c699d82 WTF?!? how could this have worked before?
2016-08-01 18:39:07 -0700 dee82bd Debug log print when a peewee operation is retried
2016-08-01 18:17:59 -0700 9c0e71e Merge branch 'feature/retry-harder' into 'master'
2016-08-01 17:53:58 -0700 9780701 Proper indents :P
2016-08-01 17:48:33 -0700 0d930ea Make RetryHarderOperationalError work + fixes
2016-08-01 17:13:37 -0700 a308e27 Retry harder
2016-08-01 11:14:57 -0700 22cddc1 Fix import order raw_round_poll
2016-08-01 11:13:53 -0700 19cf04b Fix indent for challenge_set.py
2016-08-01 03:50:02 -0700 37b5ad4 Build network-dude too
2016-08-01 03:47:17 -0700 80711ee Deploy network-dude too
2016-08-01 03:26:43 -0700 f591897 Merge branch 'fix/most-reliable-does-not-give-backdoor' into 'master'
2016-08-01 02:21:13 -0700 25fd072 Merge branch 'feat/showmap-chunky' into 'master'
2016-08-01 00:27:44 -0700 1b5ab4f ShowmapSync has input_rrt now, not input_round
2016-08-01 00:15:00 -0700 476060b Adding raw round traffic fk
2016-07-31 20:39:39 -0700 f3fd75a do not restart patcherex
2016-07-31 16:28:23 -0700 930d49f Most reliable method on ChallengeSet never returns backdoor POVs
2016-07-31 02:43:23 -0700 ac87163 Merge branch 'wip/magic-list' into 'master'
2016-07-31 01:23:14 -0700 04159a6 update the magic list
2016-07-31 00:16:51 -0700 b97c579 Allow failures for update_vi_image CI job
2016-07-31 00:07:13 -0700 454d8ae Execute CI builds in parallel
2016-07-31 00:02:14 -0700 d16ddbd Proper CI stages and fix .gitlab-ci.yml indentantion
2016-07-30 20:36:48 -0700 753a47d Merge branch 'fix/get-blob-of-test-or-crash' into 'master'
2016-07-30 20:18:20 -0700 f290584 Fix, big bug, need to call blob on output test or crash
2016-07-30 17:36:06 -0700 12a8bf5 Merge branch 'feat/colorguard-on-crashes' into 'master'
2016-07-30 14:49:03 -0700 02de970 Colorguard now has _input_blob
2016-07-28 03:15:14 -0700 305303e Merge branch 'wip/deoptimize' into 'master'
2016-07-28 03:00:06 -0700 8bba019 deprioritize optimizer for now
2016-07-28 00:29:45 -0700 38eaec8 Merge branch 'fix/slow_pov_test_creator' into 'master'
2016-07-28 00:20:47 -0700 de65804 add sha256 indices
2016-07-27 23:03:37 -0700 7324953 Merge branch 'fix/patcherex_restart' into 'master'
2016-07-27 22:48:21 -0700 8d8a50d restart = True in patcherexjob as Yan suggested
2016-07-27 21:16:57 -0700 4f41366 Merge branch 'wip/unicode-strikes-back' into 'master'
2016-07-27 21:03:41 -0700 16f4f2e fix the unicode error here
2016-07-27 18:44:34 -0700 15a0e4c Merge branch 'fix/round-creation' into 'master'
2016-07-27 17:52:57 -0700 b54b8af Add round creation
2016-07-27 15:58:12 -0700 6c892e3 Merge branch 'feat/new_mixins2' into 'master'
2016-07-27 15:45:22 -0700 4bb5603 Merge branch 'fix/reliable-expliot-or-pov-test-results' into 'master'
2016-07-27 15:39:56 -0700 b431d34 Add test case for CS.has_type1
2016-07-27 15:38:05 -0700 510230d Proper indentation
2016-07-27 14:48:05 -0700 7a553d6 _has_type method on ChallengeSet also checks if there is a successful POV for the ChallengeSet
2016-07-27 02:32:21 -0700 bd08300 Remove superfluous parenthesis
2016-07-27 02:32:02 -0700 9f65f19 Remove duplicate method
2016-07-27 02:31:44 -0700 15a7ec1 Fix list argument
2016-07-27 02:31:24 -0700 65cb407 Remove unused imports
2016-07-27 02:06:30 -0700 b1bda47 Order by Round.created_at in ChallengeSet.original_cbns
2016-07-27 01:32:09 -0700 e407a60 new mixins2
2016-07-27 01:03:09 -0700 37473a0 Merge branch 'feature/cable-per-round' into 'master'
2016-07-26 23:21:34 -0700 908ef29 Update test for CSF.create_or_update_available
2016-07-26 23:17:18 -0700 34692cd Fix CSF.create_or_update_submission
2016-07-26 23:17:18 -0700 12df810 Use IDSRuleFielding.create() directly within tests
2016-07-26 23:17:17 -0700 3e82702 Remove IDSRule.submit()
2016-07-26 23:17:17 -0700 ea15bc9 ChallengeSetFielding support for ambassador
2016-07-26 23:17:17 -0700 b57772b Fix prev_round() bug for multiple games
2016-07-26 23:17:17 -0700 b2380cf Add round to CSSubmissionCable
2016-07-26 18:43:37 -0700 a040c47 Merge branch 'fix/undefined-variable-cs-in-challenge-set-fieldings' into 'master'
2016-07-26 18:42:25 -0700 86252ee Make test for create_or_update work
2016-07-26 18:14:21 -0700 1c5a9fa Fix, undefined variable error for cs
2016-07-26 16:52:17 -0700 10c1be7 Bump to version 1.0.0
2016-07-26 16:51:53 -0700 2d2b331 Fix incorrect CS create_or_update
i farnsworth cao@stegmutt .)                                                                  git farnsworth master ::
```


UNVERIFIED WINNERS
2
XANDRA
TECHX
1
MAYHEM
FORALLSECURE
3
MECHANICAL PHISH
SHELLPHISH


MECHANICAL PHISH
THIRD PLACE


66.2 °F

The END

Questions?

References:

- all the technical details: “very soon” published in a “popular security *ezine*”
- this presentation: goo.gl/RvDbxS
- CGC final event show: youtu.be/n0kn4mDXY6I
- Twitter:@shellphish
- Twitter team: @anton00b - @caovc - @giovanni_vigna - @jac_arc - @ltFish_ @machiry_msdic - @nebirhos - @rhelmot - @zardus
- email: antoniob@cs.ucsb.edu - team@shellphish.net
- Github: github.com/shellphish - github.com/mechaphish - github.com/angr