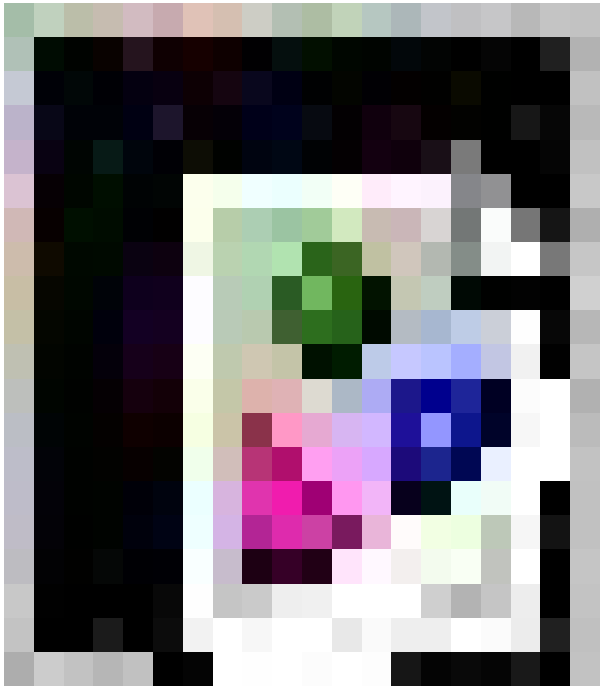
Cyber Grand Shellphish

DEFCON 24

August 7, 2016 · Track 2 - 3pm

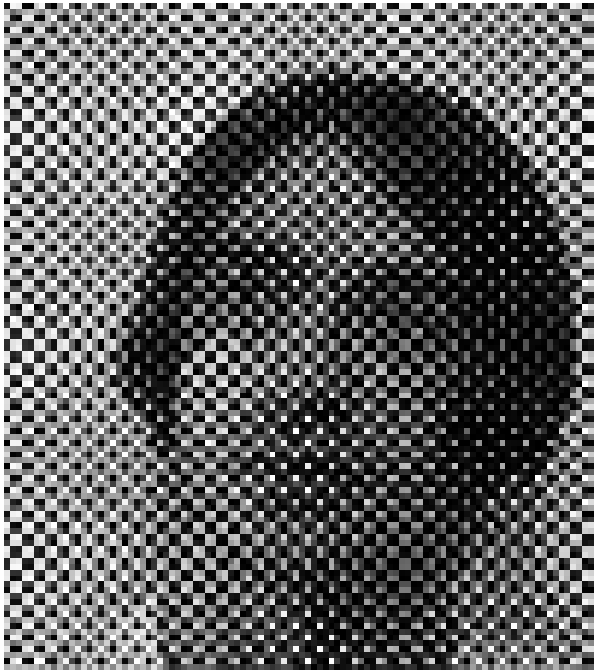
THE COMPUTER SECURITY GROUP AT UC SANTA BARBARA



Giovanni Vigna

Christopher Kruegel

zanardi



VOID


SHELLPHISH
HEX on the Beach

SIMULATION
2004

UC Santa Barbara

sicko irish
nullptr **zanardi**
balzaroth void

SHELLPHISH

SIMULATION
2005

ANTA BARBARA

TU

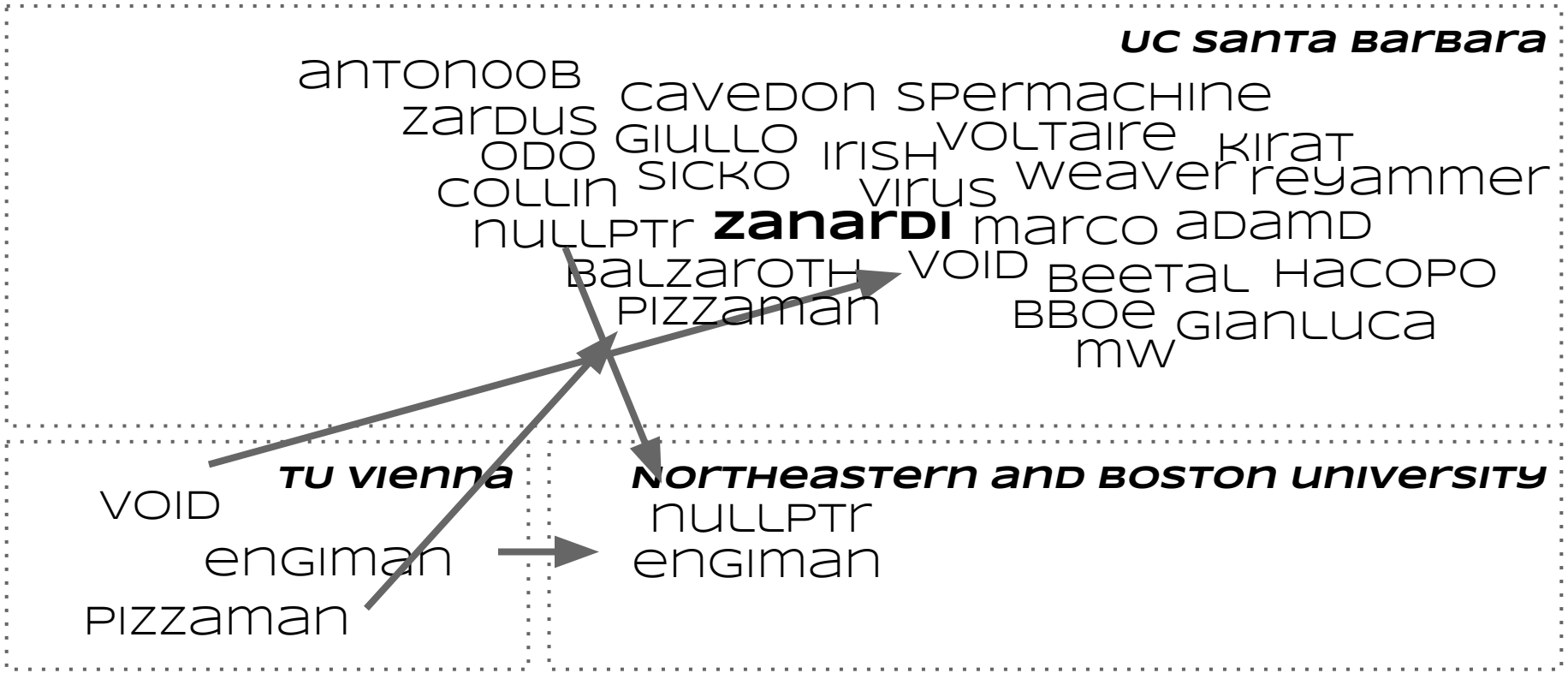
VOID

ENGINEER

PIZZAMAN

SHELLPHISH

SIMULATION
2006 - 2011



SIMULATION
2011 - 2014

UC Santa Barbara

antonoob cavedon spermachine
salls zardus giullo voltaire kirat
subwire fish odo sicko irish weaver reyammer
jay collin virus
zanardi marco adamd
balzaroth void beetal hacopo
pizzamancao bboe gianluca
rhelmot nezorg mw vitor

Northeastern and Boston University

nullptr mw acez
engiman crowell
collin mossberg pizzaman

SIMULATION 2015

UC LONDON
GIANLUCA

ASU
ADAMD

EURECOM
BALZAROTH

UC SANTA BARBARA
ANTONOOB
CAVEDON SPERMACHINE
SALLS ZARDUS
GIULLO VOLTAIRE KIRAT
SUBWIRE FISH ODO
SICKO IRISH WEAVER REYAMMER
VIRUS
DONFOS JAY
ZANARDI MARCO ADAMD
DOUBLE
ACEZ BALZAROTH VOID BEETAL HACOPO
MIKE_PIZZA
CAO BBOE GIANLUCA
RHELMOT NEZORG VITOR

NORTHEASTERN AND BOSTON UNIVERSITY
NULLPTR MW ACEZ
ENGIMAN CROWELL
COLLIN MOSSBERG PIZZAMAN

SIMULATION
MODERN DAY

UC LONDON
GIANLUCA

ASU
ADAMD

EURECOM
BALZAROTH

UC SANTA BARBARA
ANTONOOB
CAVEDON SPERMACHINE
SALLS ZARDUS
GIULLO VOLTAIRE KIRAT
SUBWIRE FISH ODO
SICKO IRISH WEAVER REYAMMER
VIRUS
DONFOS JAY
ZANARDI MARCO
DOUBLE
VOID BEETAL HACOPO
ACEZ
CAO BBOE
MIKE_PIZZA
RHELMOT NEZORG
VITOR

NORTHEASTERN AND BOSTON UNIVERSITY
NULLPTR MW
ENGIMAN CROWELL
COLLIN MOSSBERG PIZZAMAN

UC London

gianluca

SIMULATION

Modern Day

ASU

adamd

Eurecom

balzaroth

UC Santa Barbara

antonoob

cavedon

salls zardus

subwire fish odo

donfos jay

reyammer

zanardi

acez

void

hacopo

cao

mike_pizza

rhelmot nezorg

Northeastern and Boston University

nullptr mw

engiman crowell

mossberg pizzaman

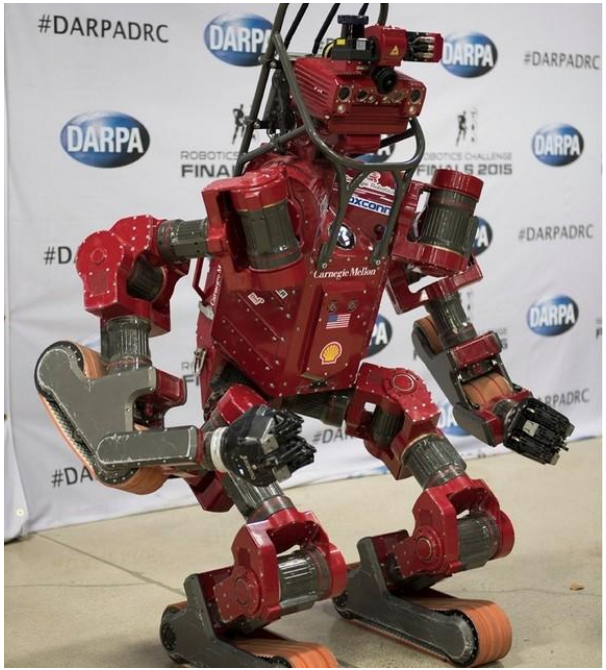
CYBER

GRAND_CHALLENGE

DARPA Competitions

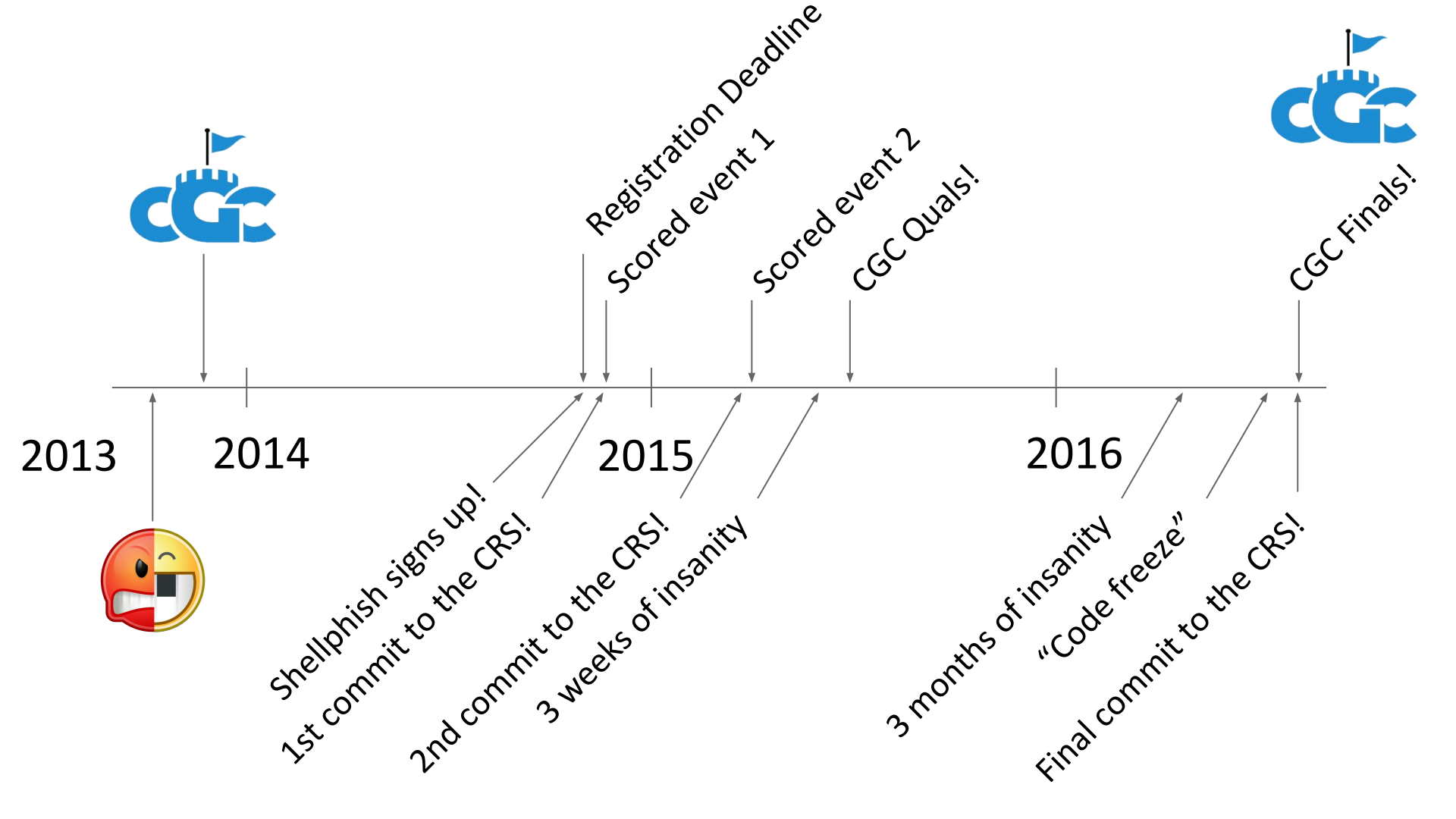
Self-driving Cars

Robots



The DARPA Cyber Grand Challenge

Programs!


Registration Deadline
Scored event 1
Scored event 2
CGC Quals!
CGC Finals!
2013
2014
2015
2016
Shellphish signs up!
1st commit to the CRS!
2nd commit to the CRS!
3 weeks of insanity
3 months of insanity
“Code freeze”
Final commit to the CRS!


SHELLPHISH


SHELLPHISH


SHELLFISH

analyze

pwn

patch

analyze

pwn

patch

- Linux-inspired environment, with only 7 syscalls
  - `transmit / receive / fdwait (≈select)`
  - `allocate / deallocate`
  - `random`
  - `terminate`

- No need to model the POSIX API!

- Otherwise real(istic) programs.

analyze

pwn

patch

- No filesystem -> no flag?

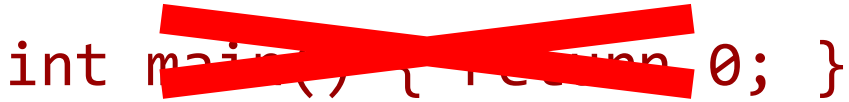
- CGC Quals: crash == exploit

- CGC Finals: two types of exploits
    1. "flag overwrite": set a register to X, crash at Y
    2. "flag read": leak the "secret flag" from memory

analyze

pwn

patch



fails functionality checks...

no signal handling!

inline QEMU-based CFI?

performance penalties...

Mechanical Phish (CQE)

A completely autonomous system

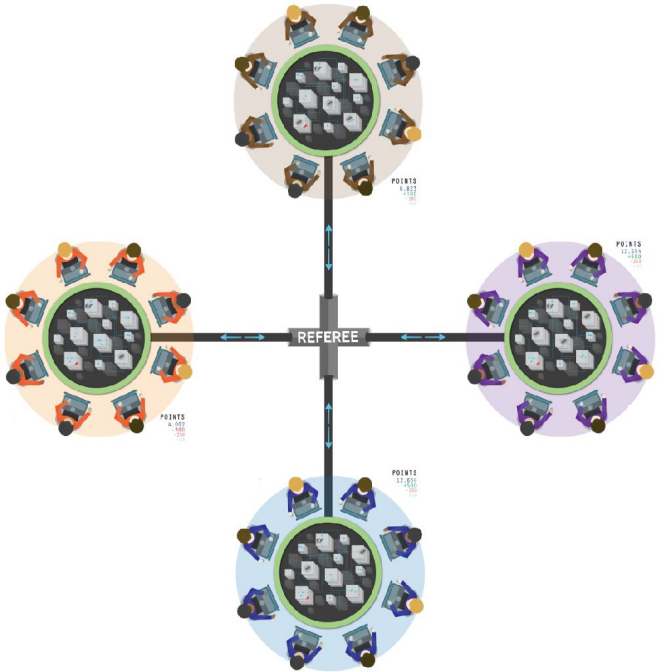
- Patch
- Crash

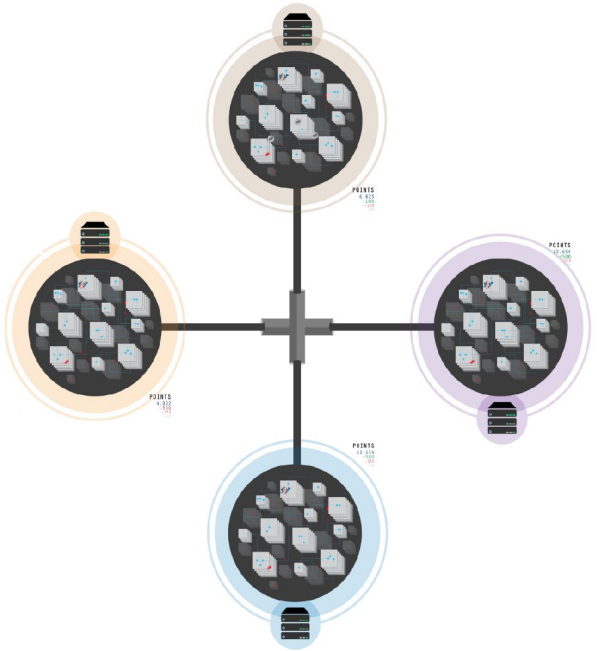
Mechanical Phish (CFE)

Completely autonomous system

- Patch
- Crash
- Exploit

The DARPA Cyber Grand Challenge





The CGC Final Event (CFE)

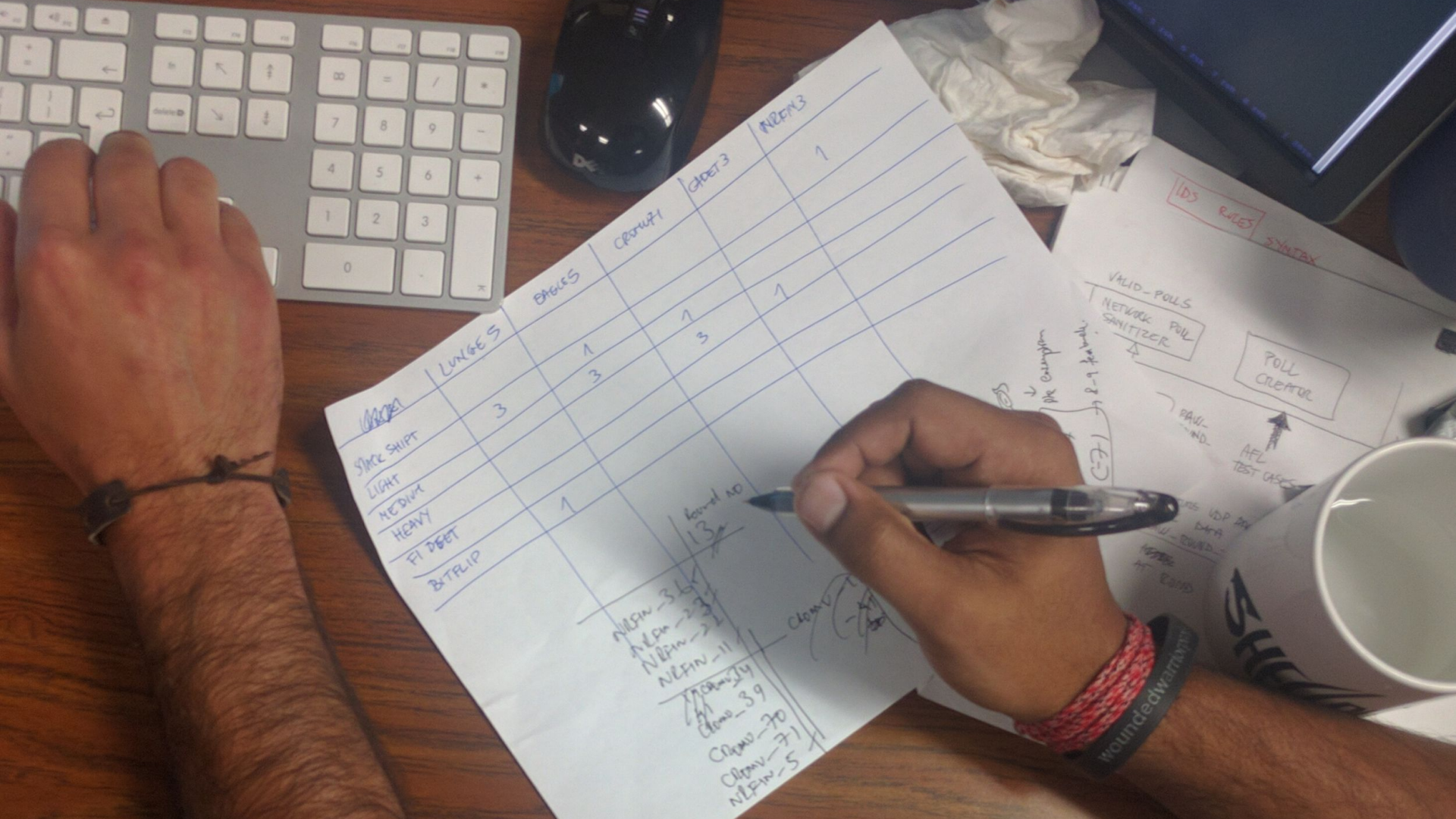
- The competition is divided in rounds (96), with short breaks between rounds
- The competition begins: The system provides a set of Challenge Binaries (CBs) to the teams' CRSs
  - Each CB provides a service (e.g., an HTTP server)
  - Initially, all teams are running the same binaries to implement each service
- For each round, a score for each (team, service) tuple is generated

The CGC Final Event (CFE)

$$\sum_{i=0}^{\#CB} Availability \times Security \times Evaluation$$

- Availability: how badly did you fuck up the binary?
- Security: did you defend against *all* exploits?
- Evaluation: how many n00bs did you pwn?

- When you are shooting blindfolded automatic weapons, it’s easy to shoot yourself in the foot…


LUNGES
EAGLES
CRONUS
CADET3
NLFIN3
SPACE SHIPS
LIGHT
MEDIUM
HEAVY
FI DBET
BITFLIP
IDS RULES
SYNTAX
VALID_POLLS
NETWORK POLL SANITIZER
POLL CREATOR
AFL TEST CASES
found AD
NLFIN_34
NLFIN_33
NLFIN_31
NLFIN_11
CRONUS_39
CRONUS_70
CRONUS_71
NLFIN_5

Code Freeze?

cao 4:01 PM
farnsworth has been freezed

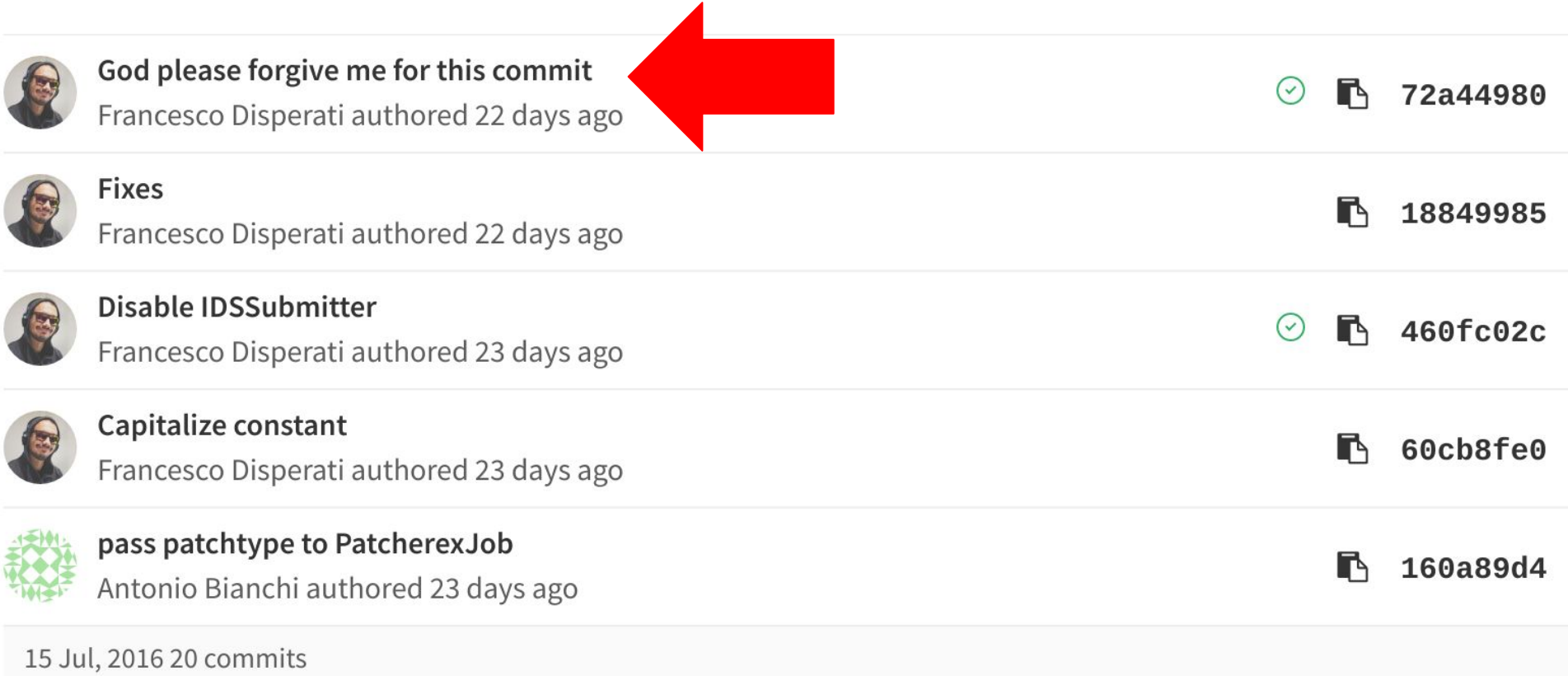
all outstanding merge requests have been merged in

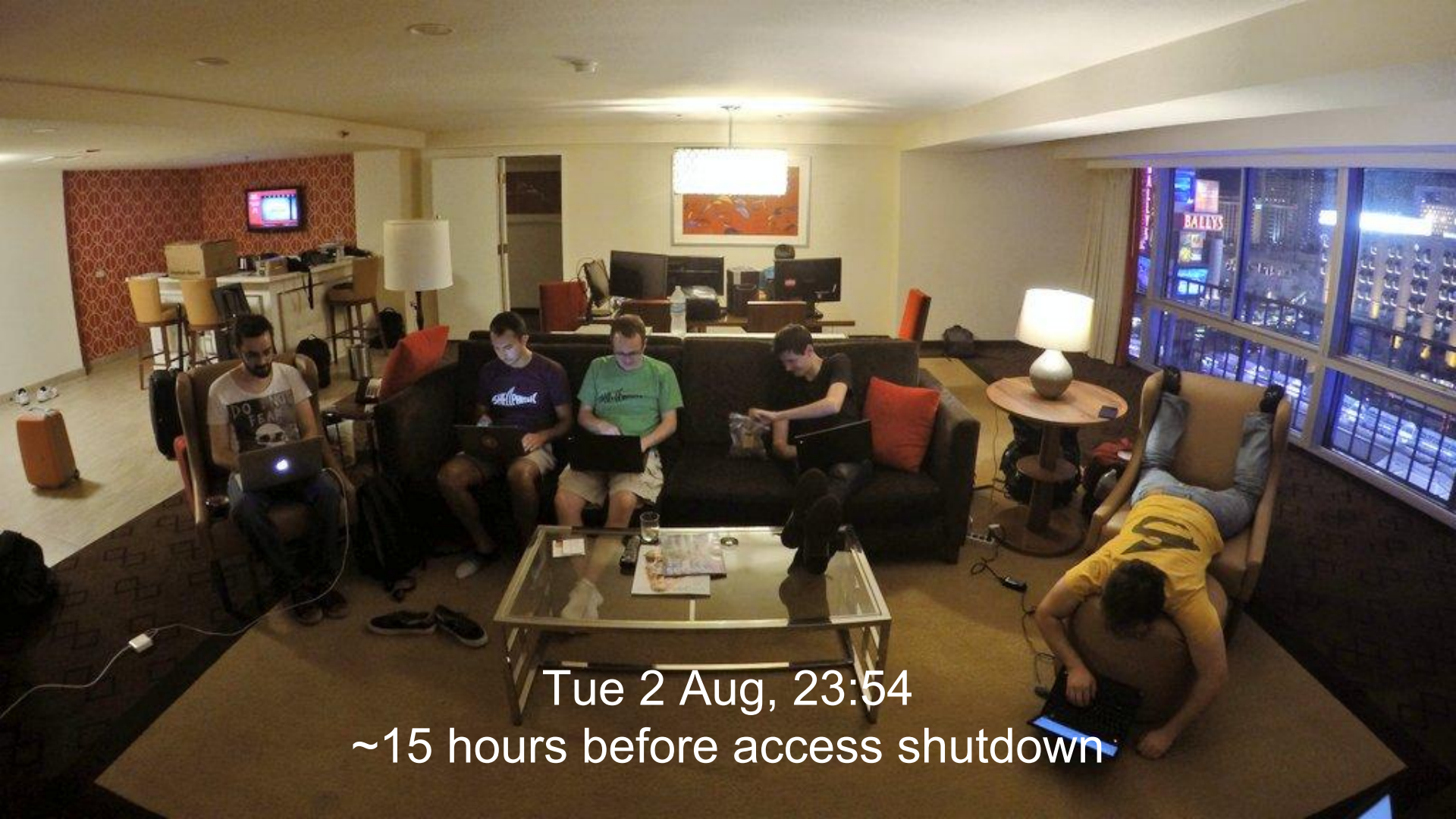
mike_pizza 4:01 PM
holy shit

cao 4:02 PM
set the channel topic: meister and farnsworth are in code freeze

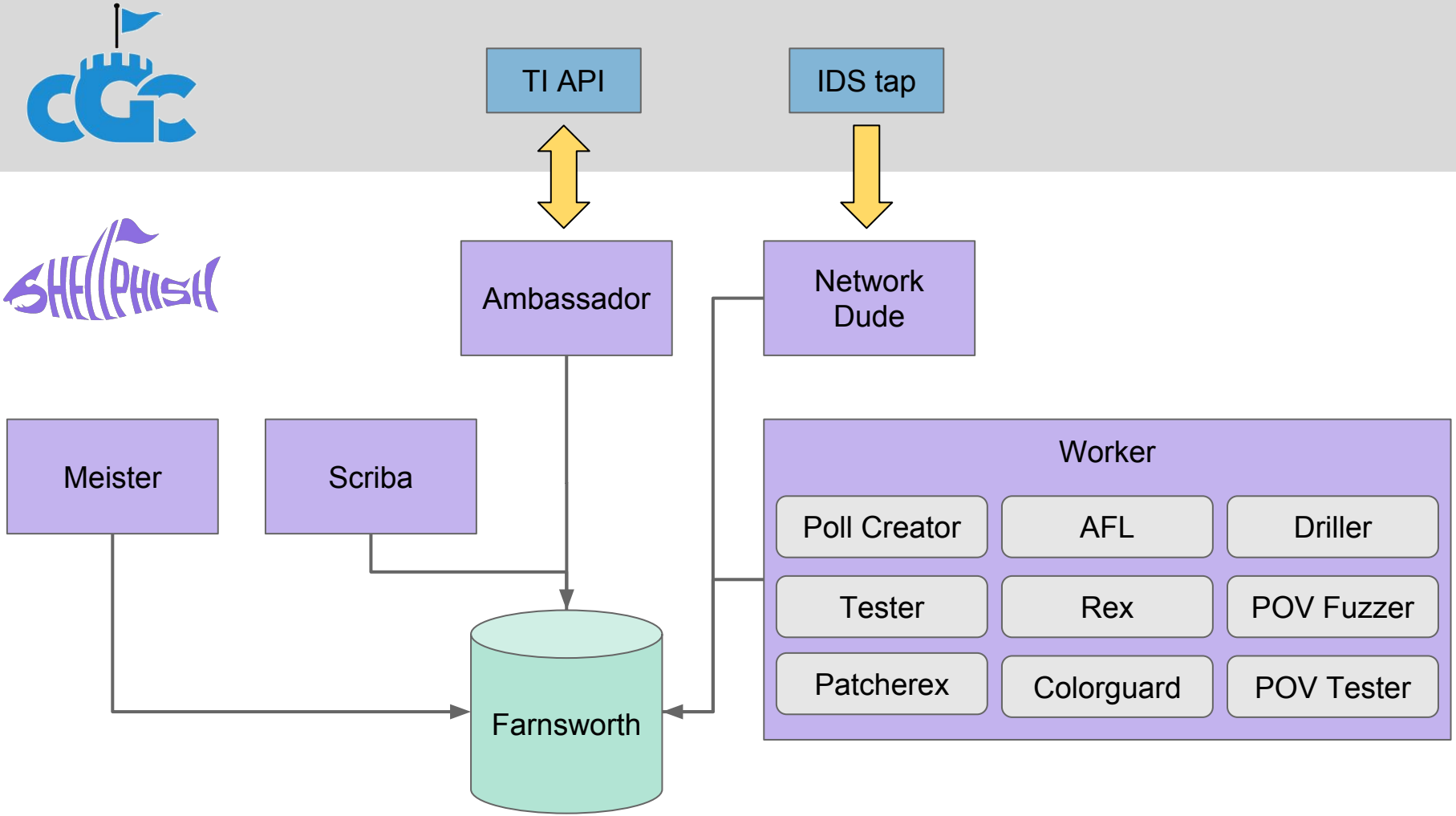
```
i meister cao@stegmutt .) git log --format="format:%C(auto)%ci %h %s" --since="2016-07-26 16:01 -07:00" --until="2016-08-03 15:00 -07:00"
2016-08-03 12:42:26 -0700 bfec79f Merge branch 'fix/colorguard-only-trace-those-untraced' into 'master'
2016-08-03 12:41:30 -0700 f90c995 Log failed pod deletion
2016-08-03 12:41:23 -0700 6f3c22e 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 38cc1f7 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 d582a92 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 fc92f0b 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 eddbd09 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 b4a5461 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 4006f7f 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 2cdf54c 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 d56c46 Merge branch 'fix/colorguard-on-crashes' into 'master'
2016-07-30 16:56:13 -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 01:16:14 -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 db723bc 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 00:55:51 -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 1332460 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 4e0d82bd 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 17:53:48 -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 f991807 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 17:36:06 -0700 930d6ff 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 00:07:31 -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 de658dd 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 4f41386 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-exploit-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 ::
```


God please forgive me for this commit
Francesco Disperati authored 22 days ago
72a44980
Fixes
Francesco Disperati authored 22 days ago
18849985
Disable IDSSubmitter
Francesco Disperati authored 23 days ago
460fc02c
Capitalize constant
Francesco Disperati authored 23 days ago
60cb8fe0
pass patchtype to PatcherexJob
Antonio Bianchi authored 23 days ago
160a89d4
15 Jul, 2016 20 commits



Tue 2 Aug, 23:54
~15 hours before access shutdown


TI API
IDS tap
Ambassador
Network Dude
Meister
Scriba
Worker
Poll Creator
AFL
Driller
Tester
Rex
POV Fuzzer
Patcherex
Colorguard
POV Tester
Farnsworth

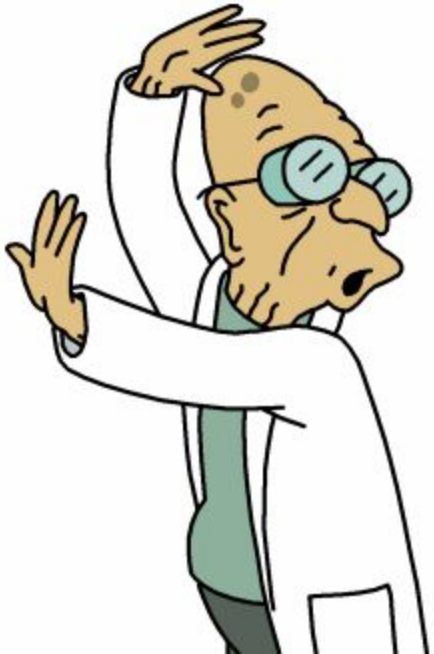
Farnsworth

Object-relational model for database:

- What CS are fielded this round?
- Do we have crashes?
- Do we have a good patch?
- ...

Our ground truth and the only component reasonably well tested*

* 69% coverage



Meister

Job scheduler:

- Looks at game state
- Asks creators for jobs
- Schedules them based on priority

```
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
```

On the Shoulders of Giants

QEMU
open source processor emulator

Z3

python

AFL

kubernetes

Unicorn
Engine

ubuntu

angr

pypy

peewee

PostgreSQL

VEX

Capstone
Engine

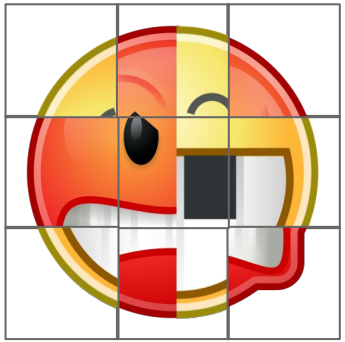
docker

angr

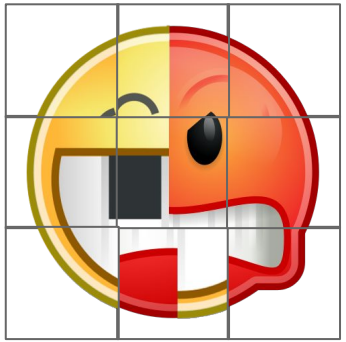
- Framework for the analysis of binaries, developed at UCSB
- Supports a number of architectures
  - x86, MIPS, ARM, PPC, etc. (all 32 and 64 bit)
- Open-source, free for commercial use (!)
  - http://angr.io
  - https://github.com/angr
  - angr@lists.cs.ucsb.edu

angr

angr



Concolic Execution



Automatic Exploitation

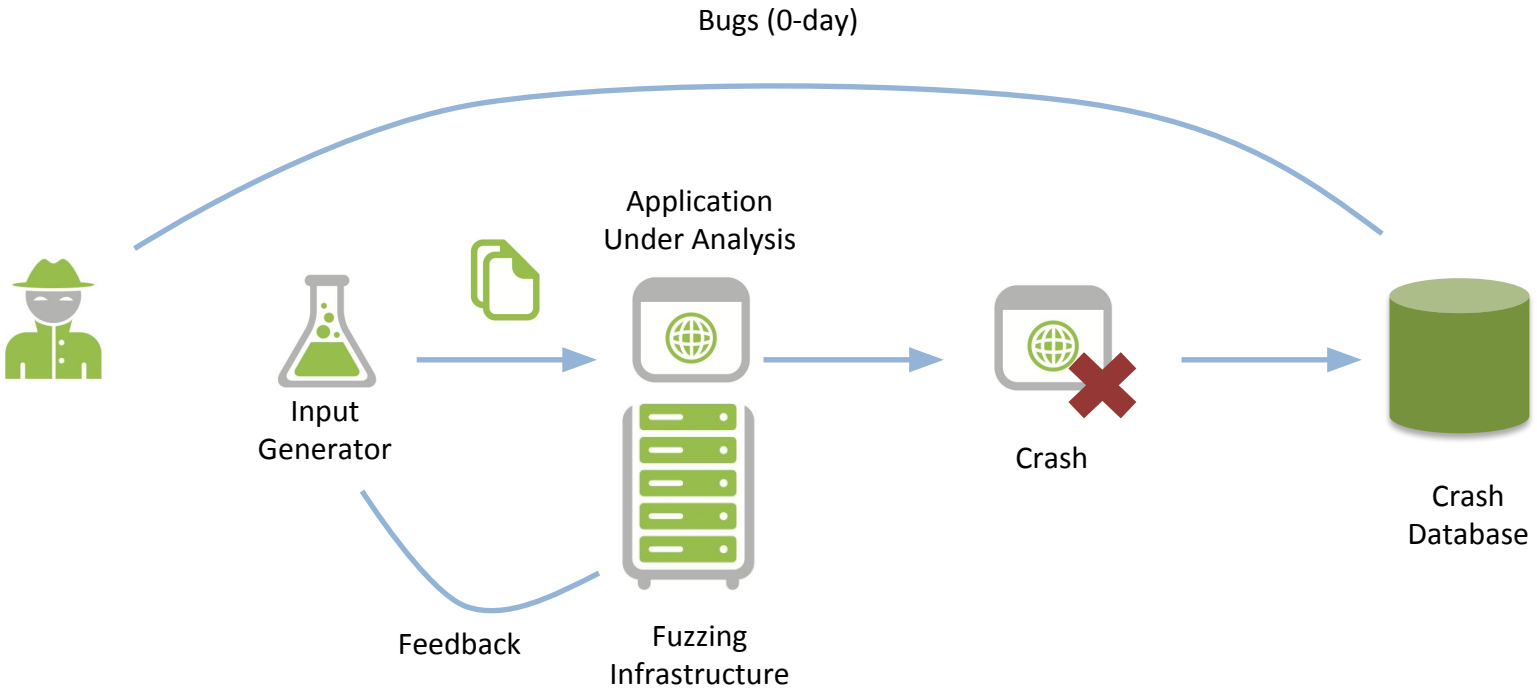
Patching


M 9686

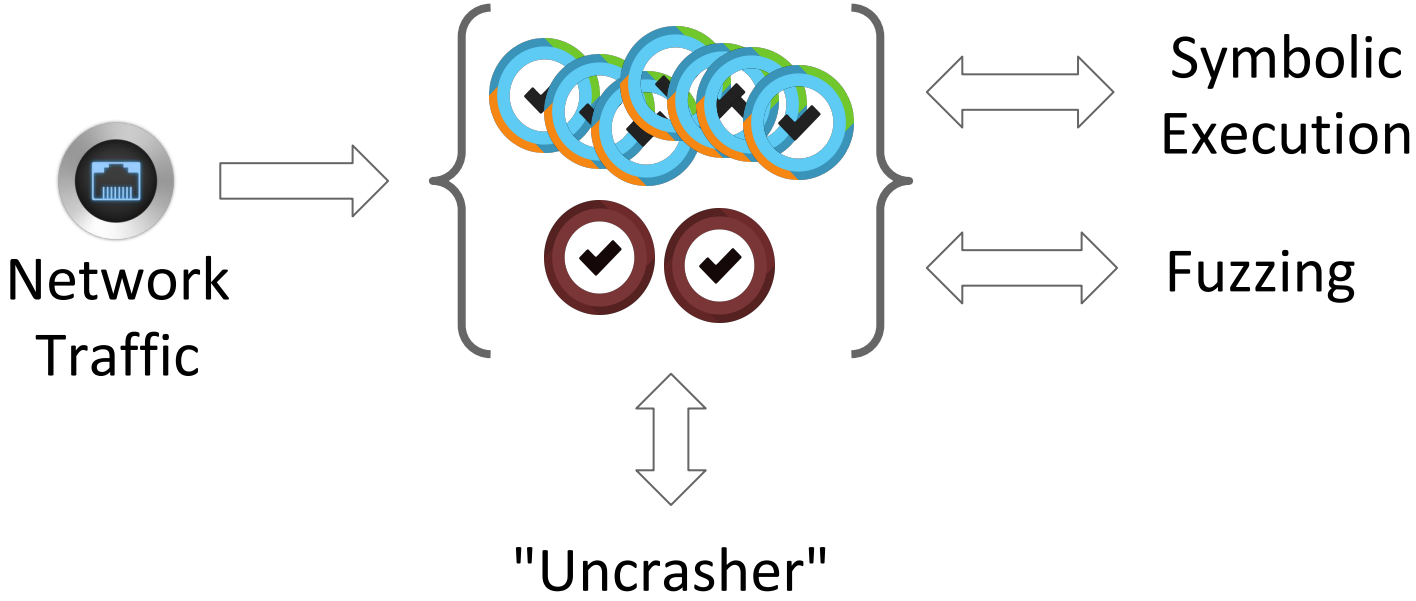
Fuzzing

- Fuzzing is an automated procedure to send inputs and record safety condition violations as crashes
  - Assumption: crashes are potentially exploitable
- Several dimensions in the fuzzing space
  - How to supply inputs to the program under test?
  - How to generate inputs?
  - How to generate more “relevant” crashes?
  - How to change inputs between runs?
- Goal: maximized effectiveness of the process

Gray/White-box Fuzzing



How do we find crashes?



Fuzzing: American Fuzzy Lop

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

Let's fuzz it!

1 ⇒ "You lose!"

593 ⇒ "You lose!"

183 ⇒ "You lose!"

4 ⇒ "You lose!"

498 ⇒ "You lose!"

42 ⇒ "You win!"

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

Let's fuzz it!

1 ⇒ "You lose!"

593 ⇒ "You lose!"

183 ⇒ "You lose!"

4 ⇒ "You lose!"

498 ⇒ "You lose!"

42 ⇒ "You lose!"

3 ⇒ "You lose!"

.....

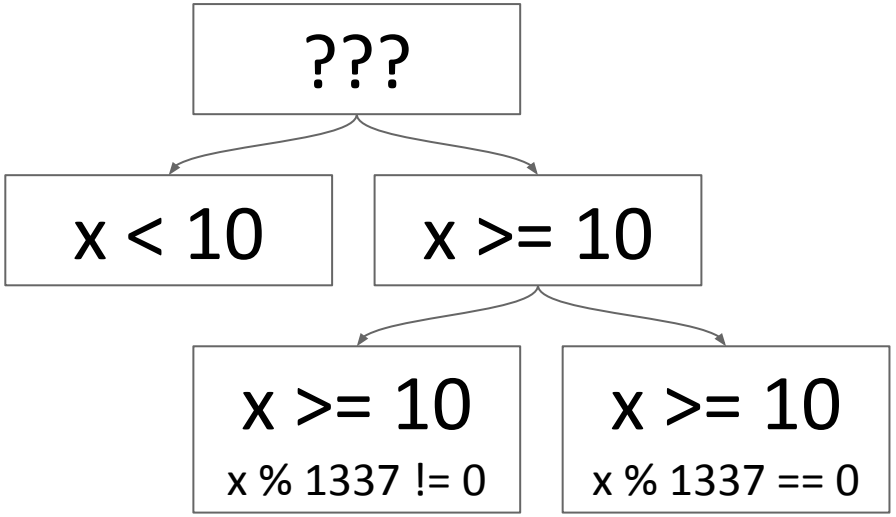
57 ⇒ "You lose!"

- Very fast!

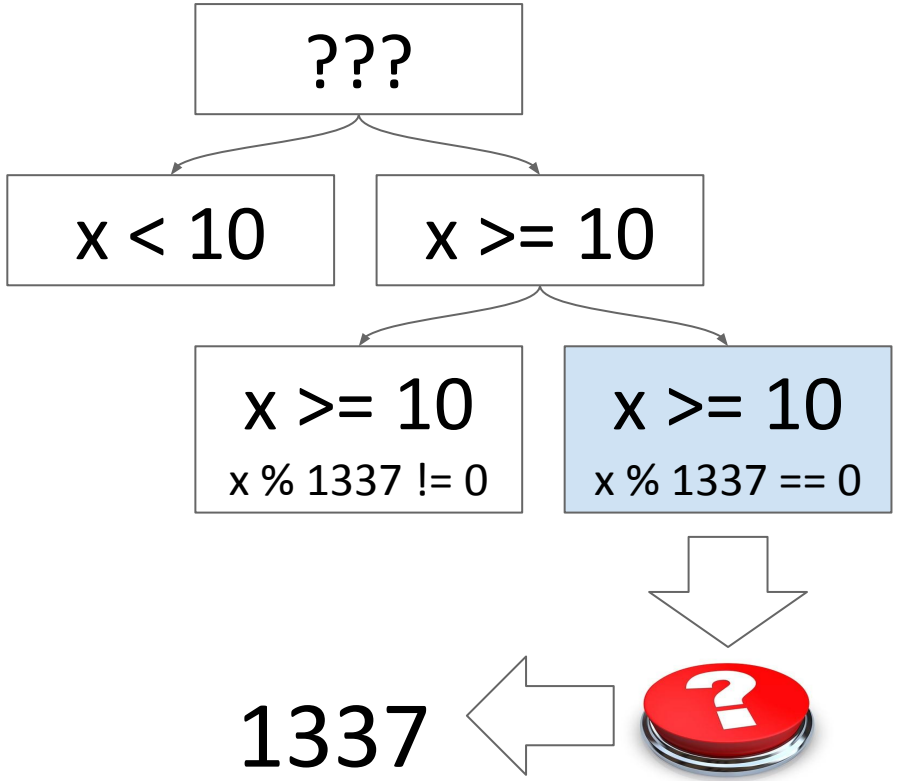
- Very effective!

- Unable to deal with certain situations:
  - magic numbers
  - hashes
  - specific identifiers

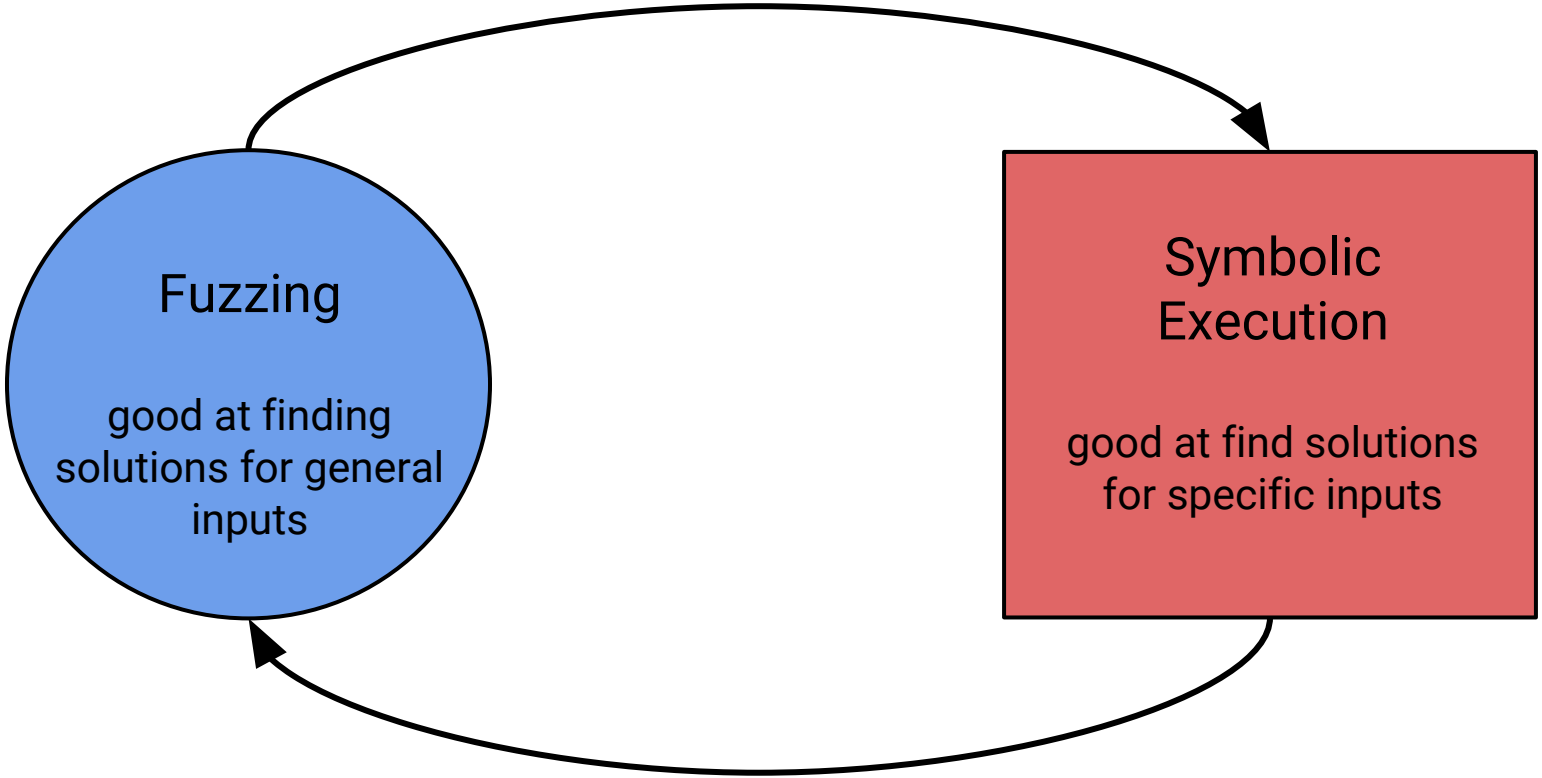
```
x = input()
if x >= 10:
    if x % 1337 == 0:
        print "You win!"
    else:
        print "You lose!"
else:
    print "You lose!"
```



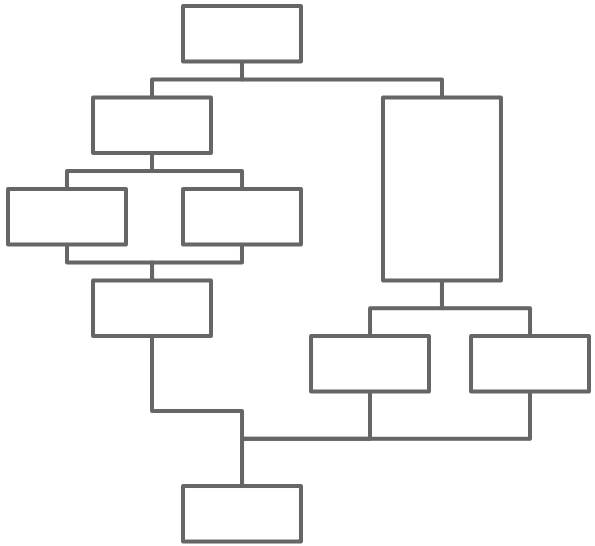
```
x = input()
if x >= 10:
    if x % 1337 == 0:
        print "You win!"
    else:
        print "You lose!"
else:
    print "You lose!"
```



Driller = AFL + angr



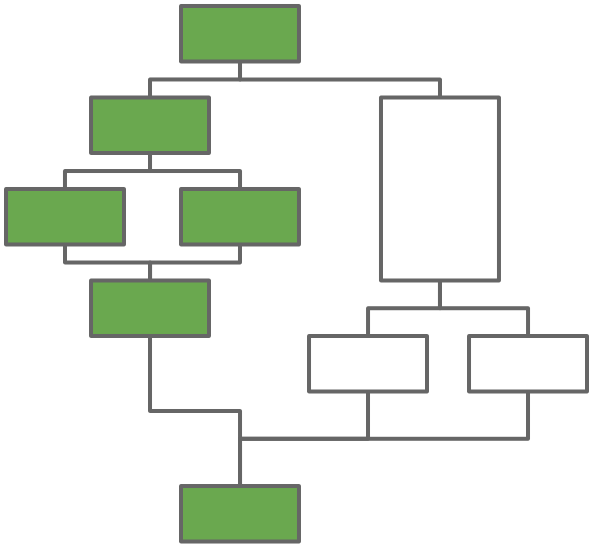
Driller



Test Cases

Driller

"Cheap" fuzzing coverage

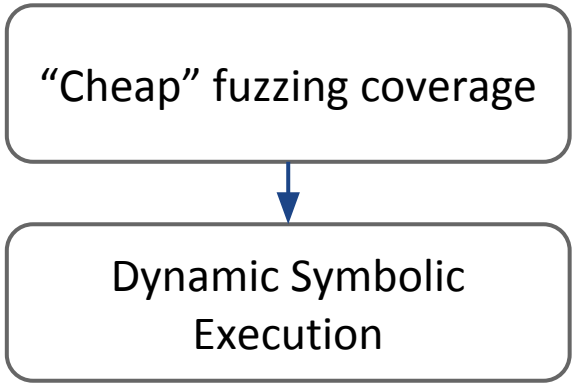


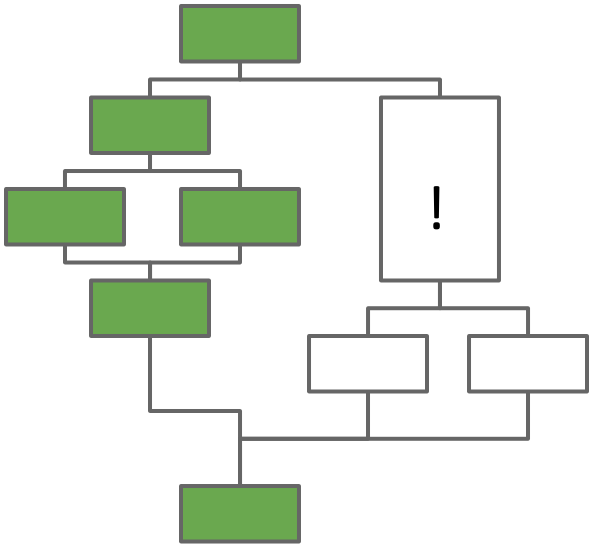
Test Cases

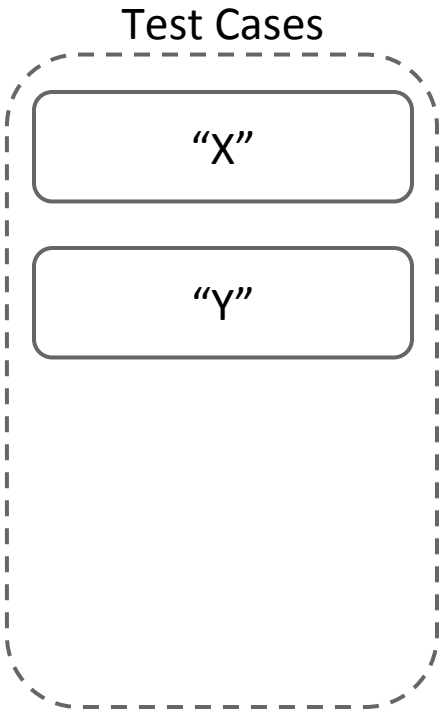
"X"

"Y"

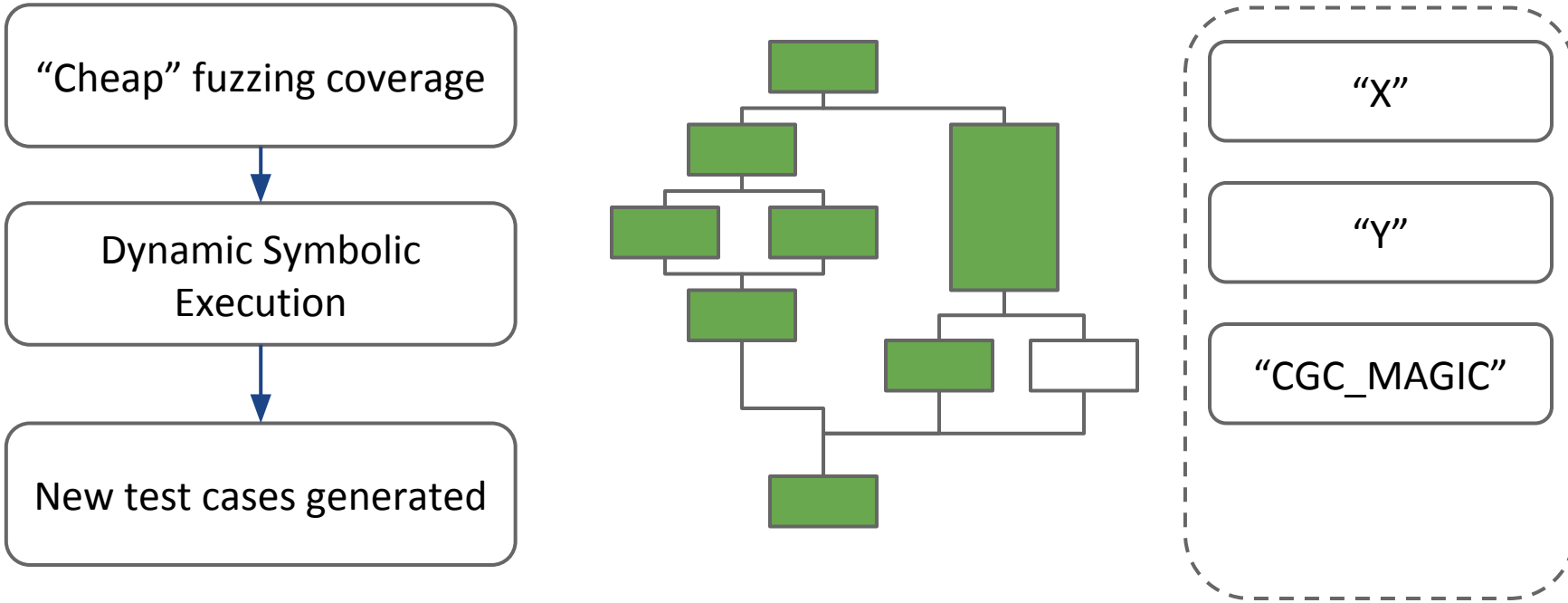
Driller


“Cheap” fuzzing coverage
Dynamic Symbolic Execution

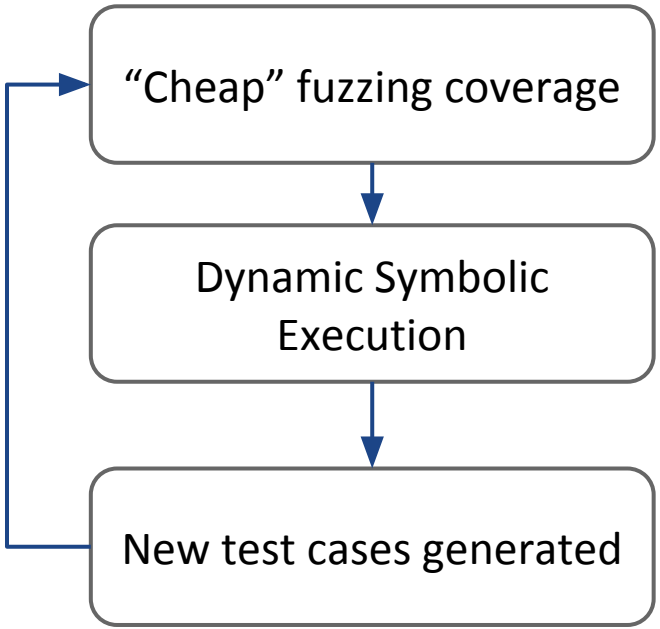

!

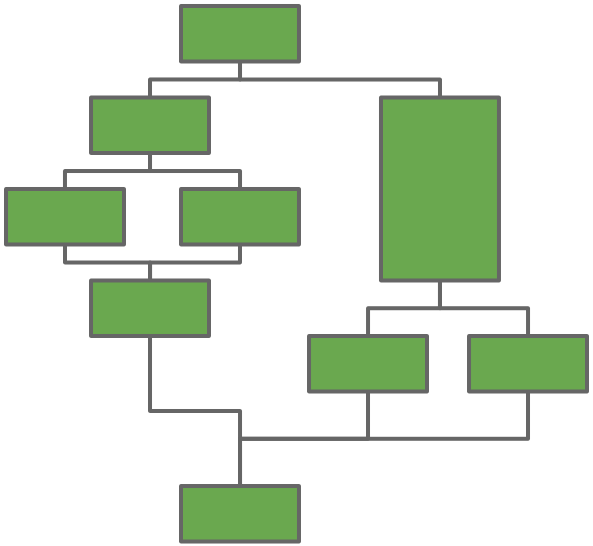

Test Cases
“X”
“Y”

Driller


“Cheap” fuzzing coverage
Dynamic Symbolic Execution
New test cases generated
Test Cases
“X”
“Y”
“CGC_MAGIC”

Driller





Test Cases

- “X”
- “Y”
- “CGC_MAGIC”
- “CGC_MAGICY”

Auto Exploitation - Simplified

```
typedef struct component {
      char name[32];
      int (*do_something)(int arg);
} comp_t;

comp_t *initialize_component(char *cmp_name) {
      int i = 0;
      struct component *cmp;

      cmp = malloc(sizeof(struct component));
      cmp->do_something = sample_func;

      while (*cmp_name)
            cmp->name[i++] = *cmp_name++;

      cmp->name[i] = '\0';
      return cmp;
}
x = get_input();
cmp = initialize_component(x);
cmp->do_something(1);
```

HEAP

```
Symbolic Byte[0]
Symbolic Byte[1]
Symbolic Byte[2]
Symbolic Byte[3]
Symbolic Byte[4]
Symbolic Byte[5]
Symbolic Byte[6]
Symbolic Byte[7]
...
```

```
Symbolic Byte[32] …
Symbolic Byte[36]
```

```
'\0'
```

call <symbolic byte[36:32]>

Auto Exploitation - Simplified

Turning the state into an **exploited** state

angr

```
assert state.se.symbolic(state.regs.pc)
```

Constrain **buffer** to contain our shellcode

angr

```
buf_addr = find_symbolic_buffer(state, len(shellcode))
mem = state.memory.load(buf_addr, len(shellcode))
state.add_constraints(mem == state.se.bvv(shellcode))
```

Auto Exploitation - Simplified

Constrain **PC** to point to the buffer

angr

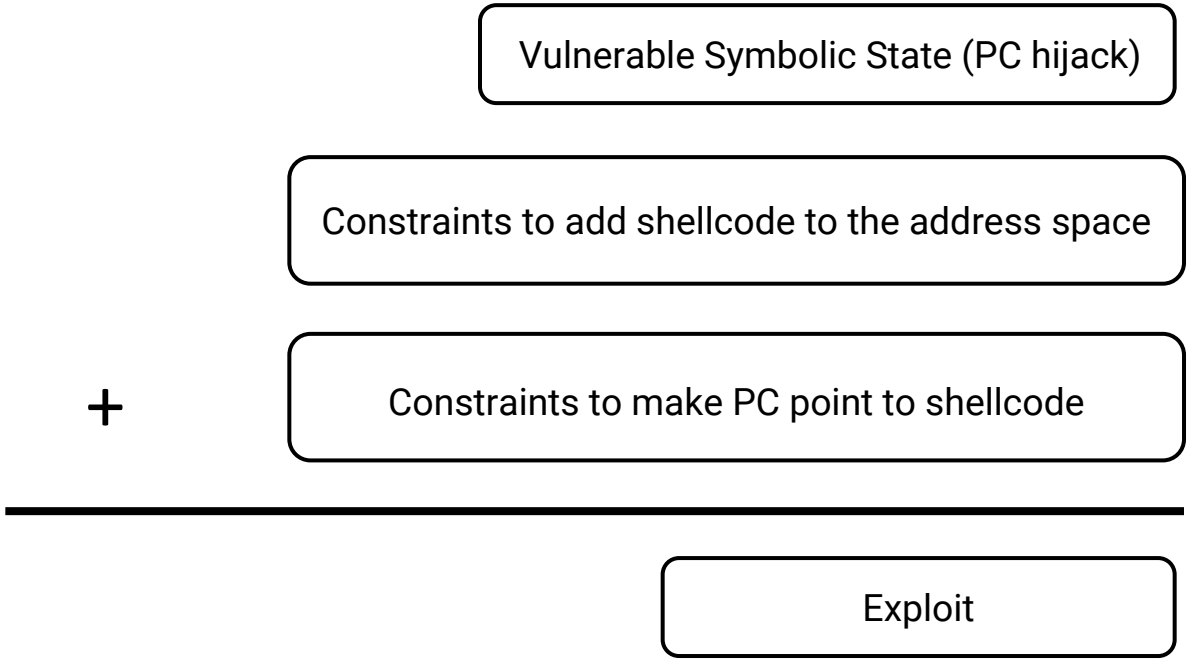
```
state.se.add_constraints(state.regs.pc == buf_addr)
```

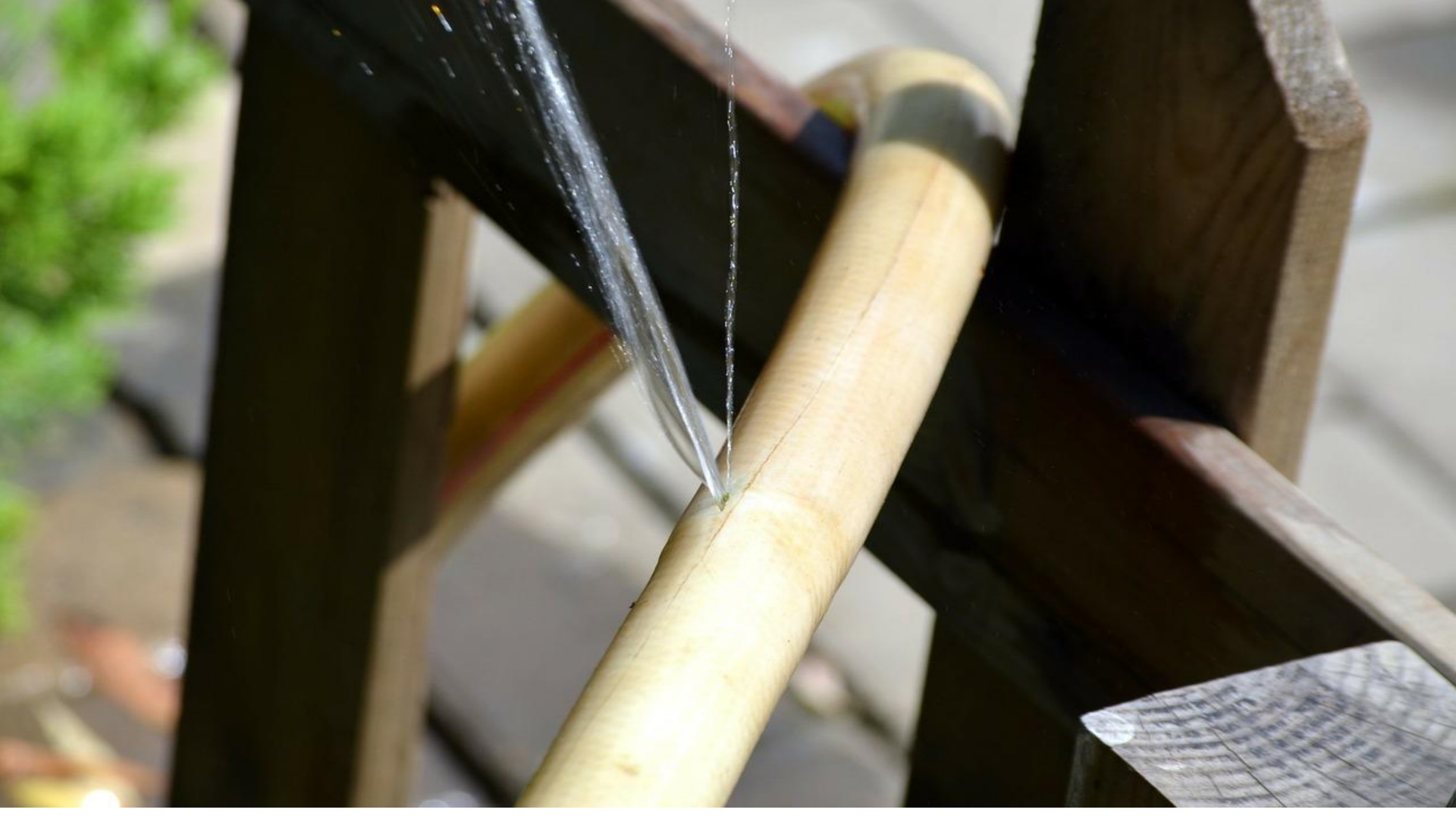
Synthesize!

angr

```
exploit = state.posix.dumps(0)
```

Auto Exploitation - Simplified

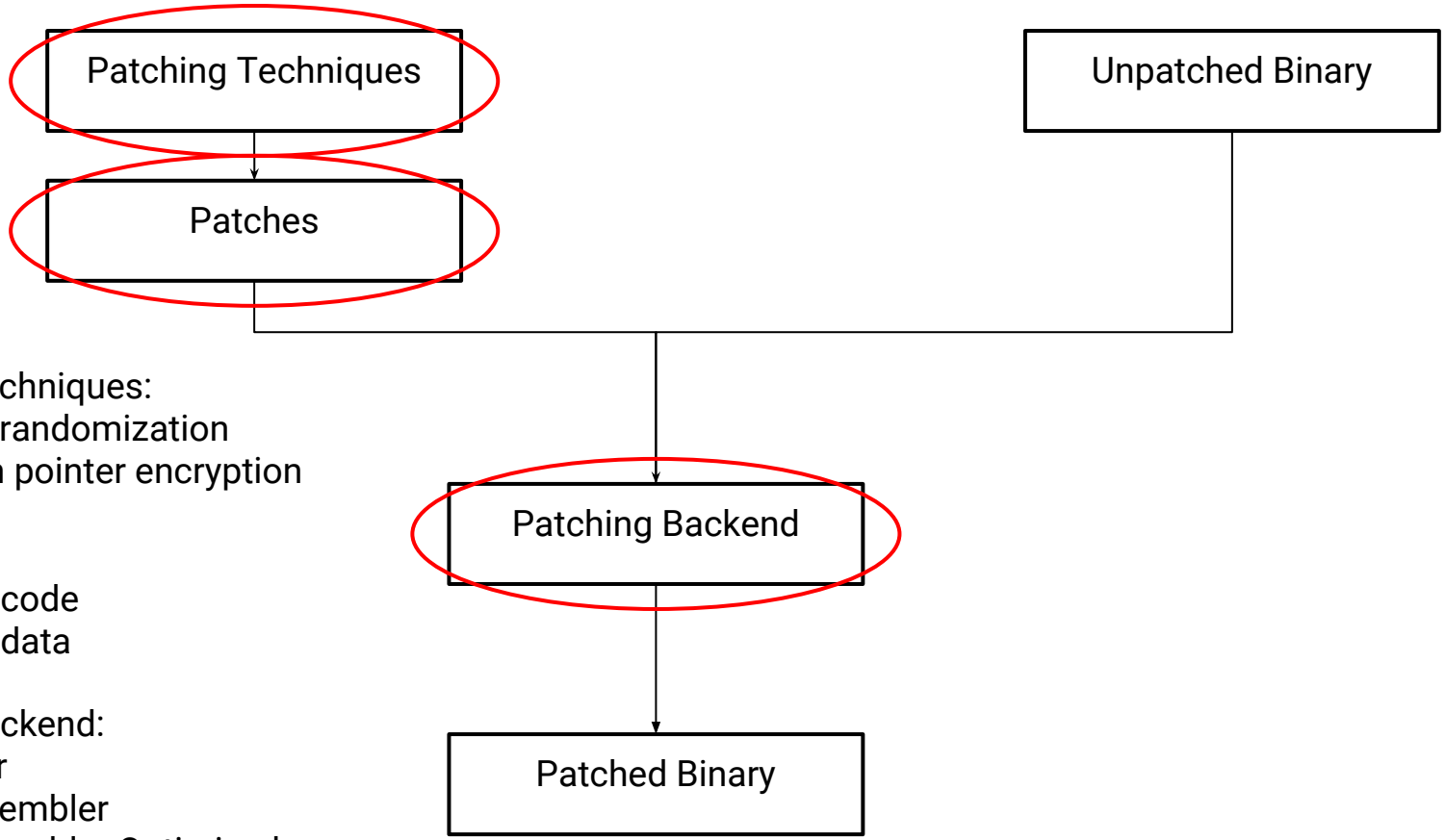




Detecting Leaks of the Flag Page

- Make only the flag page symbolic
- Everything else is completely concrete
  - Can execute most basic block with the Unicorn Engine!
- When we have idle cores on the CRS, trace all our testcases
- Solved DEFCON CTF LEGIT_00009 challenge

Patcherex



Patching Techniques:
- Stack randomization
- Return pointer encryption
- ...

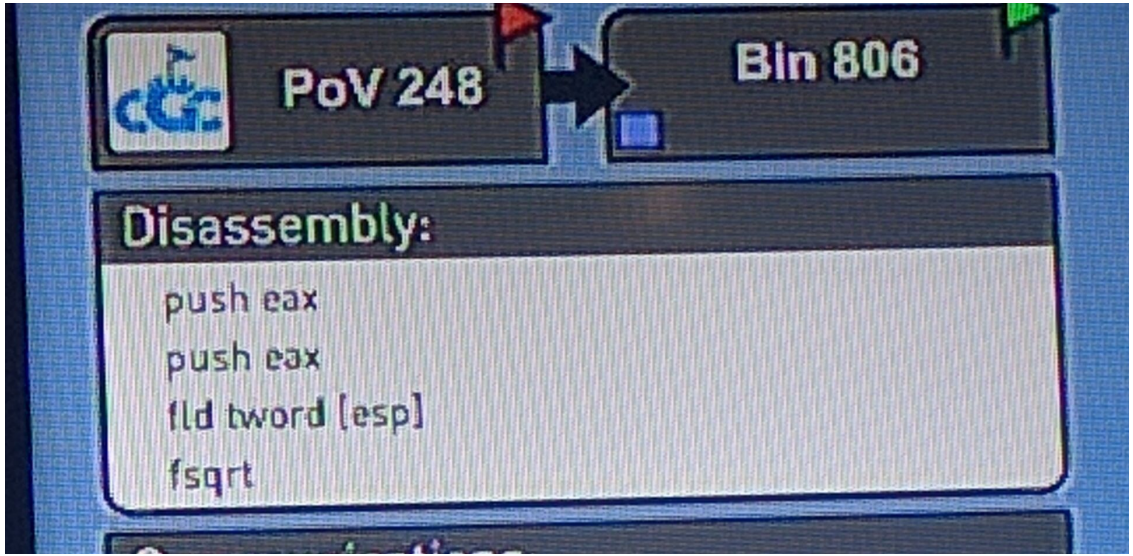
Patches:
- Insert code
- Insert data
- ...

Patching Backend:
- Detour
- Reassembler
- Reassembler Optimized

Adversarial Patches 1/2

Detect QEMU

```
xor eax, eax
inc eax
push eax
push eax
push eax
fld TBYTE PTR [esp]
fsqrt
```



Adversarial Patches 2/2

Transmit the flag

- To **stderr**!

Backdoor

- hash-based challenge-response backdoor
- not “cryptographically secure” → good enough to defeat automatic systems

Generic Patches

Return pointer encryption

Protect indirect calls/jmps

Extended Malloc allocations

Randomly shift the stack (ASLR)

Clean uninitialized stack space

Targeted Patches

Qualification event → avoid crashes!

Targeted Patches

Final event →

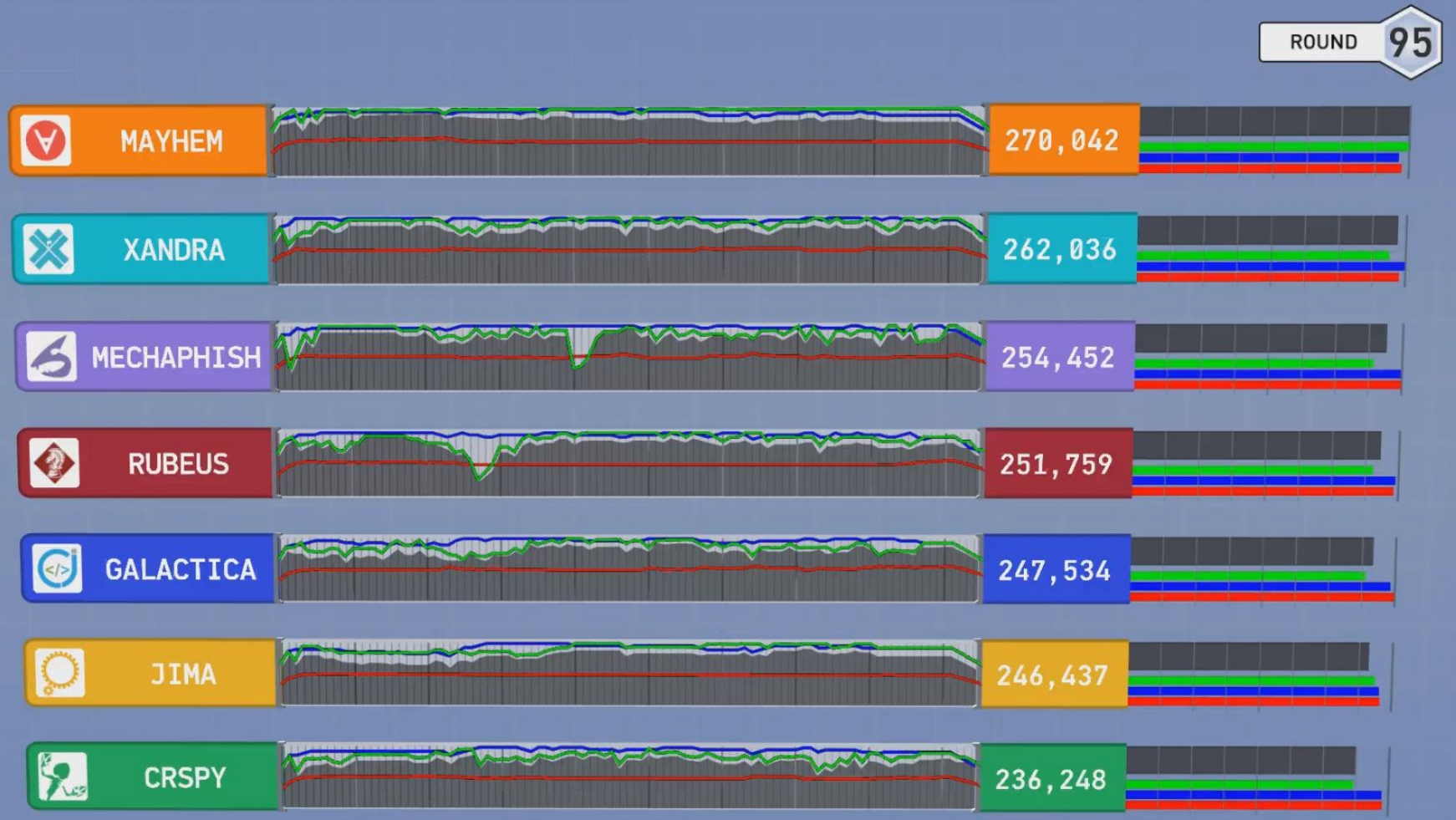
Reassembler & Optimizer

- Prototypes in **3** days

angr is awesome!!

- A big bag of tricks integrated, which worked out


THE dashboard
Zoom Out
Last 3 hours
CPU
1.0
0.8
0.6
0.4
0.2
0
17:00
17:10
17:20
17:30
17:40
17:50
18:00
18:10
18:20
18:30
18:40
18:50
19:00
19:10
19:20
19:30
19:40
19:50
Median
p95
p5
Memory
30.00%
25.00%
20.00%
15.00%
10.00%
5.00%
0%
17:00
17:10
17:20
17:30
17:40
17:50
18:00
18:10
18:20
18:30
18:40
18:50
19:00
19:10
19:20
19:30
19:40
19:50
Median
p95
p5


ROUND 95
MAYHEM
270,042
XANDRA
262,036
MECHAPHISH
254,452
RUBEUS
251,759
GALACTICA
247,534
JIMA
246,437
CRSPY
236,248

CGC CFE Statistics 1/3

- 82 Challenge Sets fielded
- 2442 Exploits generated
- 1709 Exploits for 14/82 CS with 100% Reliability
- Longest exploit: 3791 lines of C code
- Shortest exploit: 226 lines of C code
- crackaddr: 517 lines of C code

CGC CFE Statistics 2/3

100% reliable exploits generated for:

- YAN01_000{15,16}
- CROMU_000{46,51,55,65,94,98}
- NRFIN_000{52,59,63}
- KPRCA_00{065,094,112}

Rematch Challenges:

- SQLSlammer (CROMU_00094)
- crackaddr (CROMU_00098)

CGC CFE Statistics 3/3

Vulnerabilities in CS we exploited:

- CWE-20 Improper Input Validation
- CWE-119 Improper Restriction of Operations within the Bounds of a Memory Buffer
- CWE-121: Stack-based Buffer Overflow
- CWE-122: Heap-based Buffer Overflow
- CWE-126: Buffer Over-read
- CWE-131: Incorrect Calculation of Buffer Size
- CWE-190: Integer Overflow or Wraparound
- CWE-193 Off-by-one Error
- CWE-201: Information Exposure Through Sent Data
- CWE-202: Exposure of Sensitive Data Through Data Queries)
- CWE-291: Information Exposure Through Sent Data
- CWE-681: Incorrect Conversion between Numeric Types
- CWE-787: Out-of-bounds Write
- CWE-788: Access of Memory Location After End of Buffer


DEFCON

Human augmentation...

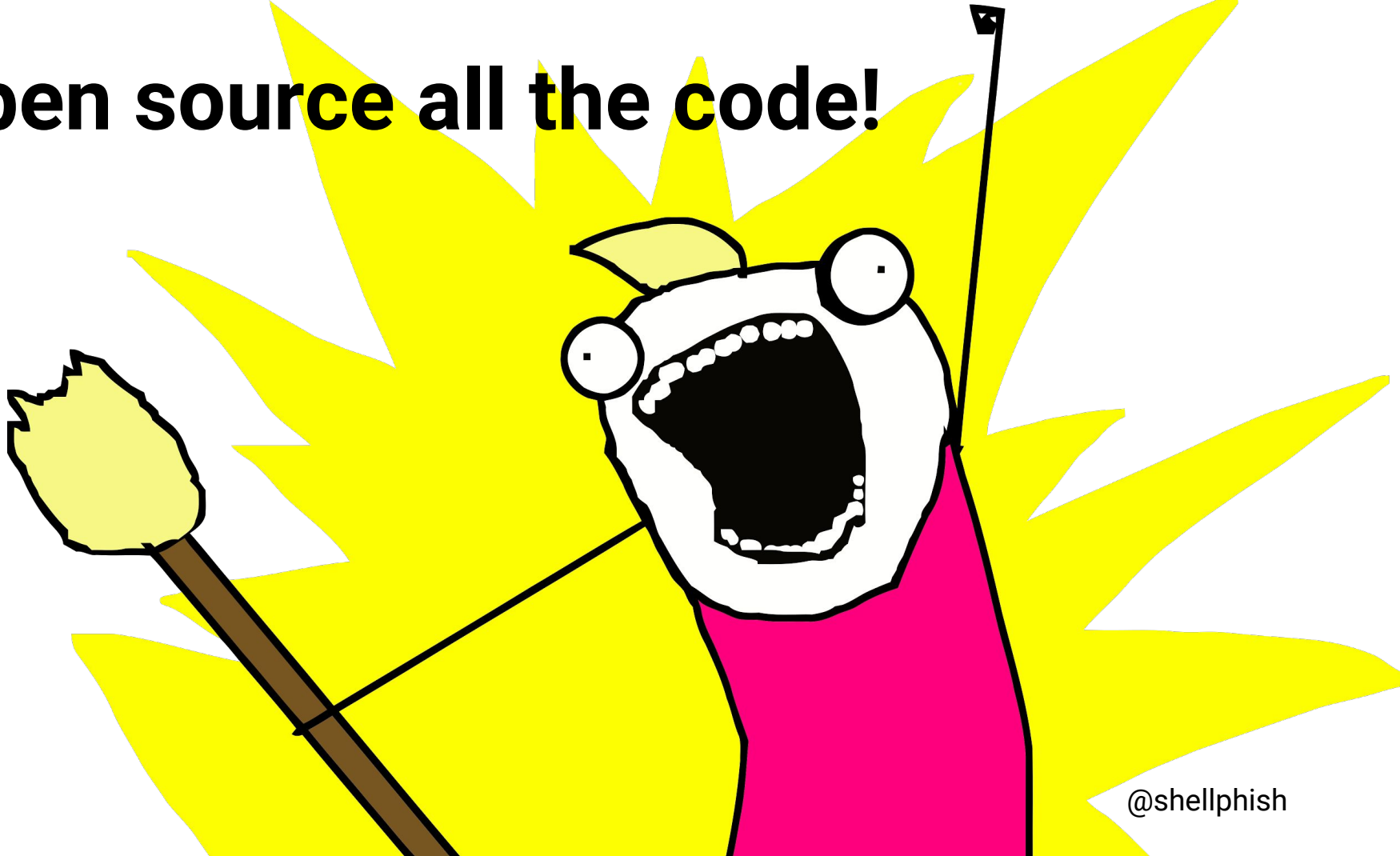
Awesome:

- CRS assisted with 5 exploits
- Human exploration -> CRS exploitation
- Backdoors!

Tough:

- API incompatibilities are brutal
- Computer programs are brittle

Open source all the code!



@shellphish

Stay in touch!

twitter: @Shellphish
email: team@shellphish.net or cgc@shellphish.net
irc: #shellphish on freenode

CRS chat: #shellphish-crs on freenode
angr chat: #angr on freenode

Backup

Conclusions

- Automated vulnerability analysis and mitigation is a growing field
- The DARPA CGC Competition is pushing the limits of what can be done in a **self-managed**, **autonomous** setting
- This is a first of this kind, but not the last
- … to the singularity!

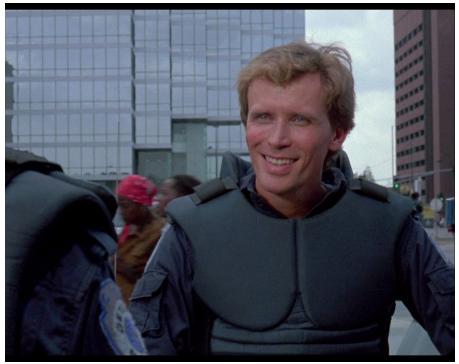
Self-Managing Hacking

- Infrastructure availability
  - (Almost) No event can cause a catastrophic downtime
    - Novel approaches to orchestration for resilience
- Analysis scalability
  - Being able to direct efficiently (and autonomously) fuzzing and state exploration is key
    - Novel techniques for state exploration triaging
- Performance/security trade-off
  - Many patched binaries, many approaches: which patched binary to field?
    - Smart approaches to security performance evaluation

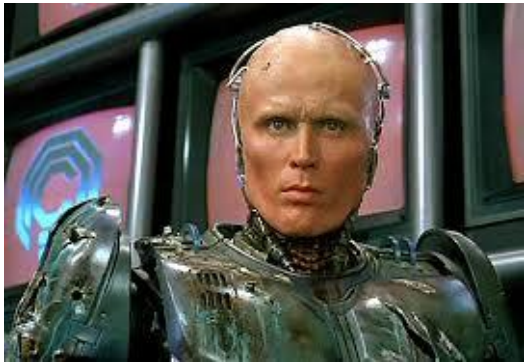
Hacking Binary Code

- Low abstraction level
- No structured types
- No modules or clearly defined functions
- Compiler optimization and other artifacts can make the code more complex to analyze
- WYSIWYE: What you see is what you execute

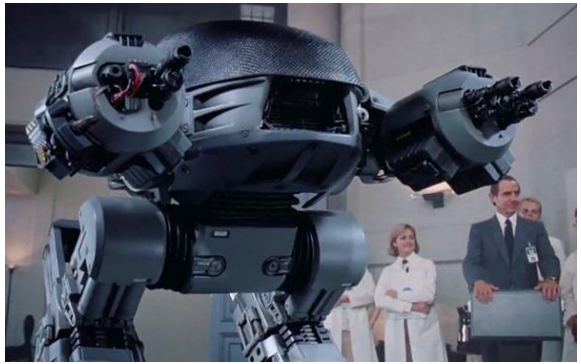
Finding Vulnerabilities



Human



Semi-Automated



Fully Automated

Manual Vulnerability Analysis

- “Look at the code and see what you can find”
- Requires substantial expertise
  - The analysis is as good as the person performing it
- Allows for the identification of complex vulnerabilities (e.g., logic-based)
- Expensive, does not scale

Tool-Assisted Vulnerability Analysis

- “Run these tools and verify/expand the results”
- Tools help in identifying areas of interest
  - By ruling out known code
  - By identifying potential vulnerabilities
- Since a human is involved, expertise and scale are still issues

Automated Vulnerability Analysis

- “Run this tool and it will find the vulnerability”
  - … and possibly generate an exploit...
  - ...and possibly generate a patch
- Requires well-defined models for the vulnerabilities
- Can only detect the vulnerabilities that are modeled
- Can scale (not always!)
- The problem with halting…

Vulnerability Analysis Systems

- Usually a composition of static and dynamic techniques
- Model how attacker-controlled information enter the system
- Model how information is processed
- Model a number of unsafe conditions

Static Analysis

- The goal of static analysis techniques is to characterize all possible run-time behaviors over all possible inputs without actually running the program
- Find possible bugs, or prove the absence of certain kinds of vulnerabilities
- Static analysis has been around for a long while
  - Type checkers, compilers
  - Formal verification
- Challenges: soundness, precision, and scalability

Example Analyses

- Control-flow analysis: Finds and reasons about all possible control-flow transfers (sources and destinations)
- Data-flow analysis: Reasons about how data flows within the program
- Data dependency analysis: Reasons about how data influences other data
- Points-to analysis: Reasons about what values can pointers take
- Alias analysis: Determines if two pointers might point to the same address
- Value-set analysis: Reasons about what are the set of values that variables can hold

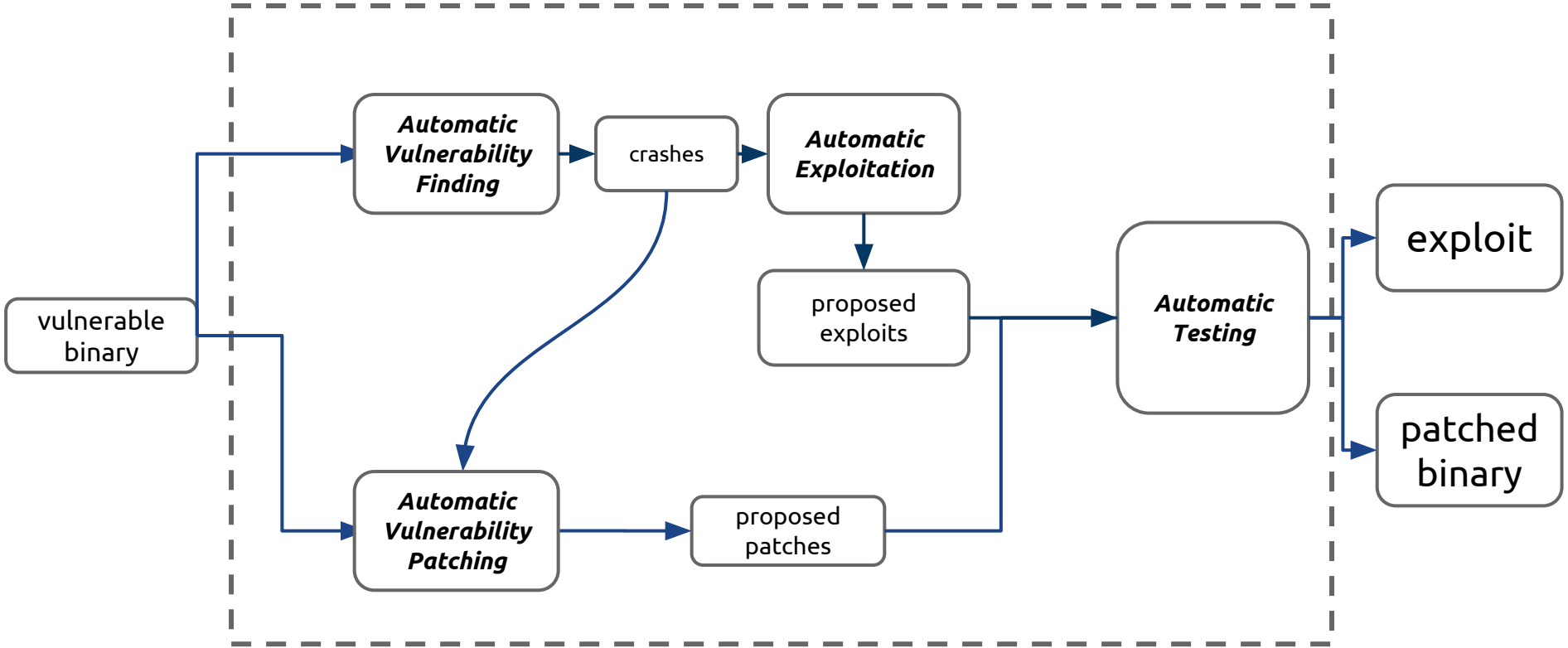
Dynamic Analysis

- Dynamic approaches are very precise for particular environments and inputs
  - Existential proofs
- However, they provide no guarantee of coverage
  - Limited power

Example Analyses

- Dynamic taint analysis: Keeps track of how data flows from sources (files, network connections) to sinks (buffers, output operations, database queries)
- Fuzzing: Provides (semi)random inputs to the program, looking for crashes
- Forward symbolic execution: Models values in an abstract way and keeps track of constraints

The Shellphish CRS: Mechanical Phish



Interactive, Online CTFs

- Very difficult to organize
- Require substantial infrastructure
- Difficult to scale
- Focused on both attacking and defending in real time
- From ctftime.org: 100+ events listed
- Online attack-defense competitions:
  - UCSB iCTF 13 editions
  - RuCTF 5 editions
  - FAUST 1 edition

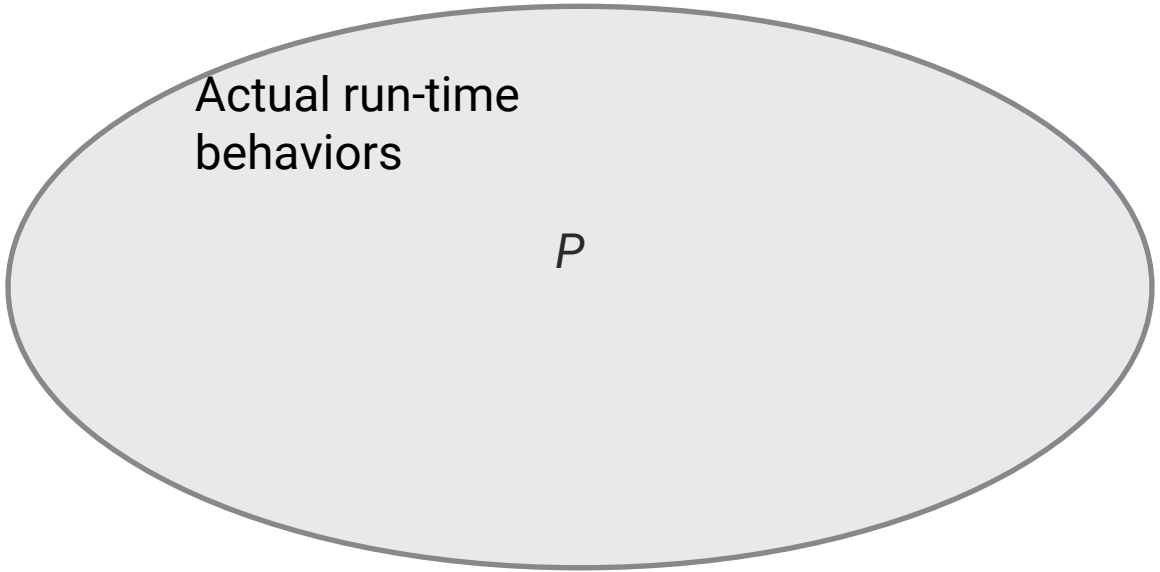
CTFs Are Playgrounds…

- For people (hackers)
- For tools (attack, defense)
- But can they be used to advance science?

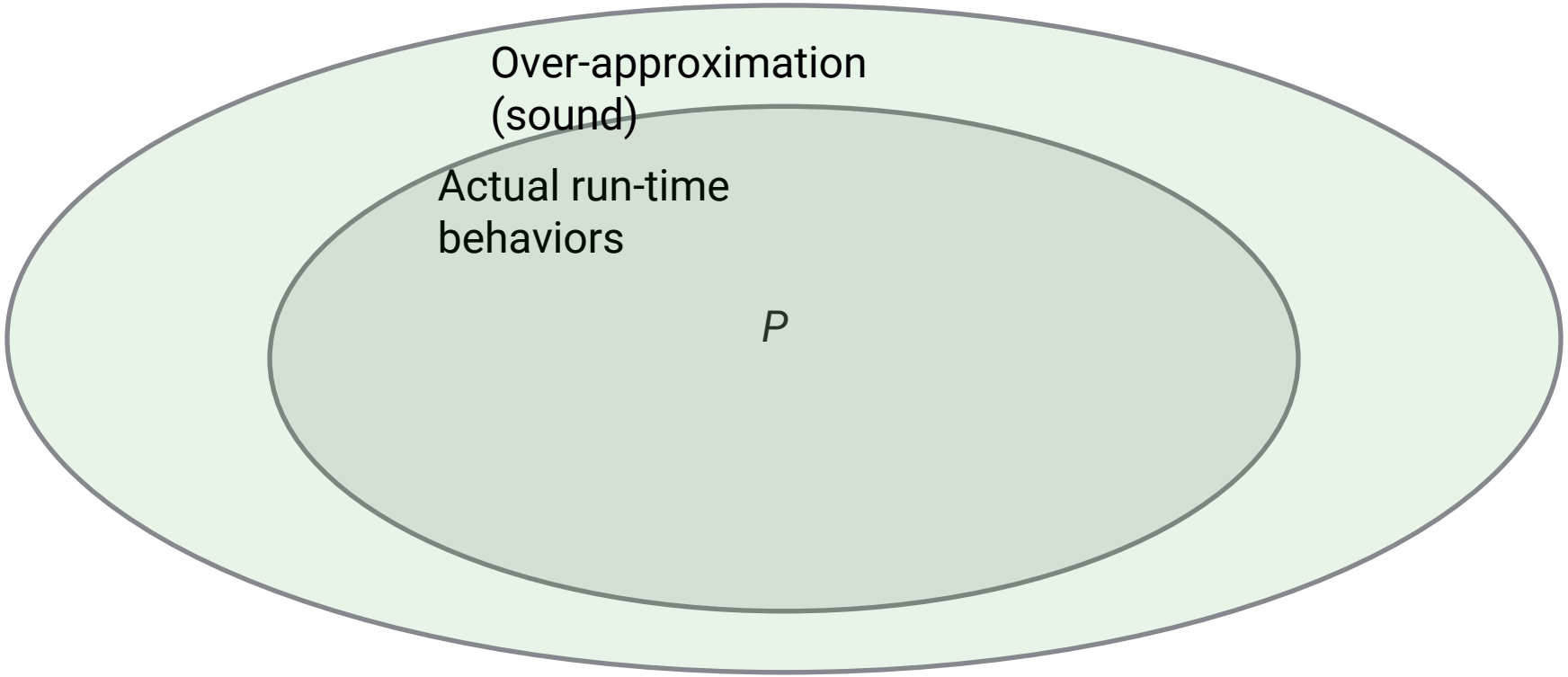
DECREE API

- `void _terminate(unsigned int status);`
- `int allocate(size_t length, int prot, void **addr);`
- `int deallocate(void *addr, size_t length);`
- `int fdwait(int nfds, fd_set *readfds, fd_set *writefds,`
  `struct timeval *timeout, int *readyfds);`
- `int random(void *buf, size_t count, size_t *rnd_bytes);`
- `int receive(int fd, void *buf, size_t count,`
  `size_t *rx_bytes);`
- `int transmit(int fd, const void *buf, size_t count,`
  `size_t *tx_bytes);`

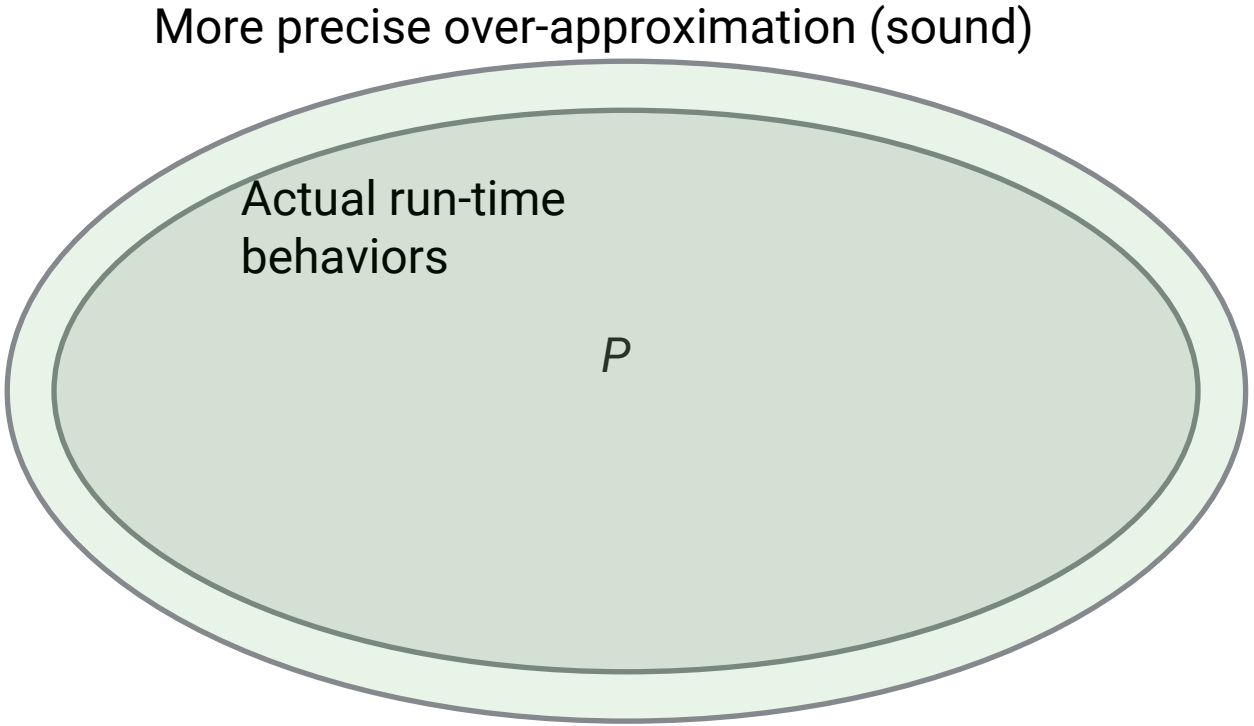
Soundness and Completeness



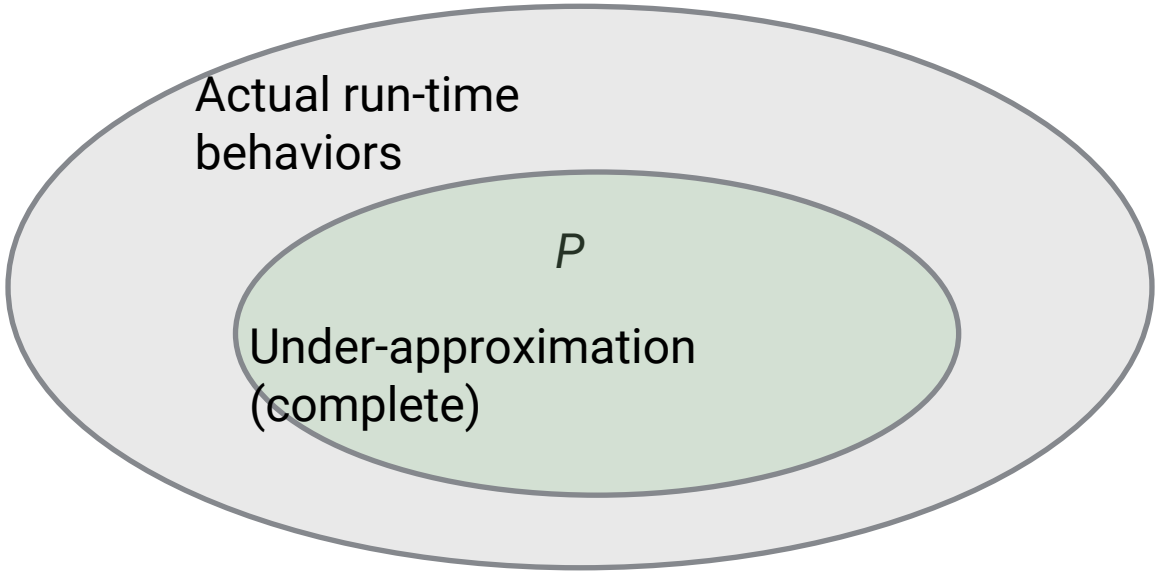
Soundness and Completeness



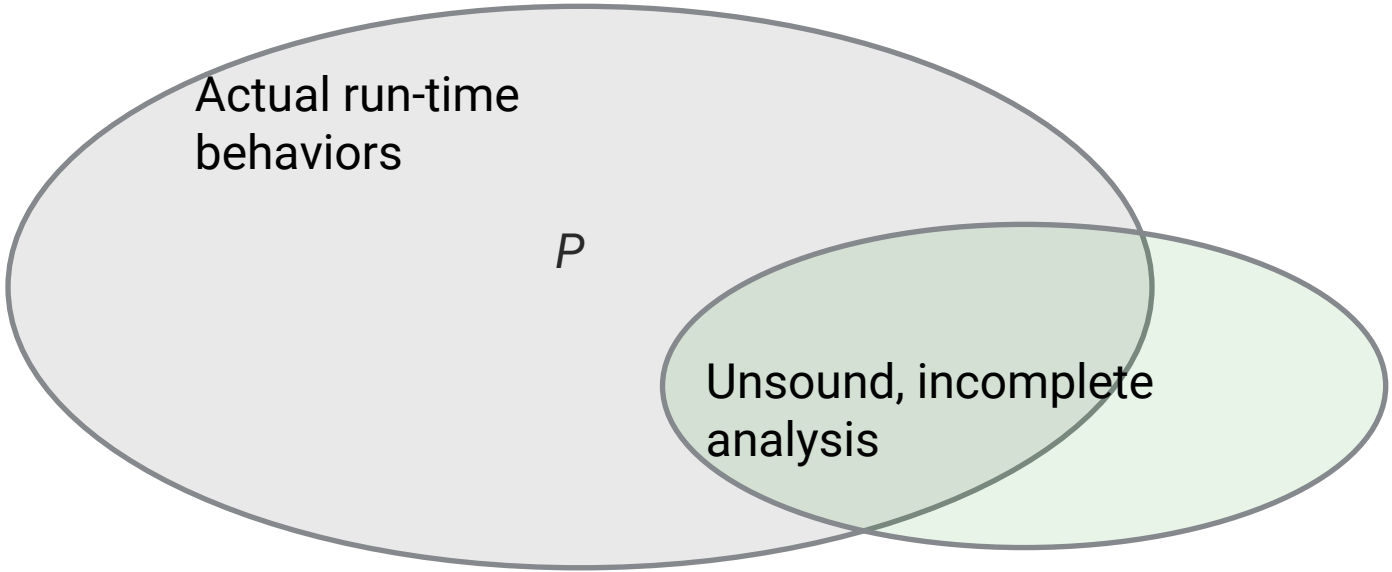
Soundness and Completeness



Soundness and Completeness



Soundness and Completeness

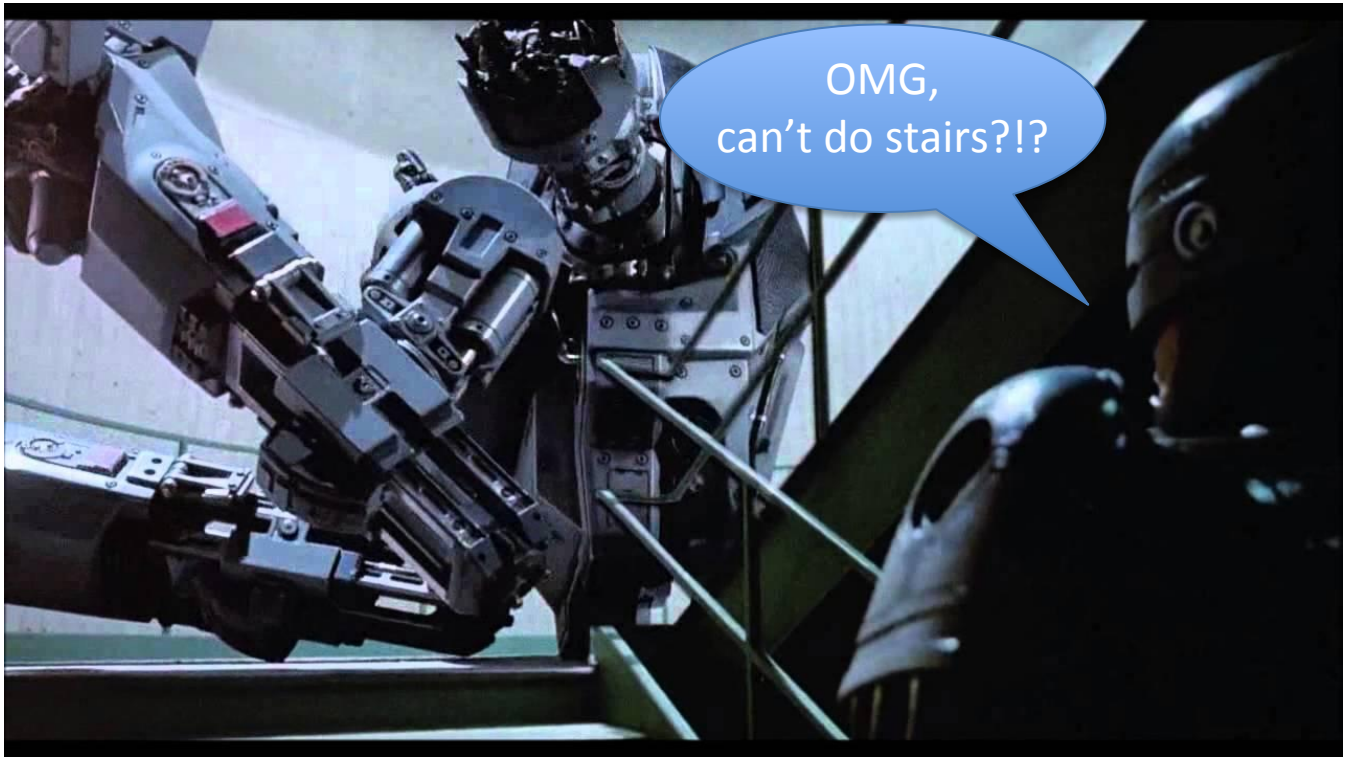


Hidden

Changed with "All the things" meme

Open the source!

Human + Machine = WIN!

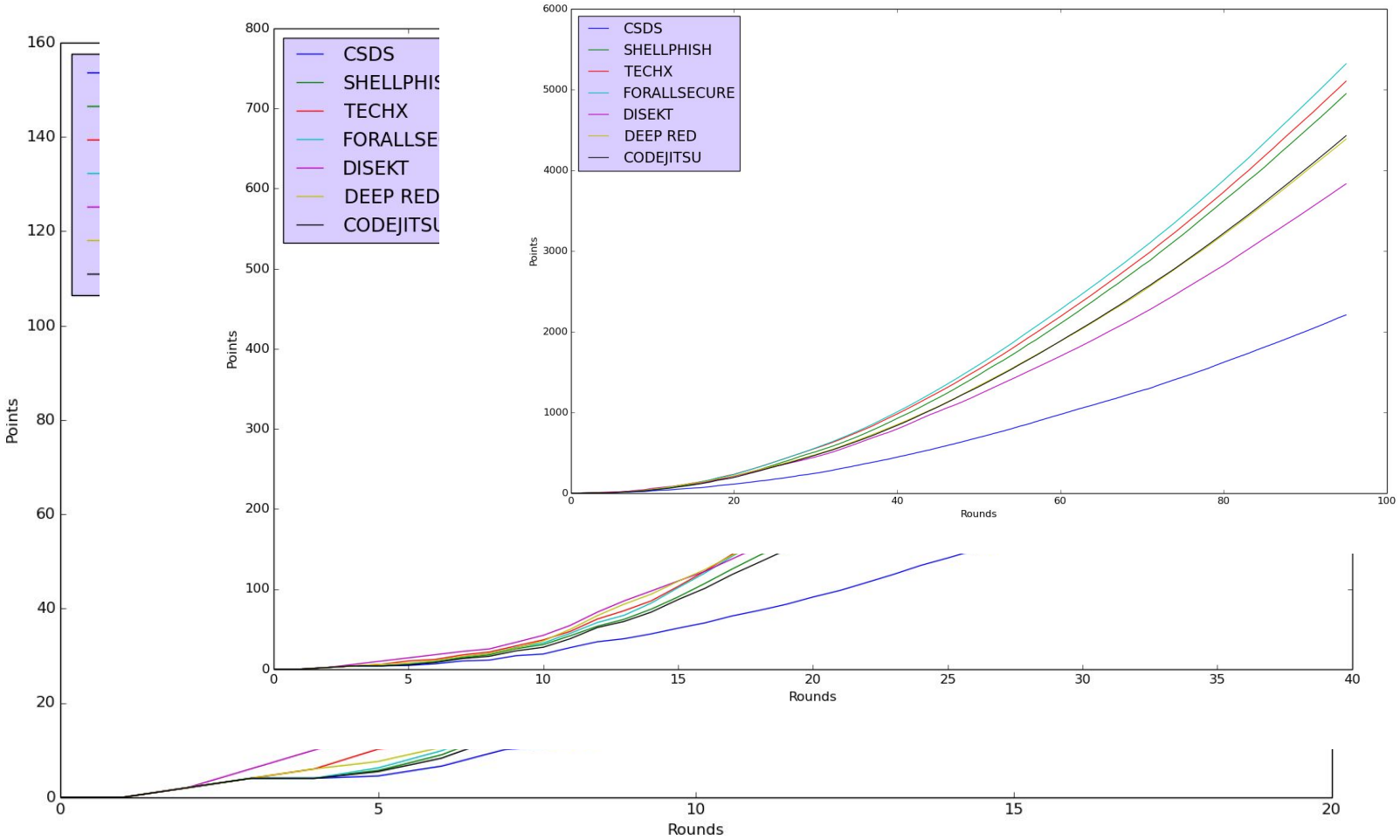


Simulation For Team Shellphish

- R00: Competition fields CB1, CB2, CB3
- R01: CRS generates PoV1, RB2
  - Points for round 00:
    - (CB1, CB2, CB3): Availability=1, Security=2, Evaluation=1 → Score = 2
    - Total score: 6
- R02: Competition fields CB1, RB2, CB3
  - Points for round 01
    - CB1: Availability=1, Security=1, Evaluation= 1+(6/6) →Score = 2
    - RB2: 0
    - CB3: Availability=1, Security=2, Evaluation=1 → Score = 2
    - Total score: 4

Simulation For Team Shellphish

- R03: Competition fields CB1, RB2, CB3
  - Points for round 02
    - CB1: Availability=1, Security=1, Evaluation=1+(3/6) → Score = 1.5
    - RB2: Availability=0.8, Security=2, Evaluation=1 → Score = 1.6
    - CB3: Availability=1, Security=2, Evaluation=1 → Score = 2
    - Total score: 5.1


CSDS
SHELLPHISH
TECHX
FORALLSECURE
DISEKT
DEEP RED
CODEJITSU
Points
Rounds