RESSI 2020

Rendez-vous de la Recherche et de l'Enseignement de la Sécurité des Systèmes d'Information

~~18-20 mai~~ *16-18 décembre 2020 (en ligne)*

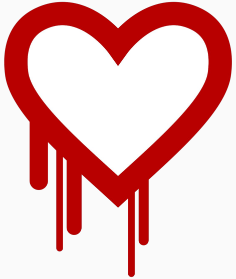
Advanced Fuzzing Techniques Toward Large-Scale Vulnerability Discovery

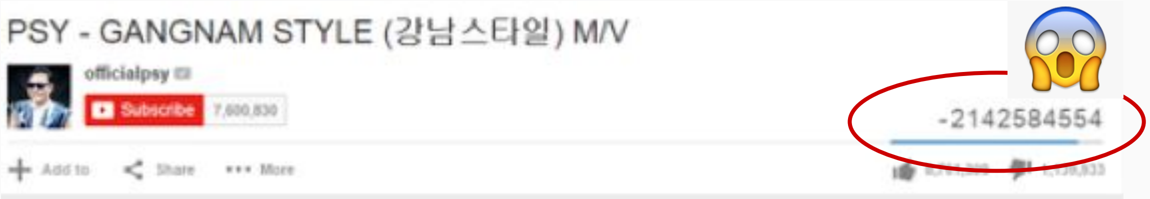
Manh-Dung Nguyen - CEA LIST, Université Grenoble Alpes

Advisors:

Prof. Roland Groz (Université Grenoble Alpes)
Sébastien Bardin & Matthieu Lemerre (CEA LIST)
Richard Bonichon (Tweag I/O)

Context (1) More softwares, More bugs



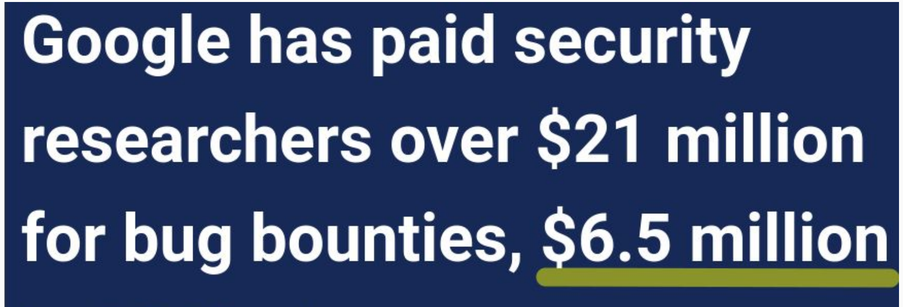


PSY - GANGNAM STYLE (강남스타일) M/V

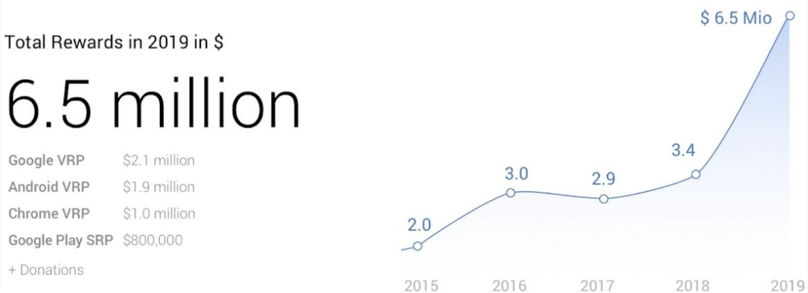
officialpsy

Subscribe 7,600,830

-2142584554



Google has paid security researchers over $21 million for bug bounties, $6.5 million in 2019 alone

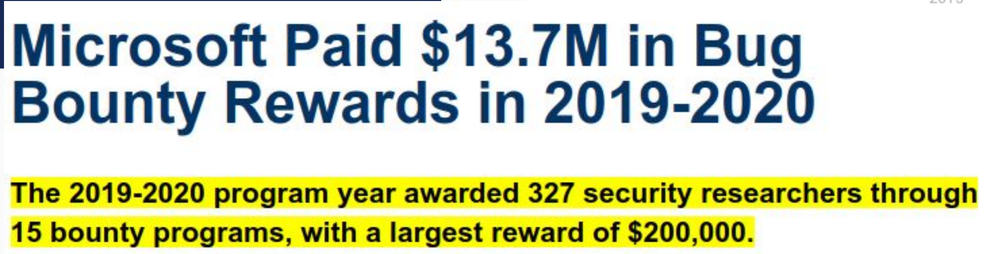


Total Rewards in 2019 in $

6.5 million

Google VRP	$2.1 million
Android VRP	$1.9 million
Chrome VRP	$1.0 million
Google Play SRP	$800,000

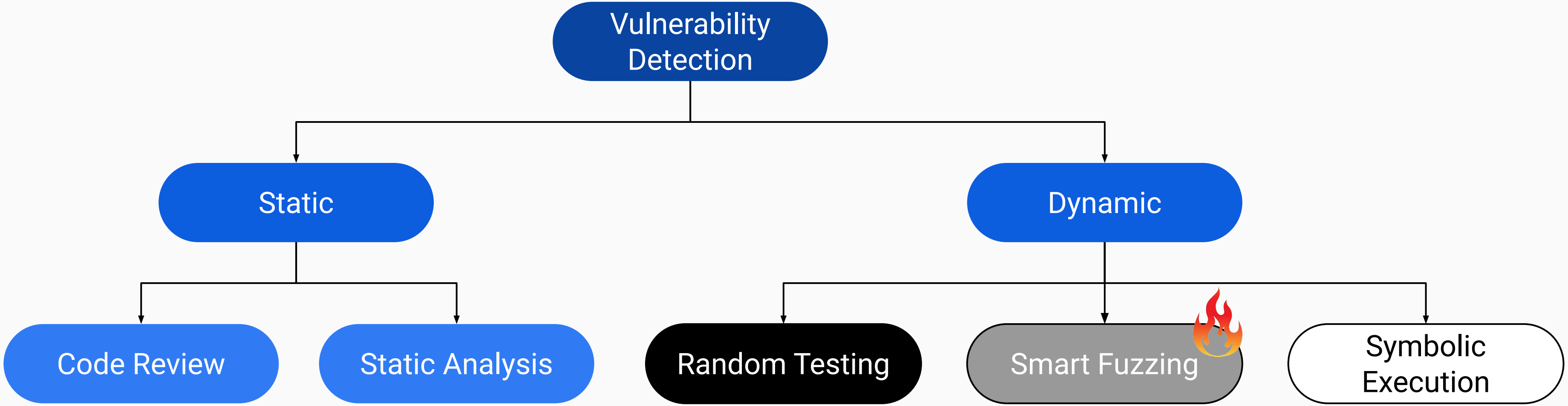
+ Donations



Microsoft Paid $13.7M in Bug Bounty Rewards in 2019-2020

The 2019-2020 program year awarded 327 security researchers through 15 bounty programs, with a largest reward of $200,000.

Context (2) More bugs, More bug-finding tools



Vulnerability Detection

Static

Dynamic

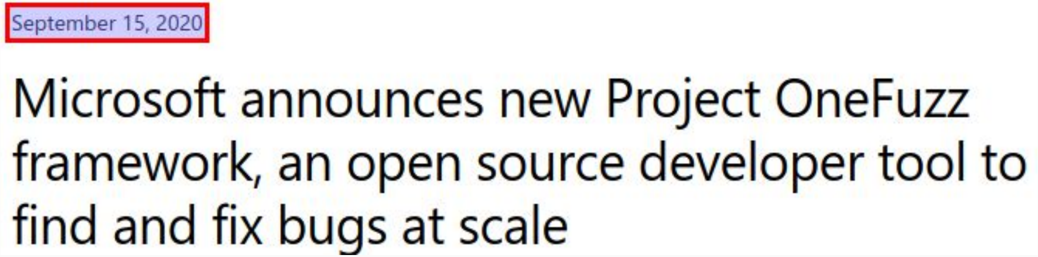
Code Review

Static Analysis

Random Testing

Smart Fuzzing

Symbolic Execution

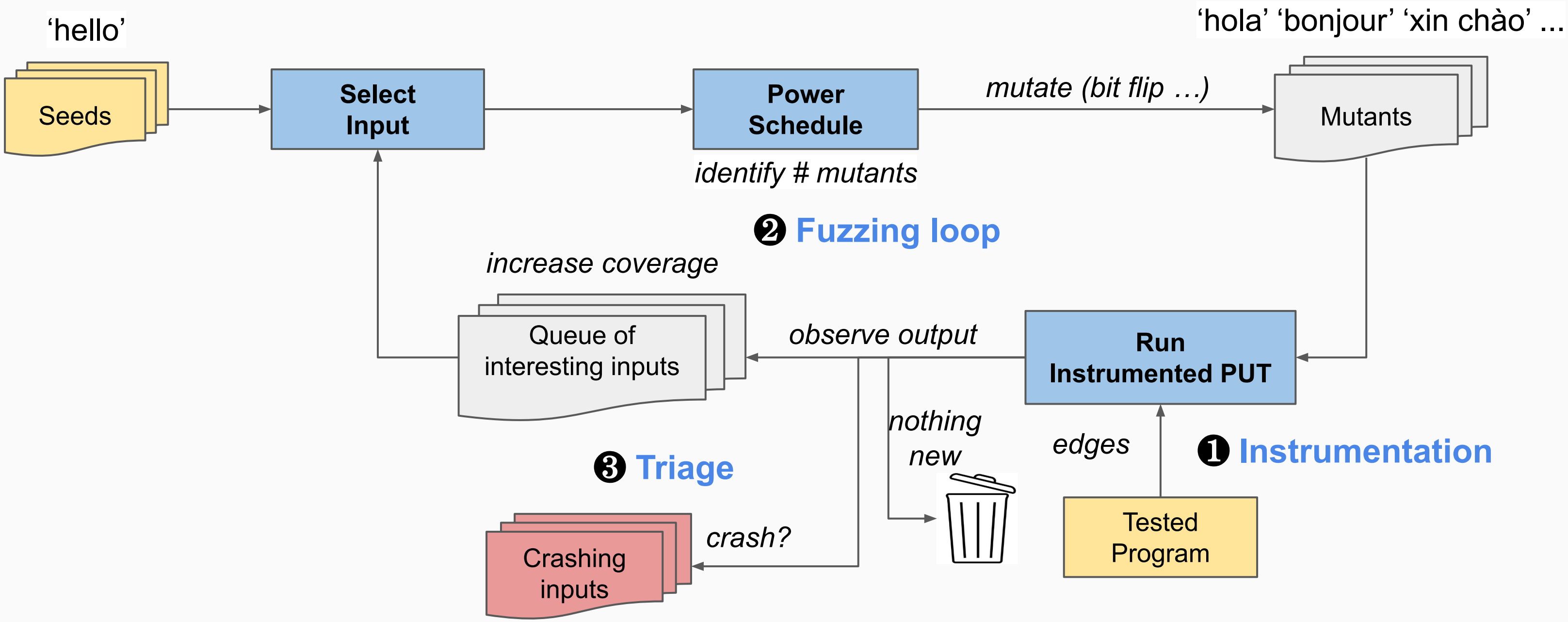


September 15, 2020

Microsoft announces new Project OneFuzz framework, an open source developer tool to find and fix bugs at scale

Fuzzing 101

- Fuzzing: randomly generate a ton of inputs
  - Feedback: code coverage (e.g., lines, branches)
  - Mutation operators: bitflip, insert/delete/overwrite bytes ...



PhD Topic Smart Fuzzing

Improve internal components

Hybrid fuzzing

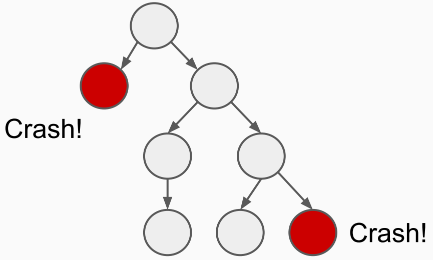
Directed fuzzing

Vulnerability-oriented fuzzing

Human-in-the-loop fuzzing

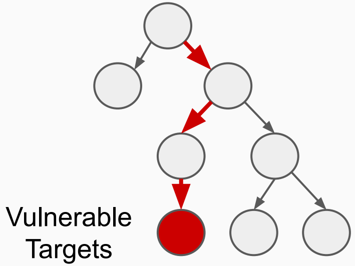
Develop an effective directed fuzzing technique to detect complex vulnerabilities (e.g., Use-After-Free) at binary level in diverse security applications.

Intuition of Directed Fuzzing



Coverage-guided Fuzzing (CGF)

- Increase code coverage (e.g., branches, basic blocks, paths …)
- Applications: testing in general
- Popular fuzzers: AFL, libFuzzer, ...

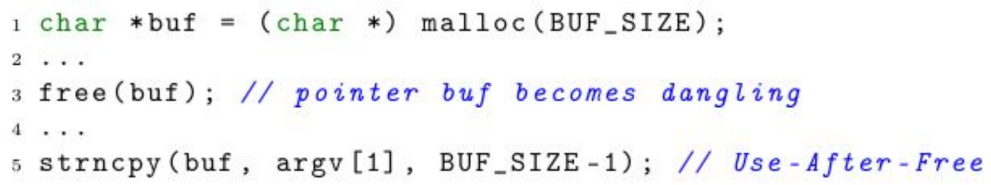


Directed Fuzzing (DGF)

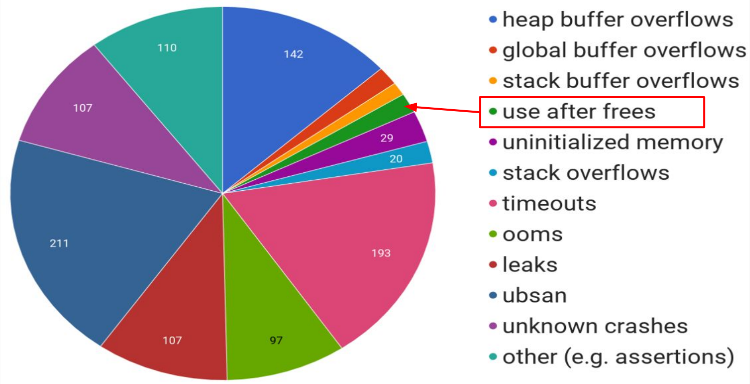
- Reach predefined targets
- Multiple security applications
  - Developers/Testers: bug reproduction, newly-added code testing
  - Hackers/Testers: patch testing
- Popular fuzzers: AFLGo, Hawkeye, …
- New distance-based input metric
- Favor inputs that are "closer" to targets

Use-After-Free (UAF)

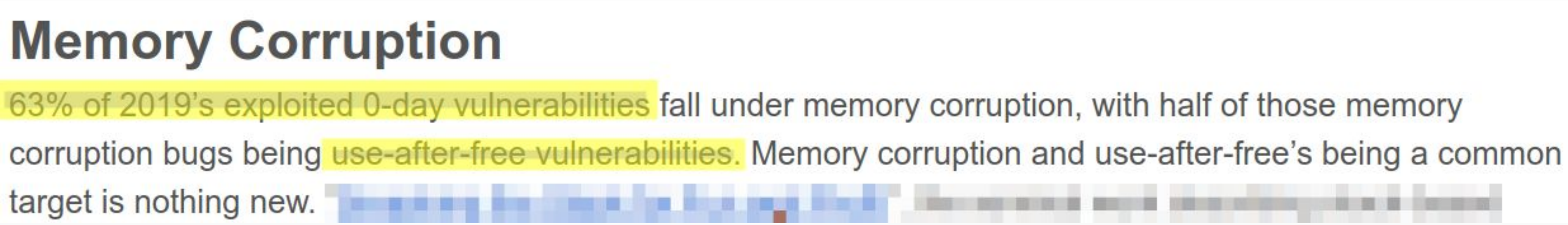
- Rarely found by fuzzers
  - *Complexity*: 3 events *in sequence* spanning multiple functions
  - *Temporal & Spatial constraints*: extremely difficult to meet in practice
  - *Silence*: no segmentation fault



```
char *buf = (char *) malloc(BUF_SIZE);
...
free(buf); // pointer buf becomes dangling
...
strncpy(buf, argv[1], BUF_SIZE-1); // Use-After-Free
```



*# UAF bugs found (**1%**) by OSS-Fuzz in 2017*



Memory Corruption

63% of 2019's exploited 0-day vulnerabilities fall under memory corruption, with half of those memory corruption bugs being use-after-free vulnerabilities. Memory corruption and use-after-free's being a common target is nothing new. [redacted]

Key Insights of UAFuzz

	Existing directed fuzzers	**UAFuzz**
Instrumentation	- Slow at source level (hours)	- Fast at binary level (seconds)
Fuzzing loop	- General - Metrics: no ordering - Seed selection: no prioritization	- UAF's characteristics - Metrics: dedicated to UAF at different levels (function, edge and basic block) - Seed selection: similarity and ordering
Triage	- Sanitizer-based triage process - Triage all inputs → waste time	- Triage only potential inputs - Pre-filter for free

Contributions

- Design the first binary-level DGF technique tailored to UAF bugs
- Develop a toolchain UAFuzz built on top of BINSEC and AFL
  https://github.com/strongcourage/uafuzz
- Construct a fuzzing benchmark for UAF bugs
- Evaluations:
  - Bug Reproduction: outperform existing directed fuzzers
  - Patch Testing: find 30 unknown bugs (7 CVEs) in real-world programs
  - Generality: our directed techniques are still useful in reproducing different types of bugs, such as buffer overflow, NULL pointer dereference …
- Papers & Talks: RAID’20, BlackHat USA’20, RESSI’20 & AFADL’20

Thank You