44th International Conference on Software Engineering (ICSE 2022)

Linear-time Temporal Logic guided Greybox Fuzzing

Ruijie Meng

ruijie@comp.nus.edu.sg

Co-authors: Zhen Dong, Jialin Li, Ivan Beschastnikh, Abhik Roychoudhury

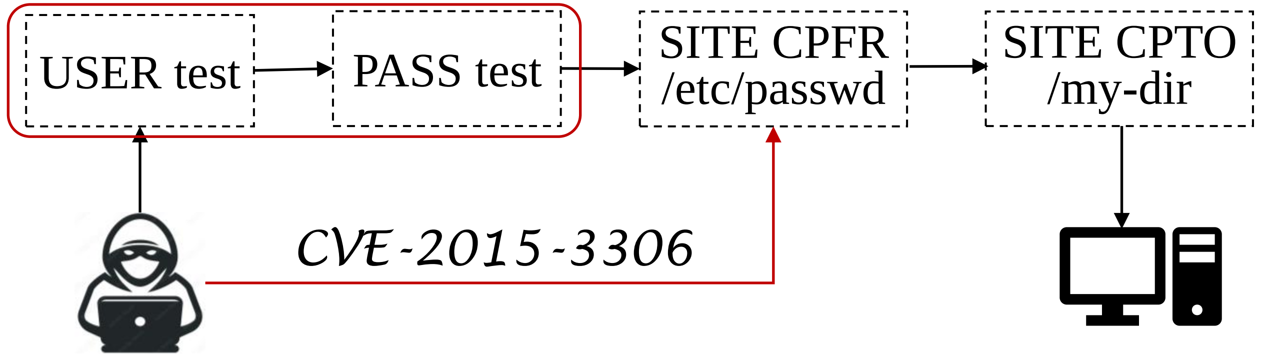
Background

Violations of Simple Oracles:

Crashes/Hangs

Overflows…

Violations of Temporal Logic Properties:



USER test

PASS test

SITE CPFR /etc/passwd

SITE CPTO /my-dir

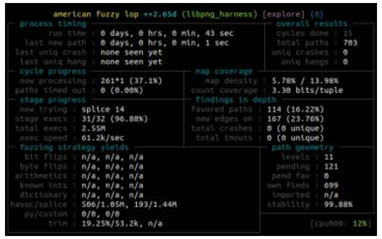
CVE-2015-3306

Fuzzing

- ☛ Automatic and dynamic testing technique
- ☛ Continuously generates inputs and feeds them to the target programs, and then reports inputs that trigger ***crashes*** or ***hangs***
- ☛ Types:
  - ➢ **Blackbox Fuzzing** (without program analysis and feedback)
  - ➢ **Whitebox Fuzzing** (heavy program analysis)
  - ➢ **Greybox Fuzzing** (lightweight feedback)

PEACH FUZZER

KLEE



```
         american fuzzy lop ++2.65d (libpng_harness) [explore] {0}
┌─ process timing ────────────────────────────────────┬─ overall results ────┐
│        run time : 0 days, 0 hrs, 0 min, 43 sec      │  cycles done : 15    │
│   last new path : 0 days, 0 hrs, 0 min, 1 sec       │  total paths : 703   │
│ last uniq crash : none seen yet                     │ uniq crashes : 0     │
│  last uniq hang : none seen yet                     │   uniq hangs : 0     │
├─ cycle progress ───────────────────┬─ map coverage ─┴──────────────────────┤
│  now processing : 261*1 (37.1%)    │    map density : 5.78% / 13.98%       │
│ paths timed out : 0 (0.00%)        │ count coverage : 3.30 bits/tuple      │
├─ stage progress ───────────────────┼─ findings in depth ───────────────────┤
│  now trying : splice 14            │ favored paths : 114 (16.22%)          │
│ stage execs : 31/32 (96.88%)       │  new edges on : 167 (23.76%)          │
│ total execs : 2.55M                │ total crashes : 0 (0 unique)          │
│  exec speed : 61.2k/sec            │  total tmouts : 0 (0 unique)          │
├─ fuzzing strategy yields ──────────┴───────────────┬─ path geometry ────────┤
│   bit flips : n/a, n/a, n/a                        │    levels : 11         │
│  byte flips : n/a, n/a, n/a                        │   pending : 121        │
│ arithmetics : n/a, n/a, n/a                        │  pend fav : 0          │
│  known ints : n/a, n/a, n/a                        │ own finds : 699        │
│  dictionary : n/a, n/a, n/a                        │  imported : n/a        │
│havoc/splice : 506/1.05M, 193/1.44M                 │ stability : 99.88%     │
│   py/custom : 0/0, 0/0                             ├────────────────────────┘
│        trim : 19.25%/53.2k, n/a                    │          [cpu000: 12%]
└────────────────────────────────────────────────────┘
```

Greybox Fuzzing

Advantages of Greybox Fuzzing

- ✓ better **coverage** than blackbox fuzzing
- ✓ better **scalability** than whitebox fuzzing
- ✓ widely used and have exposed many bugs

Challenges of Greybox Fuzzing

- ✗ Checking functional properties (e.g., linear-time temporal logic (LTL) properties), not just crashes or hangs
- ✗ Efficiently search executions of systems under test to check

There is already an approach that does that — model checking!!

But… model checking works well on models, and scales poorly to large programs

Can we have the best of the both worlds ???

Linear-time Temporal Logic (LTL)

☛ **LTL Syntax:**

- ➢ Propositional Linear-time Temporal logic
- ➢ $\varphi = X\varphi \mid G\varphi \mid F\varphi \mid \varphi_1 \, U \, \varphi_2 \mid \varphi_1 \, R \, \varphi_2 \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid Prop$
- ➢ Temporal operators: X(next state), F(eventually), G(globally), U(until), R(release)

☛ **LTL Conventions:**

- ➢ An LTL formula φ is interpreted over an infinite sequence of states $\pi = s_0, s_1, \ldots$

  Use $\mathcal{M},\pi \models \varphi$ to denote that formula φ holds in path π of system model $\mathcal{M}$

- ➢ An LTL property φ is true of a system model *iff* all its traces satisfy φ, $\mathcal{M} \models \varphi$ *iff* $\mathcal{M},\pi \models \varphi$ for all traces π in system model $\mathcal{M}$

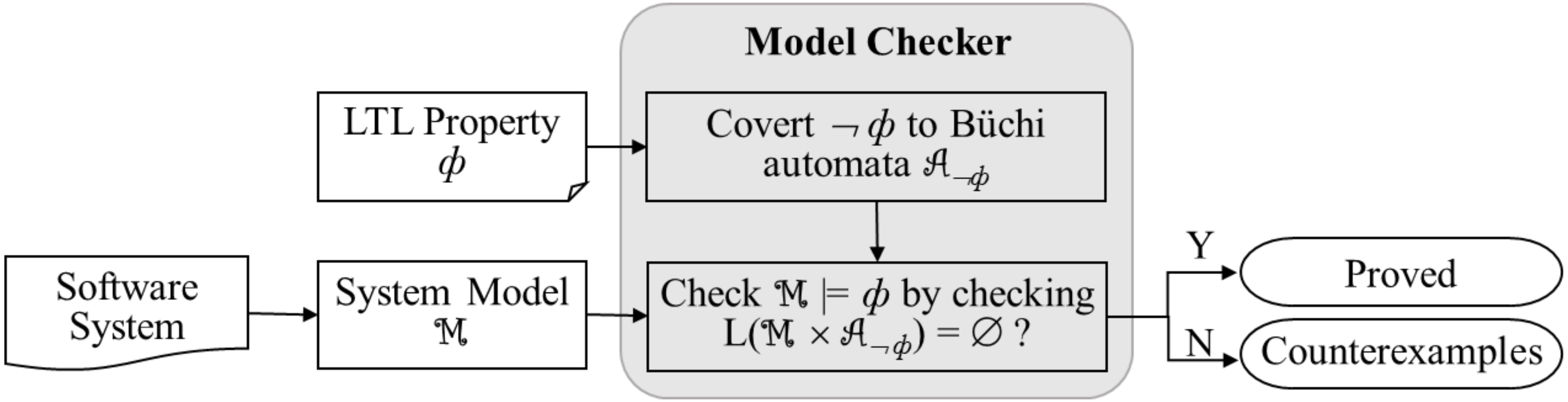
Software Model Checking

☛ A property verification technique, but common usage is bug-finding

☛ Check if a *finite-state* transition system model satisfies a temporal logic property

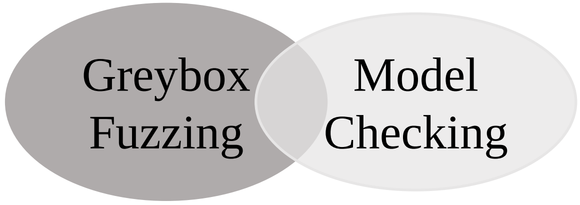
- ➢ The property constraints orderings of events
- ➢ The system model is abstracted from the software system

☛ Automata-theoretic model checking is widely used (e.g., SPIN)

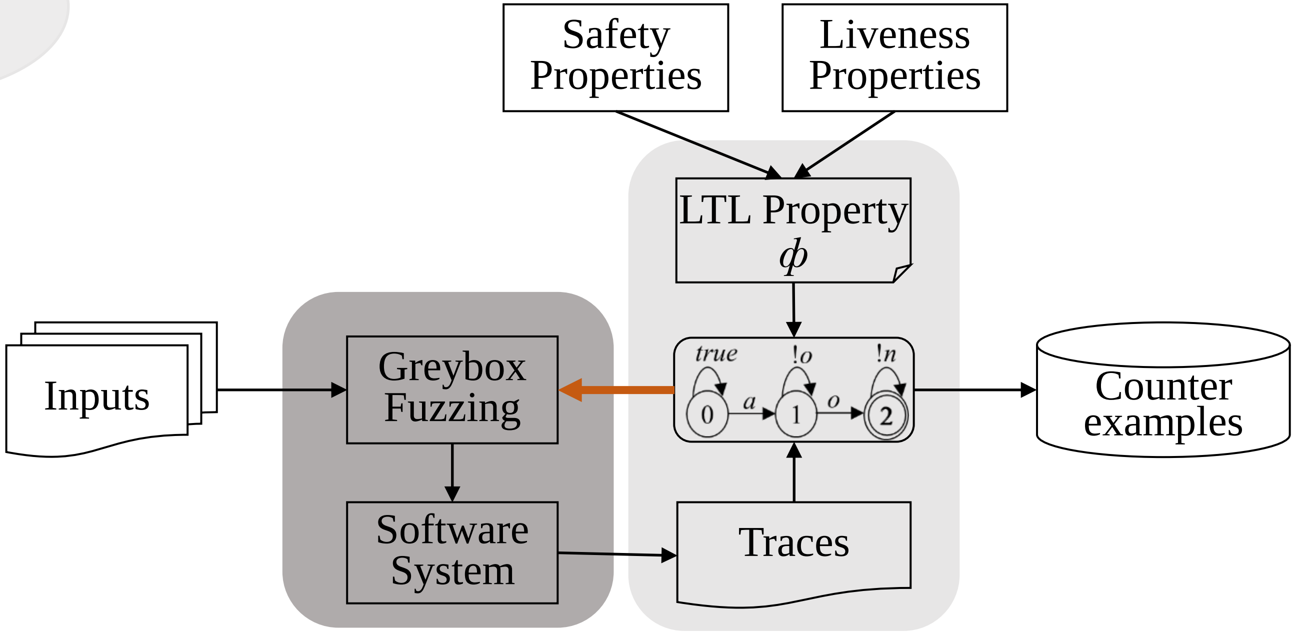

Model Checker
LTL Property ϕ
Covert $\neg\phi$ to Büchi automata $\mathcal{A}_{\neg\phi}$
Software System
System Model $\mathcal{M}$
Check $\mathcal{M} \models \phi$ by checking $L(\mathcal{M} \times \mathcal{A}_{\neg\phi}) = \varnothing$?
Y
Proved
N
Counterexamples

LTL guided Fuzzing

- Use LTL properties as test oracles and check them
- Use Büchi automata of the negated LTL properties to guide greybox fuzzing

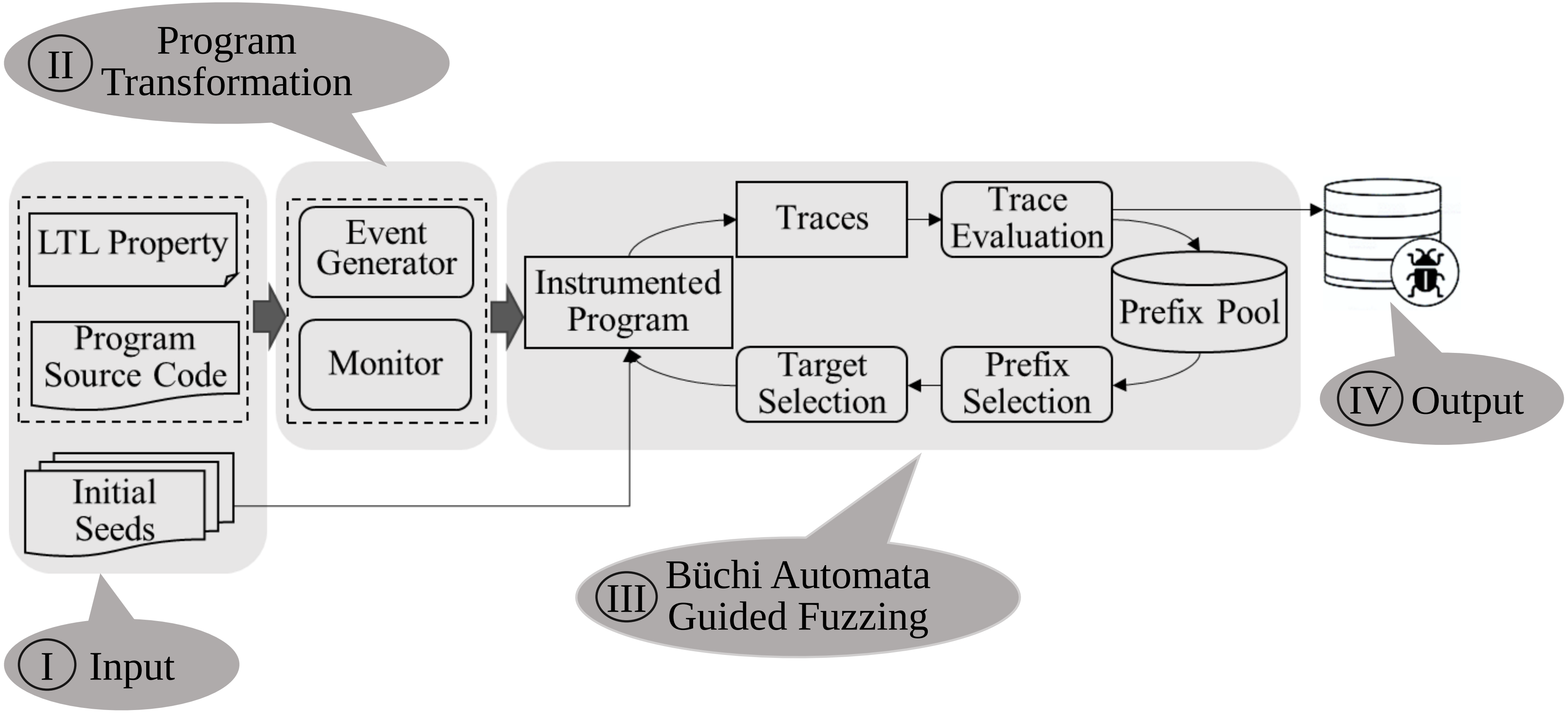


Synergy



Workflow

☛ **Work on sequential reactive stateful systems**



LTL Property Construction

Step 1

Extract one informal property from the FTP RFC

Illustrative Example:

When the *user_quota* is exceeded and the *quota mechanism* is activated, the server should finally reply *552* to stop receiving data

Step 2

Translate into LTL formula

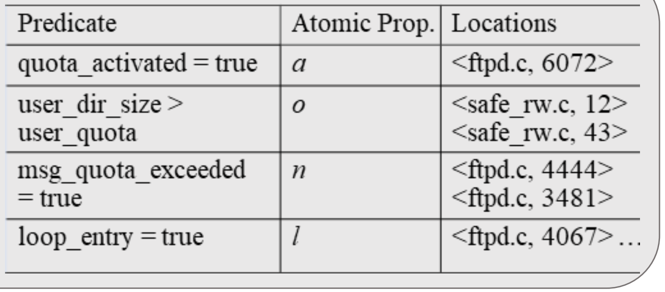
Illustrative Example:

Property ϕ: $\neg F\ (a \wedge F\ (o \wedge G\neg n)) \rightarrow F\ (a \wedge F\ (o \wedge G\neg n))$

Step 3

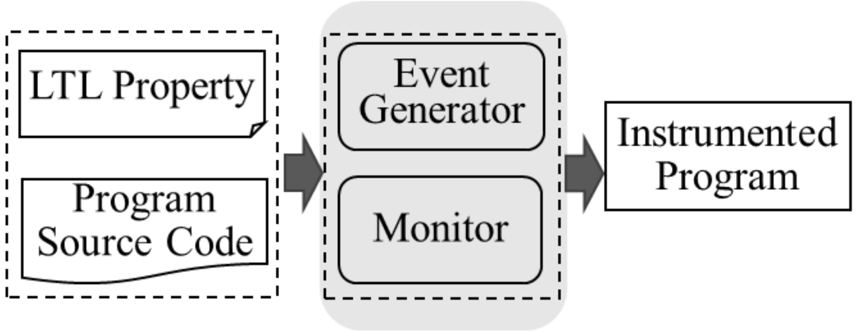
Identify program locations

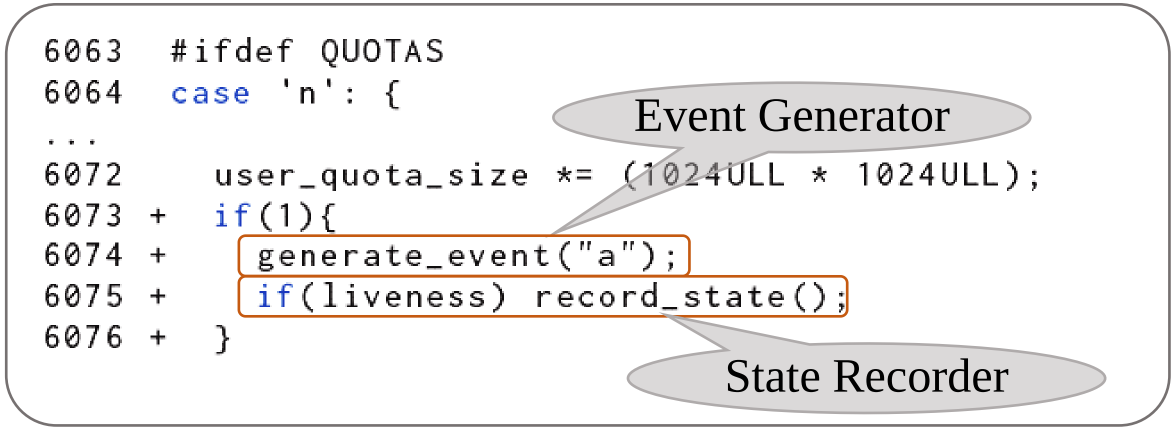
Illustrative Example:



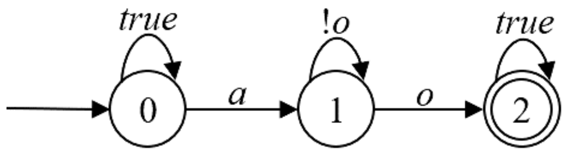
Predicate	Atomic Prop.	Locations
quota_activated = true	*a*	<ftpd.c, 6072>
user_dir_size > user_quota	*o*	<safe_rw.c, 12> <safe_rw.c, 43>
msg_quota_exceeded = true	*n*	<ftpd.c, 4444> <ftpd.c, 3481>
loop_entry = true	*l*	<ftpd.c, 4067>…

Program Transformation

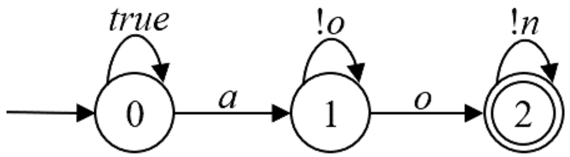




```
#ifdef QUOTAS
case 'n': {
...
     user_quota_size *= (1024ULL * 1024ULL);
+    if(1){
+      generate_event("a");
+      if(liveness) record_state();
+    }
```



[Safety properties]



[Liveness properties]

Actions:

- Instrument Monitor
- Evaluate traces

$$\pi_e = e_0 e_1 \cdots e_i e_{i+1} \cdots e_{i+h} e_i \cdots e_{i+h}$$

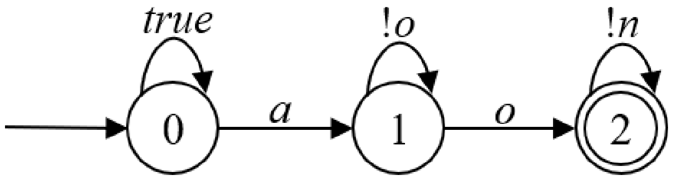
$$\pi_s = s_0 s_1 \cdots \overbrace{s_i s_{i+1} \cdots s_{i+h}}^{loop\ body} s_{i+h+1} \cdots s_{i+2h}$$

Büchi Automata Guided Fuzzing

- ☛ Büchi automata accepts traces with a specific order of propositions
- ☛ Direct fuzzing towards multiple program locations in a specific order
  - ➢ **Power scheduling (reach one target):**

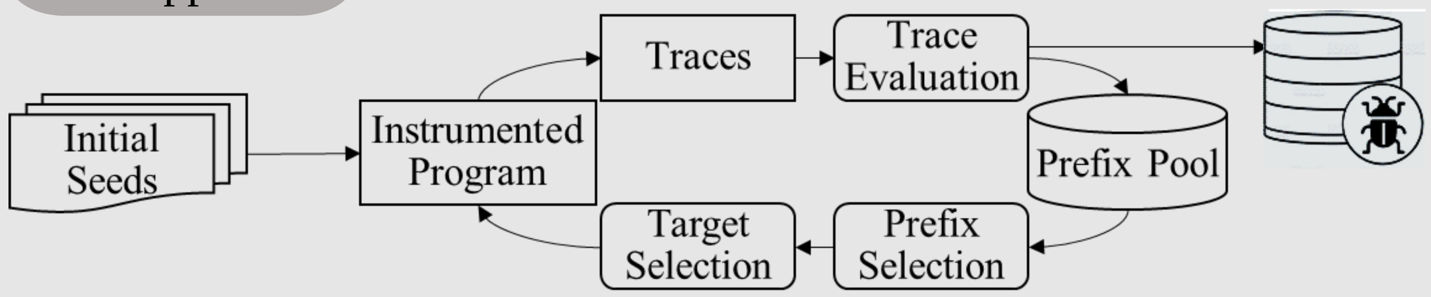
    Select seeds closer to the target on the inter-procedural control flow graph
  - ➢ **Input prefix saving (reach further targets):**

    Observe execution and save the achieved progress when reaching a target by saving input prefixes



Büchi Automata Guided Fuzzing

Our Approach

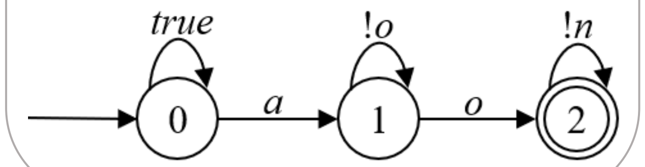


Example

1. LTL property ϕ:

$\neg F (a \wedge F (o \wedge G \neg n))$

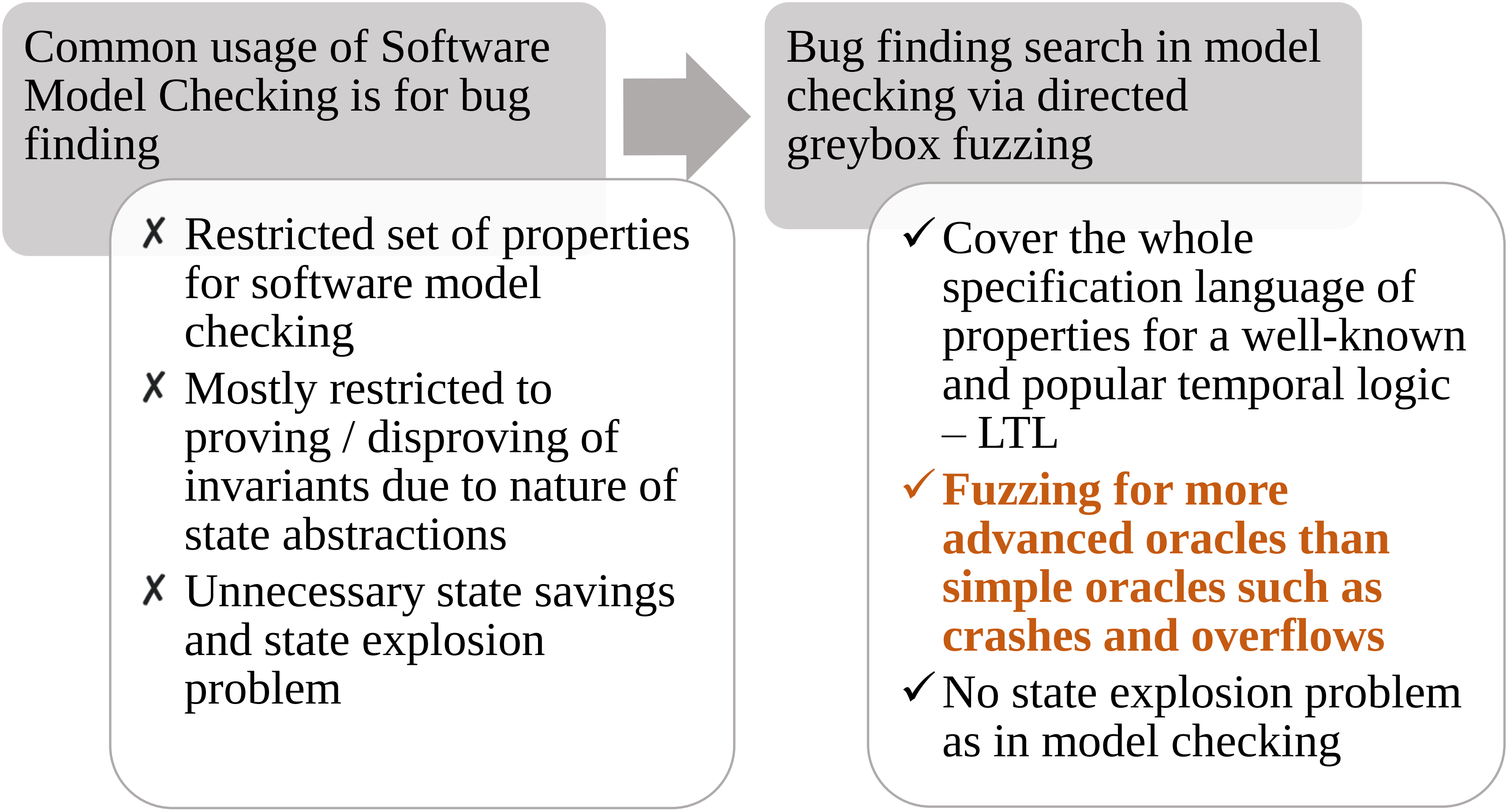
2. Büchi automata $\mathcal{A}_{\neg\phi}$:



Fuzzing Process

Prefix	State	Target	Input	Trace	Prefix Saving	Violation
--	0	*a*	xxxy	{*a*}	<1, xxx>	×
xxx	1	*o*	xxxzy	{*a*, *o*}	<2, xxxz>	×
xxxz	2	*l*	xxxzww	{*a*, *o*, *l*}	<2, xxxzw>	×
xxxzw	2	*l*	xxxzwzz	{*a*, *o*, *l*, *l*}	--	√

Finding deep bugs from Software MC via Fuzzing



Common usage of Software Model Checking is for bug finding

- ✗ Restricted set of properties for software model checking
- ✗ Mostly restricted to proving / disproving of invariants due to nature of state abstractions
- ✗ Unnecessary state savings and state explosion problem

Bug finding search in model checking via directed greybox fuzzing

- ✓ Cover the whole specification language of properties for a well-known and popular temporal logic – LTL
- ✓ **Fuzzing for more advanced oracles than simple oracles such as crashes and overflows**
- ✓ No state explosion problem as in model checking

Evaluation

Research Questions

RQ1 Effectiveness: How effective is LTL-Fuzzer at finding LTL property violations?

RQ2 Comparison: How does LTL-Fuzzer compare to the state-of-the-art tools in terms of finding LTL property violations?

RQ3 Usefulness: How useful is LTL-Fuzzer in revealing LTL property violations in real-world systems?

Subject Programs

- ProFTPD
- Live555
- OpenSSH
- Contiki-Telnet
- Pure-FTPd
- OpenSSL
- TinyDTLS

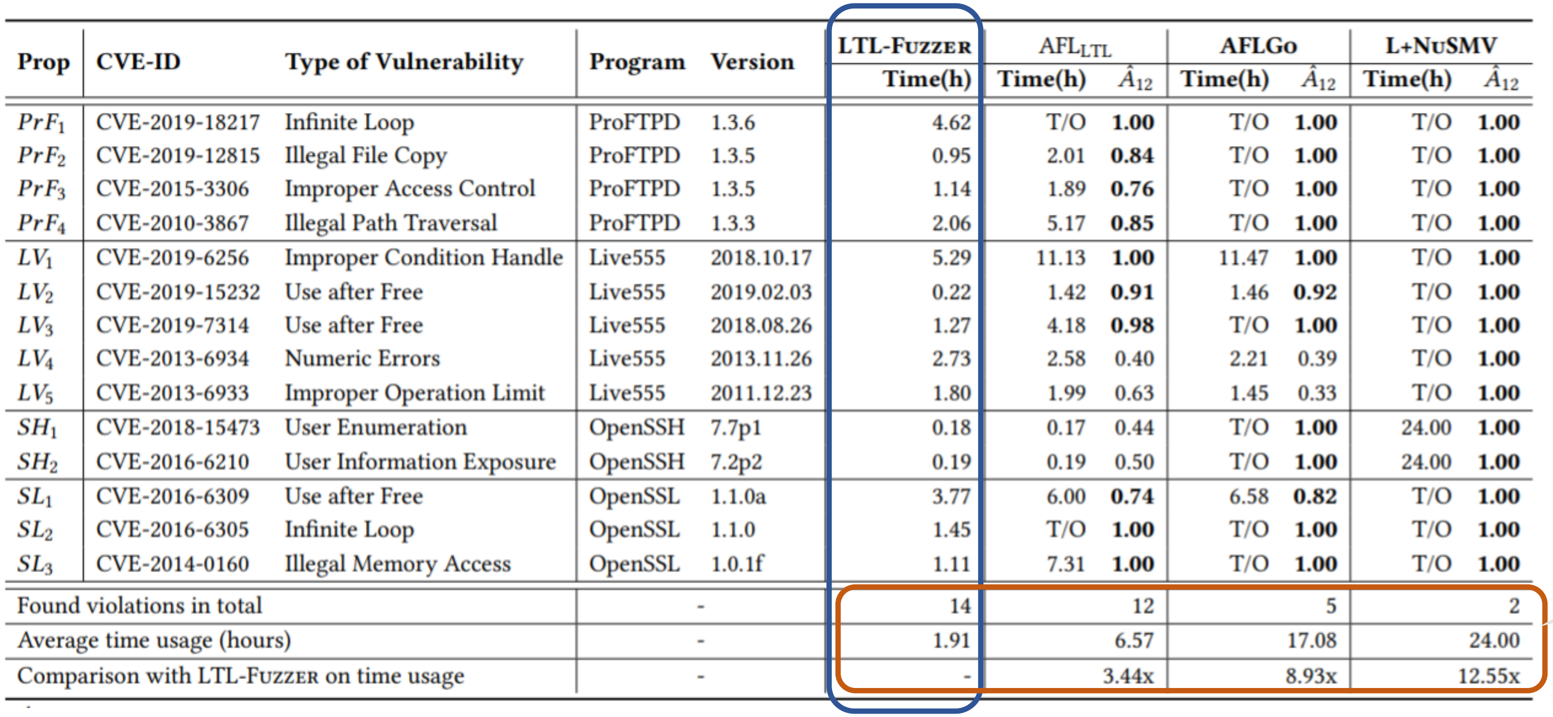
Comparisons

- AFLGo
- AFL_{LTL}
- L+NuSMV

Our tool LTL-Fuzzer and dataset are publicly available at:

https://github.com/ltlfuzzer/LTL-Fuzzer

Effectiveness & Comparison



Prop	CVE-ID	Type of Vulnerability	Program	Version	LTL-Fuzzer	AFL$_{LTL}$		AFLGo		L+NuSMV	
					Time(h)	Time(h)	$\hat{A}_{12}$	Time(h)	$\hat{A}_{12}$	Time(h)	$\hat{A}_{12}$
PrF_1	CVE-2019-18217	Infinite Loop	ProFTPD	1.3.6	4.62	T/O	**1.00**	T/O	**1.00**	T/O	**1.00**
PrF_2	CVE-2019-12815	Illegal File Copy	ProFTPD	1.3.5	0.95	2.01	**0.84**	T/O	**1.00**	T/O	**1.00**
PrF_3	CVE-2015-3306	Improper Access Control	ProFTPD	1.3.5	1.14	1.89	**0.76**	T/O	**1.00**	T/O	**1.00**
PrF_4	CVE-2010-3867	Illegal Path Traversal	ProFTPD	1.3.3	2.06	5.17	**0.85**	T/O	**1.00**	T/O	**1.00**
LV_1	CVE-2019-6256	Improper Condition Handle	Live555	2018.10.17	5.29	11.13	**1.00**	11.47	**1.00**	T/O	**1.00**
LV_2	CVE-2019-15232	Use after Free	Live555	2019.02.03	0.22	1.42	**0.91**	1.46	**0.92**	T/O	**1.00**
LV_3	CVE-2019-7314	Use after Free	Live555	2018.08.26	1.27	4.18	**0.98**	T/O	**1.00**	T/O	**1.00**
LV_4	CVE-2013-6934	Numeric Errors	Live555	2013.11.26	2.73	2.58	0.40	2.21	0.39	T/O	**1.00**
LV_5	CVE-2013-6933	Improper Operation Limit	Live555	2011.12.23	1.80	1.99	0.63	1.45	0.33	T/O	**1.00**
SH_1	CVE-2018-15473	User Enumeration	OpenSSH	7.7p1	0.18	0.17	0.44	T/O	**1.00**	24.00	**1.00**
SH_2	CVE-2016-6210	User Information Exposure	OpenSSH	7.2p2	0.19	0.19	0.50	T/O	**1.00**	24.00	**1.00**
SL_1	CVE-2016-6309	Use after Free	OpenSSL	1.1.0a	3.77	6.00	**0.74**	6.58	**0.82**	T/O	**1.00**
SL_2	CVE-2016-6305	Infinite Loop	OpenSSL	1.1.0	1.45	T/O	**1.00**	T/O	**1.00**	T/O	**1.00**
SL_3	CVE-2014-0160	Illegal Memory Access	OpenSSL	1.0.1f	1.11	7.31	**1.00**	T/O	**1.00**	T/O	**1.00**
Found violations in total			-		14	12		5		2	
Average time usage (hours)			-		1.91	6.57		17.08		24.00	
Comparison with LTL-Fuzzer on time usage			-		-	3.44x		8.93x		12.55x	

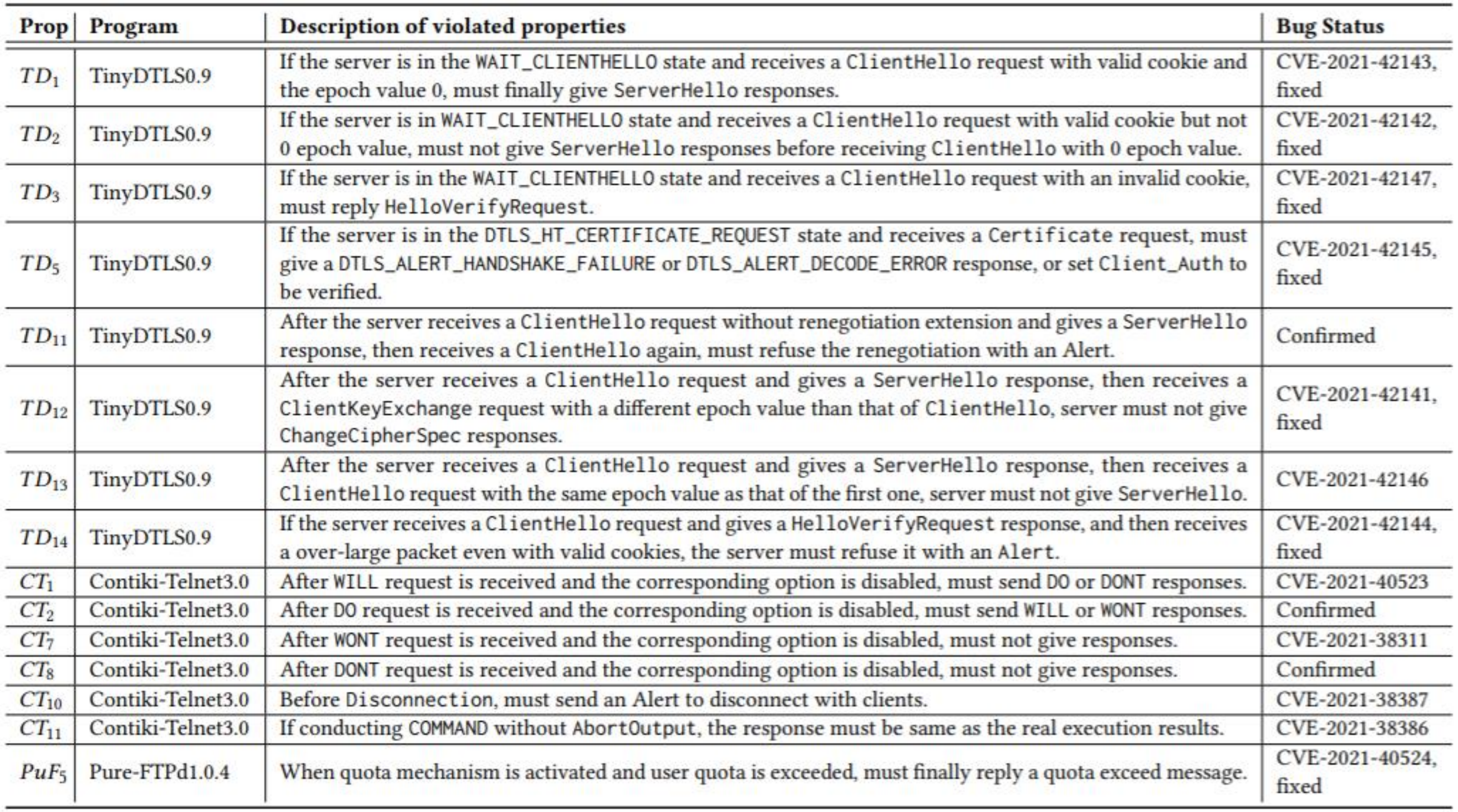
For RQ2 (Comparison):

- Our tool found the *most* violations
- Our tool was the *fastest*

For RQ1 (effectiveness):

LTL-Fuzzer discovered violations for *all* 14 properties derived from known CVEs

Usefulness



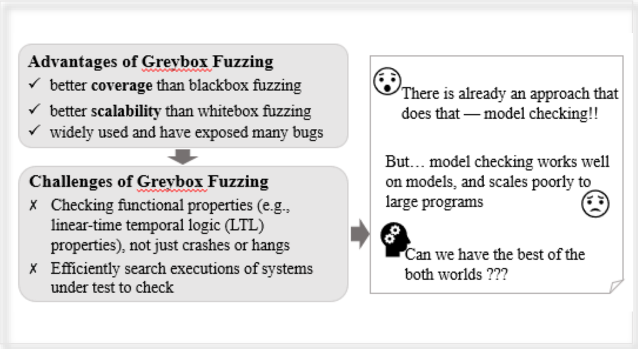
Prop	Program	Description of violated properties	Bug Status
TD_1	TinyDTLS0.9	If the server is in the WAIT_CLIENTHELLO state and receives a ClientHello request with valid cookie and the epoch value 0, must finally give ServerHello responses.	CVE-2021-42143, fixed
TD_2	TinyDTLS0.9	If the server is in WAIT_CLIENTHELLO state and receives a ClientHello request with valid cookie but not 0 epoch value, must not give ServerHello responses before receiving ClientHello with 0 epoch value.	CVE-2021-42142, fixed
TD_3	TinyDTLS0.9	If the server is in the WAIT_CLIENTHELLO state and receives a ClientHello request with an invalid cookie, must reply HelloVerifyRequest.	CVE-2021-42147, fixed
TD_5	TinyDTLS0.9	If the server is in the DTLS_HT_CERTIFICATE_REQUEST state and receives a Certificate request, must give a DTLS_ALERT_HANDSHAKE_FAILURE or DTLS_ALERT_DECODE_ERROR response, or set Client_Auth to be verified.	CVE-2021-42145, fixed
TD_{11}	TinyDTLS0.9	After the server receives a ClientHello request without renegotiation extension and gives a ServerHello response, then receives a ClientHello again, must refuse the renegotiation with an Alert.	Confirmed
TD_{12}	TinyDTLS0.9	After the server receives a ClientHello request and gives a ServerHello response, then receives a ClientKeyExchange request with a different epoch value than that of ClientHello, server must not give ChangeCipherSpec responses.	CVE-2021-42141, fixed
TD_{13}	TinyDTLS0.9	After the server receives a ClientHello request and gives a ServerHello response, then receives a ClientHello request with the same epoch value as that of the first one, server must not give ServerHello.	CVE-2021-42146
TD_{14}	TinyDTLS0.9	If the server receives a ClientHello request and gives a HelloVerifyRequest response, and then receives a over-large packet even with valid cookies, the server must refuse it with an Alert.	CVE-2021-42144, fixed
CT_1	Contiki-Telnet3.0	After WILL request is received and the corresponding option is disabled, must send DO or DONT responses.	CVE-2021-40523
CT_2	Contiki-Telnet3.0	After DO request is received and the corresponding option is disabled, must send WILL or WONT responses.	Confirmed
CT_7	Contiki-Telnet3.0	After WONT request is received and the corresponding option is disabled, must not give responses.	CVE-2021-38311
CT_8	Contiki-Telnet3.0	After DONT request is received and the corresponding option is disabled, must not give responses.	Confirmed
CT_{10}	Contiki-Telnet3.0	Before Disconnection, must send an Alert to disconnect with clients.	CVE-2021-38387
CT_{11}	Contiki-Telnet3.0	If conducting COMMAND without AbortOutput, the response must be same as the real execution results.	CVE-2021-38386
PuF_5	Pure-FTPd1.0.4	When quota mechanism is activated and user quota is exceeded, must finally reply a quota exceed message.	CVE-2021-40524, fixed

Extract 50 LTL properties from FTP, RTSP, SSL, SSH, DTLS and Telnet RFCs

For RQ3 (Usefulness):

Out of 50 LTL properties, **15** new property violations are found and **12** CVEs are assigned

Summary



Advantages of Greybox Fuzzing

- ✓ better **coverage** than blackbox fuzzing
- ✓ better **scalability** than whitebox fuzzing
- ✓ widely used and have exposed many bugs

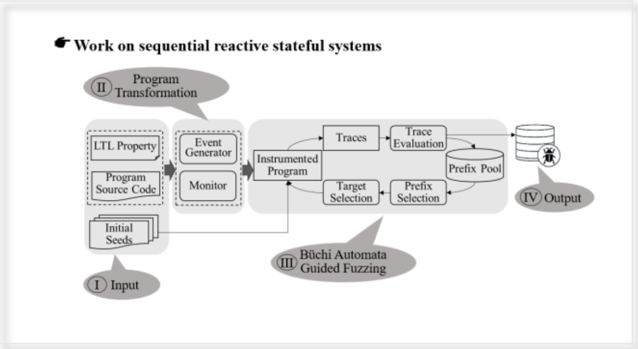
Challenges of Greybox Fuzzing

- ✗ Checking functional properties (e.g., linear-time temporal logic (LTL) properties), not just crashes or hangs
- ✗ Efficiently search executions of systems under test to check

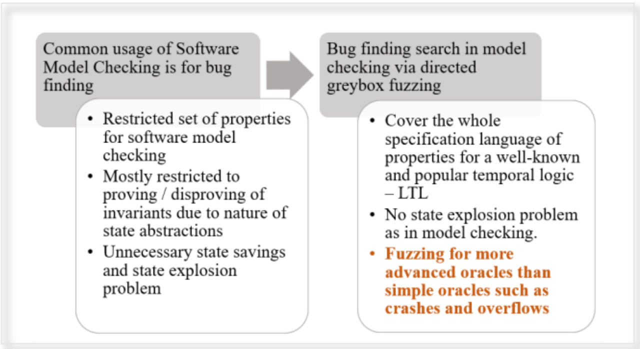
There is already an approach that does that — model checking!!

But… model checking works well on models, and scales poorly to large programs

Can we have the best of the both worlds ???



☛ **Work on sequential reactive stateful systems**

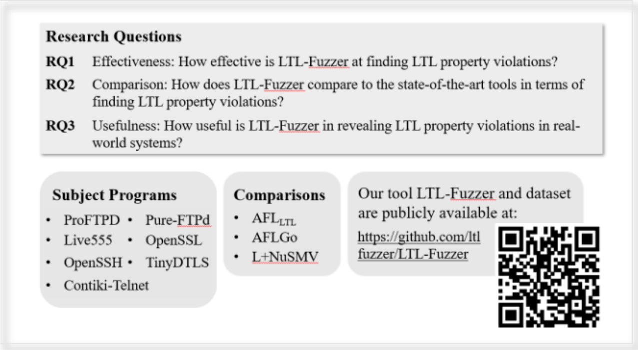


Common usage of Software Model Checking is for bug finding

- Restricted set of properties for software model checking
- Mostly restricted to proving / disproving of invariants due to nature of state abstractions
- Unnecessary state savings and state explosion problem

Bug finding search in model checking via directed greybox fuzzing

- Cover the whole specification language of properties for a well-known and popular temporal logic – LTL
- No state explosion problem as in model checking.
- **Fuzzing for more advanced oracles than simple oracles such as crashes and overflows**



Research Questions

RQ1 Effectiveness: How effective is LTL-Fuzzer at finding LTL property violations?

RQ2 Comparison: How does LTL-Fuzzer compare to the state-of-the-art tools in terms of finding LTL property violations?

RQ3 Usefulness: How useful is LTL-Fuzzer in revealing LTL property violations in real-world systems?

Subject Programs

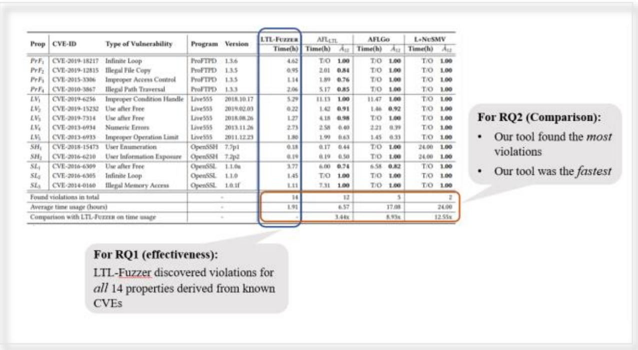
- ProFTPD
- Pure-FTPd
- Live555
- OpenSSL
- OpenSSH
- TinyDTLS
- Contiki-Telnet

Comparisons

- AFL_{LTL}
- AFLGo
- L+NuSMV

Our tool LTL-Fuzzer and dataset are publicly available at:

https://github.com/ltl-fuzzer/LTL-Fuzzer



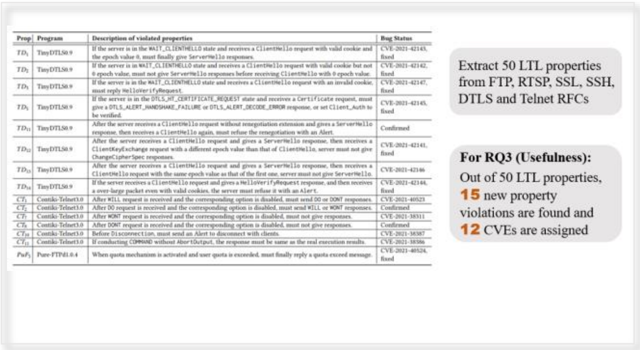
Prop	CVE-ID	Type of Vulnerability	Program	Version	LTL-FUZZER	AFL_{LTL}		AFLGo		L+NuSMV	
					Time(h)	Time(h)	$\hat{A}_{12}$	Time(h)	$\hat{A}_{12}$	Time(h)	$\hat{A}_{12}$
PrF_1	CVE-2019-18217	Infinite Loop	ProFTPD	1.3.6	4.62	T/O	**1.00**	T/O	**1.00**	T/O	**1.00**
PrF_2	CVE-2019-12815	Illegal File Copy	ProFTPD	1.3.5	0.95	2.01	**0.84**	T/O	**1.00**	T/O	**1.00**
PrF_3	CVE-2015-3306	Improper Access Control	ProFTPD	1.3.5	1.14	1.89	**0.76**	T/O	**1.00**	T/O	**1.00**
PrF_4	CVE-2010-3867	Illegal Path Traversal	ProFTPD	1.3.3	2.06	5.17	**0.85**	T/O	**1.00**	T/O	**1.00**
LV_1	CVE-2019-6256	Improper Condition Handle	Live555	2018.10.17	5.29	11.13	**1.00**	11.47	**1.00**	T/O	**1.00**
LV_2	CVE-2019-15232	Use after Free	Live555	2019.02.03	0.22	1.42	**0.91**	1.46	**0.92**	T/O	**1.00**
LV_3	CVE-2019-7314	Use after Free	Live555	2018.08.26	1.27	4.18	**0.98**	T/O	**1.00**	T/O	**1.00**
LV_4	CVE-2013-6934	Numeric Errors	Live555	2013.11.26	2.73	2.58	0.40	2.21	0.39	T/O	**1.00**
LV_5	CVE-2013-6933	Improper Operation Limit	Live555	2011.12.23	1.80	1.99	0.63	1.45	0.33	T/O	**1.00**
SH_1	CVE-2018-15473	User Enumeration	OpenSSH	7.7p1	0.18	0.17	0.44	T/O	**1.00**	24.00	**1.00**
SH_2	CVE-2016-6210	User Information Exposure	OpenSSH	7.2p2	0.19	0.19	0.50	T/O	**1.00**	24.00	**1.00**
SL_1	CVE-2016-6309	Use after Free	OpenSSL	1.1.0a	3.77	6.00	**0.74**	6.58	**0.82**	T/O	**1.00**
SL_2	CVE-2016-6305	Infinite Loop	OpenSSL	1.1.0	1.45	T/O	**1.00**	T/O	**1.00**	T/O	**1.00**
SL_3	CVE-2014-0160	Illegal Memory Access	OpenSSL	1.0.1f	1.11	7.31	**1.00**	T/O	**1.00**	T/O	**1.00**
Found violations in total			-		14		12		5		2
Average time usage (hours)			-		1.91		6.57		17.08		24.00
Comparison with LTL-FUZZER on time usage			-		-		3.44x		8.93x		12.55x

For RQ2 (Comparison):

- Our tool found the *most* violations
- Our tool was the *fastest*

For RQ1 (effectiveness):

LTL-Fuzzer discovered violations for *all* 14 properties derived from known CVEs



Prop	Program	Description of violated properties	Bug Status
TD_1	TinyDTLS0.9	If the server is in the WAIT_CLIENTHELLO state and receives a ClientHello request with valid cookie and the epoch value 0, must finally give ServerHello responses.	CVE-2021-42143, fixed
TD_2	TinyDTLS0.9	If the server is in WAIT_CLIENTHELLO state and receives a ClientHello request with valid cookie but not 0 epoch value, must not give ServerHello responses before receiving ClientHello with 0 epoch value.	CVE-2021-42142, fixed
TD_3	TinyDTLS0.9	If the server is in the WAIT_CLIENTHELLO state and receives a ClientHello request with an invalid cookie, must reply HelloVerifyRequest.	CVE-2021-42147, fixed
TD_5	TinyDTLS0.9	If the server is in the DTLS_HT_CERTIFICATE_REQUEST state and receives a Certificate request, must give a DTLS_ALERT_HANDSHAKE_FAILURE or DTLS_ALERT_DECODE_ERROR response, or set Client_Auth to be verified.	CVE-2021-42145, fixed
TD_{11}	TinyDTLS0.9	After the server receives a ClientHello request without renegotiation extension and gives a ServerHello response, then receives a ClientHello again, must refuse the renegotiation with an Alert.	Confirmed
TD_{12}	TinyDTLS0.9	After the server receives a ClientHello request and gives a ServerHello response, then receives a ClientKeyExchange request with a different epoch value than that of ClientHello, server must not give ChangeCipherSpec responses.	CVE-2021-42141, fixed
TD_{13}	TinyDTLS0.9	After the server receives a ClientHello request and gives a ServerHello response, then receives a ClientHello request with the same epoch value as that of the first one, server must not give ServerHello.	CVE-2021-42146
TD_{14}	TinyDTLS0.9	If the server receives a ClientHello request and gives a HelloVerifyRequest response, and then receives a over-large packet even with valid cookies, the server must refuse it with an Alert.	CVE-2021-42144, fixed
CT_1	Contiki-Telnet3.0	After WILL request is received and the corresponding option is disabled, must send DO or DONT responses.	CVE-2021-40523
CT_2	Contiki-Telnet3.0	After DO request is received and the corresponding option is disabled, must send WILL or WONT responses.	Confirmed
CT_7	Contiki-Telnet3.0	After WONT request is received and the corresponding option is disabled, must not give responses.	CVE-2021-38311
CT_8	Contiki-Telnet3.0	After DONT request is received and the corresponding option is disabled, must not give responses.	Confirmed
CT_{10}	Contiki-Telnet3.0	Before Disconnection, must send an Alert to disconnect with clients.	CVE-2021-38387
CT_{11}	Contiki-Telnet3.0	If conducting COMMAND without AbortOutput, the response must be same as the real execution results.	CVE-2021-38386
PuF_5	Pure-FTPd1.0.4	When quota mechanism is activated and user quota is exceeded, must finally reply a quota exceed message.	CVE-2021-40524, fixed

Extract 50 LTL properties from FTP, RTSP, SSL, SSH, DTLS and Telnet RFCs

For RQ3 (Usefulness):

Out of 50 LTL properties, **15** new property violations are found and **12** CVEs are assigned

THANKS!!