LTTng-UST dynamic tracepoints in uftrace Progress Report Meeting

Clément Guidi Mohammad Nassiri

DORSAL – Polytechnique Montréal

January 14, 2022

C. Guidi, M. Nassiri (Polytechnique Montréal)

LTTng-UST dynamic tracepoints in uftrace

January 14, 2022 1 / 17

Table of contents

- Introduction
- Previous achievements
 - Main contributions
 - Side improvements
- ITTng-UST tracepoints
 - Tracepoint definition
 - Using uftrace features
 - Visualization
 - Difficulties encountered
 - Demo
- Work in progress and future research
 - Work in progress
 - Future work
- 5 Conclusion

Introduction

About uftrace:

- \bullet function tracing tool for C/C++/Rust applications
- can instrument userspace
- empowering features, including
 - plain and regex filters for function/library names
 - execution duration and call depth filters
 - argument and return value logging

Upstream limitations:

- binary instrumentation performed before execution
- custom trace format

Previous achievements

Main contributions

Previous work at DORSAL includes:

- indirect jump resolution: improve patching success rate by identifying **indirect jump locations** (external library) [Gabriel Pollo-Guilbert]
- x86 runtime patching and unpatching: instrument binaries during execution using a **locking mechanism** and **out of line execution**; remove tracepoints [Christian Harper-Cyr, Anas Balboul, Ahmad Shahnejat and Gabriel Pollo-Guilbert]
- client command: send commands to a **libmcount daemon** running inside a uftrace target [Clément Guidi]

Previous achievements

Side improvements

Smaller improvements have been made:

- read external symbol file for stripped binaries using --with-syms=DIR option
- detailed patching statistics
- unpatch option enhancement, for the new unpatching capabilities
- bug fixes
 - Intel CET ENDBRANCH instruction was sometimes omitted
 - cache serialization: membarrier MEMBARRIER_CMD_PRIVATE_EXPEDITED_SYNC_CORE command unavailable on older kernels; use CPUID interrupt instead

Previous achievements

Side improvements

Example of detailed statistics in debug mode when instrumenting python3.11.

dynamic: dynamic patch stats for 'python3.11' dvnamic: total: 1479 dynamic: patched: 602 (40.70%) dynamic: failed: 853 (57.67%) dynamic: no detail: 0 (0.00%)64 (72.72%) dynamic: relative jump: 0 (0.00%)dynamic: relative call: 24(27.27%)dynamic: PIC: dynamic: skipped: 24(1.62%)dynamic: no match: 0

uftrace makes use of:

- custom trace format serialized timestamped events
- custom event buffering mechanism

Objective Leverage uftrace capabilities to produce LTTng CTF traces

- filters
- dynamic patching
- argument and return value logging

Solution Emitting LTTng events in regular libmcount probe entries.

Tracepoint definition

#define TRACEPOINT_PROVIDER lttng_ust_cyg_profile

```
#include <lttng/tracepoint.h>
```

```
TRACEPOINT_EVENT_CLASS(
    lttng_ust_cyg_profile,
    func_class,
    TP_ARGS(
       void *, func_addr,
       void *, call_site,
       char *, arg_ret_str),
    TP_FIELDS(
       ctf_integer_hex(unsigned long, addr, (unsigned long) func_addr)
       ctf_integer_hex(unsigned long, call_site, (unsigned long) call_site)
       ctf_string(arg_ret_str, arg_ret_str)
)
```

Tracepoint definition

```
TRACEPOINT EVENT INSTANCE (
  lttng_ust_cyg_profile,
  func_class,
  func entry,
  TP_ARGS(void *, func_addr,
            void *, call site,
            char *, arg ret str)
TRACEPOINT_EVENT_INSTANCE(
  lttng_ust_cyg_profile,
  func_class,
  func_exit,
  TP_ARGS(void *, func_addr,
            void *, call_site,
            char *, arg_ret_str)
```

LTTng-UST tracepoints Using uftrace features

- Function/library filter: works normally, handled by libmcount at function entry
- Call depth filter: works normally, handled by libmcount at function entry
- Time filter: not functional, handled at function exit. Entry event still emitted
- Argument and return value: functional, available in arg_ret_str event field

Visualization

*							Trace Co	mpass					^	×
File	To	ols Window Help												
0	E P	python-20220112-17402	25/ust/uid/100	00/64-bit	×									
6		Timestamp	Channel	CPU	Event type	Contents								
	A	<srch></srch>	<srch></srch>	<srch></srch>	<srch></srch>	rch> <srch></srch>								
92		17:40:33.383 738 401	channel0_1	1	lttng_ust_cyg_profile:func_entry	addr=0x556c9066b2e0, ca	ll_site=0x7f259c7ed31e, a	rg_ret_str=0,100, conte:	ktvpid=3320, conte	xtvtid=3320, contex	tprocname=python			
		17:40:33.383 739 665	channel0_1	1	lttng_ust_cyg_profile:func_exit	addr=0x556c9066b2e0, ca	ll_site=0x7f259c7ed31e, a	rg_ret_str=0x556c938aa	630, contextvpid=	3320, contextvtid=3	320, contextprocname=	python		
		17:40:33.383 740 445	channel0_1	1	lttng_ust_cyg_profile:func_exit	addr=0x556c906caf65, ca	l_site=0x7f259c7ed31e, ar	g_ret_str=0x556c938aa	630, contextvpid=	3320, contextvtid=3	320, contextprocname=p	ython		
		17:40:33.383 740 856	channel0_1	1	lttng_ust_cyg_profile:func_exit	addr=0x556c906ce325, ca	ll_site=0x556c908a36d4, a	rg_ret_str=, contextvp	id=3320, contextv	tid=3320, contextpr	cname=python			
		17:40:33.383 743 069	channel0_1	1	lttng_ust_cyg_profile:func_entry	addr=0x556c9066adb0, ca	ll_site=0x556c908a36e9, a	arg_ret_str=, contextvp	oid=3320, contextv	tid=3320, contextpr	ocname=python			
		17:40:33.383 743 600	channel0_1	1	lttng_ust_cyg_profile:func_exit	addr=0x556c9066adb0, ca	ll_site=0x556c908a36e9, a	irg_ret_str=, contextvp	oid=3320, contextv	tid=3320, contextpr	ocname=python			
		17:40:33.383 746 196	channel0_1	1	lttng_ust_cyg_profile:func_entry	addr=0x556c9066b1e0, ca	ll_site=0x556c908a3708, a	irg_ret_str=, contextvp	oid=3320, contextv	tid=3320, contextpr	ocname=python			
	_	17:40:33.383 754 360	channel0_1	1	lttng_ust_cyg_profile:func_exit	addr=0x556c9066b1e0, ca	ll_site=0x556c908a3708, a	erg_ret_str=, contextvp	oid=3320, contextv	tid=3320, contextpr	ocname≕python			_
		17:40:33.383 759 095	channel0_1	1	lttng_ust_cyg_profile:func_entry	addr=0x556c9066a5a0, ca	ll_site=0x556c908a3716, a	irg_ret_str=0x556c938a	a630,100,[<u>**</u>](36m&_	O_2_1_stdin_[[[0m, <	ontextvpid=3320, conte	xtvtid=3320, context	_procname=python	
		17:40:33.663 845 044	channel0_1	1	lttng_ust_cyg_profile:func_exit	addr=0x556c9066a5a0, ca	ll_site=0x556c908a3716, a	rg_ret_str=[;](35m*\n*[[0m, contextvpid	=3320, contextvtid=	3320, contextprocname	=python		
		17:40:33.663 869 549	channel0_1	1	lttng_ust_cyg_profile:func_entry	addr=0x556c9066a750, ca	ll_site=0x556c908a3788, a	irg_ret_str=[](35m*\n*	0m, contextvpid	=3320, contextvtid+	3320, contextprocname	=python		
		17:40:33.663 880 478	channel0_1	1	Ittng ust cyg profile:func exit	addr=0x556c9066a750, ca	II site=0x556c908a3788, a	ing ret_str=1, context.	pid=3320, context.	vtid=3320, context, p	rocname=python			
		17:40:33.663 894 252 channel0_1 1			ittng ust cyg profile:func entry	addr=0x556c906ce325, ca	Il_site=0x556c908a37d8, a	rg_ret_str=0x556c938a	a630,2, context_vpi	i=3320, context_vtid	=3320, contextprocname	=python		
	17:40:33.663 907 295 channel0_1 1 Ittng_ust_cyg				ittng_ust_cyg_profile:func_entry	addr=0x556c906caf65, ca		g_ret_str=0,0x556c938a	a630,2, context_vp	d=3320, context_vtic	=3320, context_procnam	e=python		
		17:40:33.663 914 295	channel0_1	1	lttng_ust_cyg_profile:func_entry	addr=0x556c9066b2e0, ca	ll_site=0x7f259c7ed31e, a	rg_ret_str=0x556c938aa	630,2, context_vpic	=3320, context_vtide	3320, contextprocname	=python		
		17:40:33.663 918 608	channel0_1	1	lttng_ust_cyg_profile:func_exit	addr=0x556c9066b2e0, ca	ll_site=0x7f259c7ed31e, a	rg_ret_str=0x556c938aa	630, contextvpid=	3320, contextvtid=2	320, contextprocname=	python		
17:40:33.663 920 873 channel0_1 1 lttng_ust_cyg_profile:func_exit addr=0x556c906caf65, call_site=0x7f259c7ed31e, arg_ret_str=0x556c938aa630, context_vpid=3320, context_vtid=3320,										320, contextprocname=p	ython			
	411	Histogram 🔲 Propertie	s = Flame	Chart x	Elame Graph (incubator)						2 1 h to a 1	BRB- 0	A & &	
	1	7:40:32.400	17:40:32	.600	17:40:32.800	17:40:33.000	17:40:33.200	17:40:33.400	17:	10:33.600	17:40:33.800	17:40:34.000	17:40:34.200	0
	man A Grandan													_
					pymain_init			ry often and	pyn	ain_run_python			Py_FinalizeEx	_
					Py_InitializeFromConfig				P	Run_AnyFileExFlags			Py_FinalizeEx	
					pyinit_main				_PyRu	n_InteractiveLoopObject		PyGC_Collect	finalize_modules	P
	H-	P Shine Collision			init interp main	7 Delaward	Immediate	PyRun In	teractiveOneObjectEx	PyRun_Interactiv	eO PyRun_InteractiveOne	o gc colect main	YGC_CollectN PyGC	C C n
		PyFunction Vectors	al Prim	config	get codec name Pytmpor	Pyimport	ort Import	PyPegen ru	n parser from file poi	ter PyPegen run p	rs PyPagen run parser			onec
		PyEval Wector	- iv	Pyc	odec_Lookup PyObject	PyObjec	CalFunction	- 1 /2	egen_run_parser	PyPegen_run_p	arser PyPegen run parse		ACT DISCOURSE OF THE P	
		PyEval_EvalFrameDef	aut P	PyCodec	Registry Init PyO PyObje	_PyObject	CallFunctionVa		yPegen_parse	PyPegen_par	sePyPegen_parse			
		PyImport_ImportModule	н <mark>Р</mark> Р	vimport_I	ImportModule Py PyObje	_PyOb)e	t_MakeTpCal	Py	Pegen fill token	PyPegen_fil_to	ken PyPegen fill token			
		object cannetribulu	10 (1	Pythipt	Califunction Py builtin	builtin	imont		tok net	tok off	tok net	-	0.00	
	10	PyFunction_Vectorca	al Py. I	yObject_	CaliFunctionVa Pyo Pyimpor	Pyimport_Impc	rtModuleLevelObject		tok_nextc	tok_nextc	tok_nextc		in thi	-ii
	m	PyObject_MakeTpCall Py PyObject_MakeTpCall Py			_PyObject_Ca	IlMethodIdObjArgs	P C C C C C C C C C C C C C C C C C C C	PyOS_Readline PyOS_Readline PyOS_Readline				100 ED	-11-	
	10	PyEval EvalFrameDefault_ob cfunction_callcfu] object_v]			obje	sct_vacall	Py0	PyOS_StdioReadline PyOS_StdioReadline				<u>1</u> 11		
		PyUbject vectorca	P	buittin	import Buil Pytunc	Pyrunci Pyr	ion vectorcal		inis	init	_ink			-
	<u>0</u>	builtin exec	P Pys	ibject Ca	IMethodidObj Py	PyEval E	ralFrameDefault							
_	m+	III DuEval EvalCode	15-1	ritriar	t warat fast in bettelar	P Dodaka	et Mactoreall					i i		
i, M. Nassiri (Polytechnique Montréal)						LTTn	g-UST dynam	ic tracepoin	LTTng-UST dynamic tracepoints in uftrace					

11 / 17

Difficulties encountered

- Preserving registers: mount saves the registers it alters before executing the probe. LTTng tracepoints are using previously untouched vector registers ymm0 and ymm1. mount needs to save/restore them.
- Time filter: the time filter mechanism is incompatible with LTTng, as function entry events would need to be sent conditionally at the end of the function. This alters the timestamp.
 - Hints: only log exit event with the duration, keeping a stack of conditional events
- Upstream refactoring: conflicting refactoring was applied upstream.

C. Guidi, M. Nassiri (Polytechnique Montréal)

LTTng-UST dynamic tracepoints in uftrace

January 14, 2022 12 / 17

LTTng-UST tracepoints Demo

How to use uftrace with LTTng:

ITTng session: add vpid, vtid and procname userspace context

```
lttng create my-session
lttng enable-event -u -a # all userspace events
lttng add-context -u -t vpid -t vtid -t procname
lttng start
```

- Intrace: instruct uftrace to use libmcount-lttng.so library using --libmcount-lttng option
- instrumentation: instrumenting the target is not mandatory, it can be done at runtime.
 Use --dynamic to initialize the relevant mechanism
- runtime: send patching/unpatching instruction with the client, using regular --patch/-P and --unpatch/-U options

≣ ୬૧୯

LTTng-UST tracepoints Demo



LTTng-UST dynamic tracepoints in uftrace

January 14, 2022 14 / 17

Work in progress and future research Work in progress

Evaluating the performance of dynamic binary instrumentation according to the following criteria:

- patching success rate: the percentage of locations that are successfully instrumented
- patching perturbation: the time needed to instrument functions or remove instrumentation, and global slowdown caused to the target
- probe overhead: slowdown caused by the execution of probes
- memory consumption

We will evaluate performance on a list of around 30 applications with the following characteristics:

- C, C++ or Rust language
- low or high function count
- small or big binary size
- single- or multi-threaded

Work in progress and future research

Next project steps:

- improve instrumentation methods to increase patching success rate and efficiency
 - use 2-byte relative jumps with intermediate trampolines
 - use instruction punning
- support ARM platforms
- validate the robustness
- attach on the fly to running process
 - a PR exists on GitHub but is on hold
- apply methods to other tools: Kprobes, GDB
- support adaptative tracing: continuously patch and unpatch function based on usage

Source code repository: https://gitlab.com/dorsal1/uftrace

Questions?

January 14, 2022 17 / 17