

USENIX Security '25 Artifact Appendix: Attacker Control and Bug Prioritization

Guilhem Lacombe Université Paris-Saclay, CEA, List, France guilhem.lacombe.97@gmail.com

A Artifact Appendix

A.1 Abstract

As bug-finding methods improve, bug-fixing capabilities are exceeded, resulting in an accumulation of potential vulnerabilities. There is thus a need for efficient and precise bug prioritization based on exploitability. In this work, we explore the notion of control of an attacker over a vulnerability's parameters, which is an often overlooked factor of exploitability. We show that taint as well as straightforward qualitative and quantitative notions of control are not enough to effectively differentiate vulnerabilities. Instead, we propose to focus analysis on feasible value sets, which we call domains of control, in order to better take into account threat models and expert insight. Our new Shrink and Split algorithm efficiently extracts domains of control from path constraints obtained with symbolic execution and renders them in an easily processed, human-readable form. This in turn allows to automatically compute more complex control metrics, such as weighted Quantitative Control, which factors in the varying threat levels of different values. Experiments show that our method is both efficient and precise. In particular, it is the only one able to distinguish between vulnerabilities such as cve-2019-14192 and cve-2022-30552, while revealing a mistake in the human evaluation of cve-2022-30790. The high degree of automation of our tool also brings us closer to a fully-automated evaluation pipeline.

This artifact contains the necessary material to reproduce all experimental results from the paper, including the source code of our tool Colorstreams, our benchmarks and scripts to run experiments and analyze results. In addition, we provide tutorials and documentation for Colorstreams.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

This artifact does not pose any threat to the security and privacy of evaluators.

Sébastien Bardin Université Paris-Saclay, CEA, List, France sebastien.bardin@cea.fr

A.2.2 How to access

The artifact is openly available on Zenodo: https://doi.org/10.5281/zenodo.14699098.

A.2.3 Hardware dependencies

There are no specific hardware requirements but we recommand having a CPU with at least 16 cores.

A.2.4 Software dependencies

Any OS that can run docker is suitable. For reproducing the experiments without the docker image, any Linux distro compatible with the Nix package manager should be suitable, although the artifact was only tested on Ubuntu.

Here is a list of Colorstreams' main software dependencies, which are automatically handled through Nix and thus **do not need to be installed manually**:

Binsec: symbolic execution Intel PIN: tracing Z3: SMT solver Bitwuzla: SMT solver Q3B: SMT solver popcon: model counting, includes d4 ganak: model counting approxmc: model counting gdb: debugging

Note that this list does not include sub-dependencies and the (many) benchmark dependencies. For more details, check out the artifact's Nix files (*nix/sources.json* in particular).

A.2.5 Benchmarks

The experiments require two benchmarks. The first was assembled by us and the second is based on the MAGMA fuzzing benchmark (https://hexhive.epfl.ch/magma/). The artifact contains both of them with documentation; no manual setup is required.

A.3 Set-up

A.3.1 Installation

With docker: Download *docker_image.tar.gz* from Zenodo, then run *docker image load < docker_image.tar.gz*.

To start a container, run *docker run -it -rm colorstreams:usenix25*. We recommend setting up a shared volume to easily transfer files from the container, such as figures. To do so, add *-v <local_dir>:<mount_point>* to the container startup command (paths must be absolute). For example, the full command could be *docker run -it -rm -v /home/aaa/bbb:/share colorstreams:usenix25*. Alternatively, *docker cp* can also be used, although it is less convenient.

Without docker: Given the amount of dependencies, we only provide compilation through Nix. This should also help to future-proof the building process. To install Nix, follow the instructions at https://nixos.org/download. Then, download *colorstreams.tar.gz* from Zenodo, decompress it and open a shell in the artifact's root directory. To build and install *colorstreams*, run *make* and *sudo make install*. To build the benchmark programs, run *make xps_build*.

A.3.2 Basic Test

To test the artifact, run *make test* (~ 15 minutes). Then, run *make test_stats* and *make test_results*. You should obtain results similar to Listings 1 and 2 respectively.

A.4 Evaluation workflow

A.4.1 Major Claims

- (C1): Our Shrink and Split algorithm is more precise than the state-of-the-art (Newsome et al.) and even competitive against model counters.
- (C2): Our Shrink and Split algorithm is faster than the stateof-the-art (Newsome et al.) and more robust that model counters.
- (C3): Our prioritization approach using weighted quantitative control is more precise than other scoring methods.
- (C4): Our method can be used in realistic end-to-end prioritization scenarios.

A.4.2 Experiments

If you are using the docker image, please refer to Section A.3.1 to start a container. Otherwise, open a shell inside the artifact's root directory.

(E1): Mainline experiments (10 human-minutes + 10 compute-hours): Analyses of both benchmarks for reproducing the results from the paper, excluding appendices. Execution: run *make xps_less*

Results: run *make figs_less SAVEFIG=<directory>*. Figures will be saved to the directory specified with SAVEFIG, which can be your shared directory if you are using docker and set up one.

- (C1): fig1.pdf (Figure 1), fig3.pdf (Figure 2)
- (C2): tab4.txt (Listing 3), speedups.txt (Listing 4)
- (C3): tab5_oob.txt (Listing 5), tab5_cfh.txt (Listing 6)(C4): fig5.pdf (Figure 3)
- (E2): Full experiments (10 human-minutes + 20 computehour or 10 compute-hours if (E1) was run first): Analyses of both benchmarks for reproducing all results from the paper, including appendices.
 - Execution: run make xps_full

Results: run *make figs_full SAVEFIG=<directory>*. Figures will be saved to the directory specified with SAVEFIG, which can be your shared directory if you are using docker and set up one.

You can find more details about the experiments in the artifact's readmes.

A.5 Notes on Reusability

The artifact includes tutorials and API documentation for Colorstreams, as well as instructions for using and expanding the benchmarks. Please refer to the readmes (in *colorstreams.tar.gz*).

A.6 Version

Based on the LaTeX template for Artifact Evaluation V20231005. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at https://secartifacts.github.io/usenixsec2025/.

> make test_s Executed inst	tats ructions and analysis	runtime (s) for eac	h bug:		
Bug	Analysis	Symbolic Instr.	Total Instr.	SE Runtime	Total Runtime
can	Symbolic Policy <1>	2	2709	0.0055747	14.6759
can2	Symbolic Policy <1>	3	2830	0.010097	17.4844
ссору	Symbolic Policy <1>	6	27	0.00972962	7.04594
cfi	Symbolic Policy <1>	12	2249	0.0351627	7.89045
cfi2	Symbolic Policy <1>	5	2218	0.0258272	7.94325
copy	Symbolic Policy <1>	2	22	0.00235724	12.9396
div	Symbolic Policy <1>	3	23	0.00308895	16.2794
grub	Symbolic Policy <1>	84	3868	0.200363	107.172
impflow	Symbolic Policy <1>	3	40	0.00736499	6.63708
koobe	Symbolic Policy <1>	12	4075	0.0235684	26.1484
тсору	Symbolic Policy <1>	3	23	0.00285673	6.39718
minesweeper1	Symbolic Policy <1>	278	68911	16.3686	96.9821
minesweeper2	Symbolic Policy <1>	254	107110	13.5707	107.404
mixdup	Symbolic Policy <1>	13	33	0.0125771	63.9717
motex1	Symbolic Policy <1>	18	78	0.0429707	30.1603
motex2	Symbolic Policy <1>	15	1848	0.0337915	15.7375
mul	Symbolic Policy <1>	3	23	0.00335526	35.8923
popent	Symbolic Policy <1>	39	59	0.045434	7.2513
spray	Symbolic Policy <1>	8	2224	0.0165174	24.4337
spray2	Symbolic Policy <1>	16	2248	3.29333	80.0886
spray3	Symbolic Policy <1>	20	89	3.73511	58.8148
sub	Symbolic Policy <1>	3	24	0.00478458	5.81582
sum	Symbolic Policy <1>	4	32	0.00477386	43.1932
uafubi	Symbolic Policy <1>	11	2851	0.738554	178.45
uafubi2	Symbolic Policy <1>	2	2843	0.00483179	15.5785
Total SE runt Total overall	ime: 38.196424007415 runtime: 978.809723	77 3772278			

Listing 1: Result of make test_stats

> make test_results Results compared to grou	und truth (d4 or	ganak, when available):				
Algorithm	True (TP/TN)	Approx (<100%/>100%)	False (FP/FN)	Timeout	Unknown	Total Runtime
Taint	24/2		4/0			0 s
WeakControl	24/6	-	0/0	-	-	0 s
StrongControl	7/23	-	0/0	-	-	1 s
d4	26	-	-	4	-	42 s
ganak	26	-	-	4	-	40 s
approxmc	23	1/0	-	4	2	59 s
Newsome	10	14/2	_	0	4	8m54s
S&S (100 max splits)	23	1/2	-	0	4	47 s
S&SFB (100 max splits)	25	0/1	-	0	4	47 s

Listing 2:	Result	of make	test	results



Figure 1: fig1.pdf

QC precision comparison between S&S and other algorithms (1) S&S (100 max splits) (2) S&SFB (100 max splits)



Figure 2: fig3.pdf

Results compared to groun	nd truth (d4 or	ganak, when available):				
Algorithm	True (TP/TN)	Approx (<100%/>100%)	False (FP/FN)	Timeout	Unknown	Total Runtime
Taint	162/65		24/0			0 s
WeakControl	162/89	-	0/0	_	-	3 s
StrongControl	81/170	-	0/0	-	-	46 s
d4	246	-	-	5	-	35m35s
ganak	245	-	-	6	-	32m49s
approxmc	228	18/0	-	1	4	15m08s
Newsome	105	137/4	-	1	5	2h08m48s
S&S (10 max splits)	235	6/5	-	0	5	10m56s
S&S (50 max splits)	237	4/5	-	0	5	11m12s
S&S (100 max splits)	237	4/5	-	0	5	11m52s
S&S (500 max splits)	237	4/5	-	0	5	15m19s
S&S (1000 max splits)	237	4/5	-	0	5	19m33s
S&SFB (10 max splits)	238	7/1	-	0	5	10m26s
S&SFB (50 max splits)	240	5/1	-	0	5	10m45s
S&SFB (100 max splits)	241	4/1	-	0	5	10m55s
S&SFB (500 max splits)	241	4/1	-	0	5	13m05s
S&SFB (1000 max splits)	241	4/1	-	0	5	15m27s

Listing 3: tab4.txt

Runtime averages (5% trimmed):
Newsome: 30.8 (22.2)
S&S (10 max splits): 2.62 (0.505)
S&S (50 max splits): 2.68 (0.611)
S&S (100 max splits): 2.84 (0.64)
S&S (500 max splits): 3.67 (0.637)
S&S (1000 max splits): 4.67 (0.622)
S&SFB (10 max splits): 2.5 (0.491)
S&SFB (50 max splits): 2.57 (0.564)
S&SFB (100 max splits): 2.61 (0.558)
S&SFB (500 max splits): 3.13 (0.574)
S&SFB (1000 max splits): 3.7 (0.569)
Speedup averages (5% trimmed):
S&S (10 max splits): 79.4 (71.5)
S&S (50 max splits): 80 (71.2)
S&S (100 max splits): 80.9 (72.9)
S&S (500 max splits): 83.1 (73.2)
S&S (1000 max splits): 82.8 (73.4)
S&SFB (10 max splits): 87.2 (77)
S&SFB (50 max splits): 84.7 (75.4)
S&SFB (100 max splits): 88.9 (77.5)
S&SFB (500 max splits): 84.9 (74.9)
S&SFB (1000 max splits): 84.8 (75.3)

Listing 4: speedups.txt

Basic scores base	d on counts and dom	ains :		
Bug	Sink	Size	QC	wQC
motex2	of_wsize <1>	64	0.0832	0.0837
motex1	of_wsize <1>	64	0.0832	0
heartbleed	payload_size <11>	32	0.5	0.5
cve-2023-37837	read_off <0>	16	1	1
cve-2022-30790_2	of_woff2 <5>	64	0.249	0.037
cve-2022-30790	of_woff <7>	64	0.0248	0.0066
cve-2022-30790	of_wsize <5>	64	0.219	8.59e-08
cve-2022-30552	of_woff <3>	64	0.0248	0.0066
cve-2022-30552	of_wsize <1>	64	0.219	8.59e-08
cve-2021-3246	of_size <1>	64	0.275	0.221
cve-2019-14202	of_wsize <1>	32	0.499	0.499
cve-2019-14192	of_wsize <1>	32	0.499	0.499

Listing 5: tab5_oob.txt

Note: to obtain the scores from the paper, multiply the scores in the table by the bit size of the sink. For vulnerabilities with two parameters, add both scores together.

CFH capability scor	es:			
Bug	ID	Туре	QC Score	wQC Score
cve -2020-14393.cfh	0	Jump	0.996	1
cve-2023-43338.cfh	0	Call	1.53e-05	1
cve-2021-26567.cfh	0	Ret	0.969	0
cve-2024-41881.cfh	0	Ret	0.939	0
cve-2024-43700.cfh	0	Ret	0.776	0

Listing 6: tab5_cfh.txt



Figure 3: fig5.pdf