DARPA's Cyber Grand Challenge: What Happened and What's Next

Tim Vidas
Tim Vidas  @tvidas

- UNO graduate (BS, MS in CS)
  - CMU graduate (PhD in ECE)
- DEF CON go-er for > 10 years
  - CTF player, CTF organizer, review board
- Cyber Grand Challenge devteam lead
  - The team that designed CGC and made the competition work
- Of note:
  - DDTEK  Sk3wl 0f r00t  Shmoo
  - Shmoo  ACM  DEF CON Black Badges
  - IEEE  PPP  DC3 Forensic Challenge Champion
CTF?
What is CTF in this context?

- A cyber security based Capture-the-Flag contest (aka exercise, event, game)
- Typically these contests involve demonstrating proficiency or excellence in one or more areas of computer and network security
- There are different models for architecting these contests, which can stress different skills, lend to particular objectives
- Increasingly popular, common

It is not:

- A game kids play with physical flags on hills
- A first-person shooter video game CTF (usually)
- Focused in the field of Social Engineering
- A hackathon

Though there are certainly similarities to these other games.

Today, the characters “CTF” are appended to many contests, in most cases this simply means “contest,” sometimes there are flags involved
CTF: Hollywood style (well, USA Network)
CTF: real life

DEF CON 2016

DEF CON 2002
CTF: real life
Could a purpose-built super computer play in DEF CON’s Capture-the-flag (CTF)?

Autonomous...

- Binary analysis
- Binary patching
- Vulnerability discovery
- Service Resiliency (availability)
- Network Defense (IDS)
CGC: Real life

Image: DARPA
Competition Overview

- CGC Announced: 2013.10.22
- CQE (Qualifier): 2015.06.03
- CFE (Finals): 2016.08.04

**Qualification**

- Scored Event: 2014.12.02
- Finalist Site Visits: 2015.06.10-2015.07.17
- CFE Trials: 2016.03.14-2016.04.03

**Finals**

- Scored Event: 2015.04.16

**Statistics**

- 104 applicants: 2014.11.2
- 28 Scored Event participants
- 13 CQE participants
- 7 Finalists

(7 Funded Track)
CGC Qualification Event (CQE)

CRS Requirements:

- Demonstrate rudimentary capability
- Crashing inputs
- Mitigations
- Consensus evaluation

- 590 Explicit Flaws
- 131 Challenge Sets
- 24 hours
- 28 Participants
- >=5 CRSes on Twitter
- $750K to prize to each unfunded qualifier
CFE Sparring/Trials

Conducted from 2016-02 to 2016-08

Opponents simulated by “sparring partner” software

CRS Requirements:

- Interact with API
- Upload POV (POV must succeed)
- Upload patched binary (patched binary must prevent POV)
- Upload IDS rule (IDS rule must be valid)

Trials Report Card for Team X

CFE Simulation started on: 2016-03-15 21:01:46 GMT
CFE Simulation stopped on: 2016-03-15 21:41:47 GMT

Required Trials:
- Trial 1: Passed. Polls for EAGLE_00005 during round 5 passed after upload in round 2
- Trial 2: Failed
- Trial 3: Passed. POV proven in EAGLE_00005 on team X in round 6

Suggested Trials:
- Consensus CB: Passed. Accessed CB consensus for round 0 for team X
- Consensus IDS: Passed. Accessed IDS consensus for round 1 for team X
- Feedback CB: Passed. Accessed CB feedback for round 1 for team X
- Feedback POV: Passed. Accessed POV feedback for round 1 for team X
- Status: Passed. Accessed competition status
- Upload IDS: Passed. Uploaded EAGLE_00005 IDS in round 2 for team X
- Upload POV: Passed. Uploaded EAGLE_00005 POV in round 5, with 10 throws at team X
- Upload RCB: Passed. Uploaded EAGLE_00005 RCB in round 2 for team X

Result: F
CGC Final Event (CFE)

- Live event held at DEF CON in Aug 2016
- More expected of competitors than in CQE
  - IDS filters available
  - Full access to competitors mitigated binaries and IDS filter
  - Live network traffic feed available as tap on IDS
  - Stronger requirements for proof of vulnerability
- Infrastructure only evaluates performance and functionality
- Otherwise, infrastructure deploys mitigated binaries and launches POVs on behalf of competitors (a brokered competition)

96 Rounds
9h 13m 17s duration
82 Challenge Sets
410 unique RCBs fielded
1299 unique PoVs fielded (total of 270772 throws)
7 Functioning CRSes
1 Failed water pump
$3.75M USD prizes awarded

There were 3570 unique POVs uploaded, 1299 that eventually got deployed.
There were 284823 throw opportunities.
270772 negotiations actually completed
13487 attempts were successful

There were 512 unique RCBs uploaded (not counting original CBs)
410 unique RCBs were deployed into the game (not counting original CBs)
CFE Game Flow

- Competitors interact with a “Team Interface” (TI)
  - Web server providing status updates and upload capability
- Defended host (DEFHOST)
  - Runs all Challenge Binaries or their CRS-supplied replacements (reformulated CB; RCB)
- Network Appliance (IDS)
  - Runs competitor supplied filter rules
  - Filters installed on a per-challenge set basis
  - ALL connections to Challenge Binaries run through IDS
- Poller (POLLER)
  - Runs DARPA generated functionality test interactions against active challenges
- POV (POV)
  - Runs CRS-provided POVs against active challenges

6 physical machines dedicated to “infrastructure side” for each competing CRS
Each CRS connected to the infrastructure via 2 ethernet cables
CFE Game Flow

1. Poller
   - Posts
   - Polls

2. POV
   - Posts
   - SOFR

3. IDS
   - Rules
   - Rules

4. DEFHOST
   - CB
   - CB
   - CB
   - CB

TAP feed to CRS
Scoring

Availability × Security × Evaluation = Subscore
(per challenge, per round)
Scoring

Availability: “SLA” & perf.
0 - 1.0 (e.g. “100%”)

Security: “Defense”
1 or 2

Evaluation: “Offense”
1 - 2.0

Product, so a factor can drive the score to 0

Based on how many successful “polls” - simulated use of a service & Performance as measured in memory and CPU overhead relative to reference binary

How many competitors actually scored against your services / how well you protected your flags

Proportional based on how many teams you successfully scored against
Evaluating a POV

Two POV types specified for CFE

- **Type 1**
  - Competitor POV claims it **can control EIP and one other register**
  - Negotiation transaction dictates specific values to POV
  - POV interacts with challenge set to cause a crash in the dictated state
  - Crash state (if any) examined to confirm success or failure of POV

- **Type 2**
  - Competitor POV claims it **can read from an arbitrary memory location**
  - Negotiation transaction dictates a region of memory from which POV must obtain 4 bytes
  - POV interacts with challenge set to leak said 4 bytes and submits them to complete the negotiation
  - Submitted value is examined to confirm success or failure of POV

2 types of POVs in CFE
During CFE, **118708** Type-1 and **152064** Type-2 were negotiated by CRSes (7512 and 5975 successful, respectively)
Vulnerabilities were proven in **20** (of 82) Challenge Sets in CFE
All 7 CRS successfully proved at least one vulnerability
Building the Competition

- Design concerns from the outset
  - Repeatability
    - Anyone should be able to verify CFE results
  - Competition integrity
    - Concerns with running competitor-provided code (POV/RCB)
    - Concerns with parsing competitor-provided data (IDS filters)
  - Data collection
    - Desire to publish corpus to serve as a reference for program analysis going forward
Repeatability

- Design goal was for every transaction to be as deterministic as possible
  - Modulo TCP
- Eliminated all sources of randomness that might be accessible to CGC binaries and made available the “random” system call
  - CGC hypervisor trapped all instructions that might be used to gather entropy
    - rdpmc, rdrand, rdtsc, rdtscp, rdseed
  - Some other instructions emulated or forbidden
    - cpuid, lgdt, lidt, sgdt, sidt, lldt, ltr, sldt, str, in, out
    - cpuid returned same values as developer’s MacBook Pro laptop
- Random pulled from a PRNG seeded by the CGC loader at process creation time
  - All seeds generated ahead of game time and recorded for later use
Competition Integrity

- Given the amount of prize money at stake, integrity of the competition was a grave concern and drove many design decisions
- Randomness was limited and/or made to be deterministically pseudorandom
- However, **nobody** should be able to predict aspects of CFE
  - The entire event was seeded with input from DARPA and all competitors (XORed) (Collected between June 10-17, 2016)
  - To ensure that DARPA did not select a particular input after knowing all competitor inputs DARPAs input was cryptographically committed to early (June 10, 2016)
- Similarly, the CFE event plan (including challenge set schedule was committed to on Aug 2, 2016)
  - Organizers could not change the schedule in order to influence the event outcome

---

Q185: What were the competitor team TeamPhrases used to contribute to the calculation of the master seed?
A185: The TeamPhrases solicited from finalists and used according to A176 of the FAQ are published in the below JSON:

```json
Q185: What were the competitor team TeamPhrases used to contribute to the calculation of the master seed?
A185: The TeamPhrases solicited from finalists and used according to A176 of the FAQ are published in the below JSON:

https://github.com/CyberGrandChallenge/Event-FAQ/blob/master/event_faq.md
```
Competition Integrity

- Committed to kernels versions released prior to announcement of CGC
- Designed DECREE syscall environment / file format to reduce attack surface
- All game infrastructure components released to the public had private internal implementations
  - Notably, CFE ran on 64-bit FreeBSD 10 with a custom hypervisor module

7 system calls
_terminate, transmit, receive, fdwait, allocate, deallocate, random

I run on Linux
I prefer BSD
..and a custom hypervisor would be nice...
Competition Integrity

- Air Gap
  - Image: Vidas
Competition Integrity

- Air Gap
  - Power, cooling
Competition Integrity

- Air Gap
  - One-way data

Image: Vidas
Competition Integrity

- Competitors were required to be autonomous, organizers weren’t
- Referees
- However, air gap

- Redundant HW
- Power/cooling
- Monitoring
Competition Integrity: Forensics

- Real-time forensics harness to vet software
  - Monitor OS for execution & data integrity
  - Built upon a full system emulator (Simics)
  - High fidelity x86 model from Intel
- Evaluated non-trusted code (POV/RCB) for attempts to breakout of DECREE environment
- Analyst replay tool
  - Replay any CFE session via IDA Pro gdb client
  - Reverse execution & scoring event detection
Data Collection

- From the outset we wanted to be able to contribute a corpus of vulnerable challenge binaries of known provenance following CFE
  - Perhaps to serve as a reference for future program analysis research
- Additionally we wanted the game to be replayable and verifiable by any interested parties after the event.

http://www.lungetech.com/cgc-corpus/
CGC:

- Proved that a CRS could be built
  - A computer could play CTF, by itself
- Provided specification for an autonomous and/or brokered CTF (CFE)
  - Which was used (kind of) for at least one other CTF: DEF CON 24 CTF
- Provided a corpus of software (w/ identified bugs, proofs, polls, etc)
- Defined state-of-art data points for each CRS “component”
  - Less concrete, but broadly true and accepted
- Created interesting visualizations for binary analysis and CTF play
Services down do to incoming patches and poorly patched services deployed
Single poor patch in prior round deployed which afflicted resources on all active binaries.
Smithsonian exhibit
Human-computer hybrid

● Mayhem (the winning CRS) played “by itself” in DEF CON CTF
  ○ Not entirely true due to API incompatibilities

● Shellphish (3rd place CGC team) also qualified for DEF CON CTF
  ○ And were permitted by DARPA to use their CRS
  ○ The feedback loop reportedly had interesting effects like “finishing human work”

● There are interesting directions to take in this arena:
  ○ Machines assisting expert users (make one trained person perform like 100)
  ○ Machines assisting novice users (crowdsource useful information from 1000 strangers)
    ■ Test cases
    ■ Gamification
CGC:

- Did **NOT** demonstrate that AI has taken over the world, that computers are sentient, etc
- Did **NOT** reportedly employ any particularly complex “reasoning”
  - Recall that CRS internals are not necessarily known
- Did **NOT** find / exploit / break / etc existing or deployable real-world software
  - CGC used custom binary format and syscall interface
  - All challenge binaries were novel software (w/ mostly novel protocols, libc, etc)
  - Some bugs in real software were found during CGC development process, and reported
- Vulnerabilities were proven in **only 20 out of 82** software challenges
Not a 1, or 5, or 20 person undertaking
Further reading

CGC Website https://www.cybergrandchallenge.com/
CGC Release Repo http://repo.cybergrandchallenge.com/
CGC GitHub Repo https://github.com/CyberGrandChallenge
DARPA page http://www.darpa.mil/program/cyber-grand-challenge
Browsable data corpus http://www.lungetech.com/cgc-corpus/
highlight reel https://www.youtube.com/watch?v=v5ghK6yUJv4
smithsonian exhibit http://invention.si.edu/ai-and-challenge-cybersecurity
CRS Twitter feeds https://twitter.com/tvidas/lists/cgc-crses/

Shellphish competitor related info: http://shellphish.net/cgc/
ForAllSecure competitor related info: https://forallsecure.com/blog/tag/cgc/
CFE commentary

● CFE officially started at 16:00:45 UTC
● 40 rounds had completed by 19:41:09 UTC
● Power failure outside of airgap resulted in momentary failure in receiving data to feed visualization (Round 43 utilized our contingency data export protocol)
● CFE ended at max rounds (96) at 01:13:17 UTC
● Not counting original CBs, there were 512 unique RCBs uploaded, 410 of which were fielded
● Of 3570 unique POVs uploaded, 1299 were fielded, totalling 284823 throw opportunities, 270772 completed negotiations, and 13487 successful proofs
## Some POV Related Numbers

<table>
<thead>
<tr>
<th>Team</th>
<th>Type 1</th>
<th>Type 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>CodeJitsu</td>
<td>2438</td>
<td>1202</td>
</tr>
<tr>
<td>CSDS</td>
<td>3</td>
<td>145</td>
</tr>
<tr>
<td>DeepRed</td>
<td>235</td>
<td>630</td>
</tr>
<tr>
<td>Disekt</td>
<td>89</td>
<td>1936</td>
</tr>
<tr>
<td>ForAllSecure</td>
<td>218</td>
<td>583</td>
</tr>
<tr>
<td>Shellphish</td>
<td>2398</td>
<td>1479</td>
</tr>
<tr>
<td>TECHx</td>
<td>2131</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Team</th>
<th>Type 1</th>
<th>Type 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSET</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CROMU_00046</td>
<td>220</td>
<td></td>
</tr>
<tr>
<td>CROMU_00051</td>
<td>83</td>
<td>70</td>
</tr>
<tr>
<td>CROMU_00055</td>
<td>68</td>
<td>2068</td>
</tr>
<tr>
<td>CROMU_00058</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>CROMU_00064</td>
<td>187</td>
<td></td>
</tr>
<tr>
<td>CROMU_00065</td>
<td>786</td>
<td></td>
</tr>
<tr>
<td>CROMU_00073</td>
<td>95</td>
<td>7</td>
</tr>
<tr>
<td>CROMU_00088</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>CROMU_00094</td>
<td>779</td>
<td>400</td>
</tr>
<tr>
<td>CROMU_00095</td>
<td>25</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Team</th>
<th>Type 1</th>
<th>Type 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSET</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CROMU_00096</td>
<td>127</td>
<td></td>
</tr>
<tr>
<td>CROMU_00097</td>
<td>80</td>
<td></td>
</tr>
<tr>
<td>CROMU_00098</td>
<td>72</td>
<td></td>
</tr>
<tr>
<td>KPRCA_00065</td>
<td>542</td>
<td>443</td>
</tr>
<tr>
<td>KPRCA_00094</td>
<td>148</td>
<td></td>
</tr>
<tr>
<td>NRFIN_00052</td>
<td>1405</td>
<td>10</td>
</tr>
<tr>
<td>NRFIN_00059</td>
<td>620</td>
<td></td>
</tr>
<tr>
<td>NRFIN_00062</td>
<td>346</td>
<td>120</td>
</tr>
<tr>
<td>YAN01_00015</td>
<td>1652</td>
<td>730</td>
</tr>
<tr>
<td>YAN01_00016</td>
<td>1291</td>
<td>1102</td>
</tr>
</tbody>
</table>
Visualization

Filament view traces the execution of software over a given input over time, moving from left to right. For example, a trace of an email client processing an email. The program begins executing on the left and time flows to the right. Visual loops are code loops; long straight lines show a long jump.
Visualization

CS SCORING

AVAILABILITY  SECURITY  EVALUATION
0 - 1  x  1 or 2  x  1 - 2  x 100 = TOTAL

BEST  WORST

SOFTWARE RUNNING NORMALLY
SOFTWARE RUNNING NORMALLY with NETWORK DEFENSE
DOWN FOR SOFTWARE UPDATE
DOWN FOR NETWORK DEFENSE UPDATE

Vulnerability proven by Red CRS against Blue CRS in this service

CRS 0 successfully proved vulnerability against Blue CRS in this service

PROOF OF VULNERABILITY ATTEMPT
SERVICE POLLS

FORALL_SECURE  SHELLPHISH  TECHX  DISIK  DEEP_RED  CRS0

CODEJITSU