UNIVERSITÀ DEGLI STUDI DI CATANIA

27/09/2024

Benchmarking Stateful Fuzzers over Lighttpd

Leonardo Cantarella

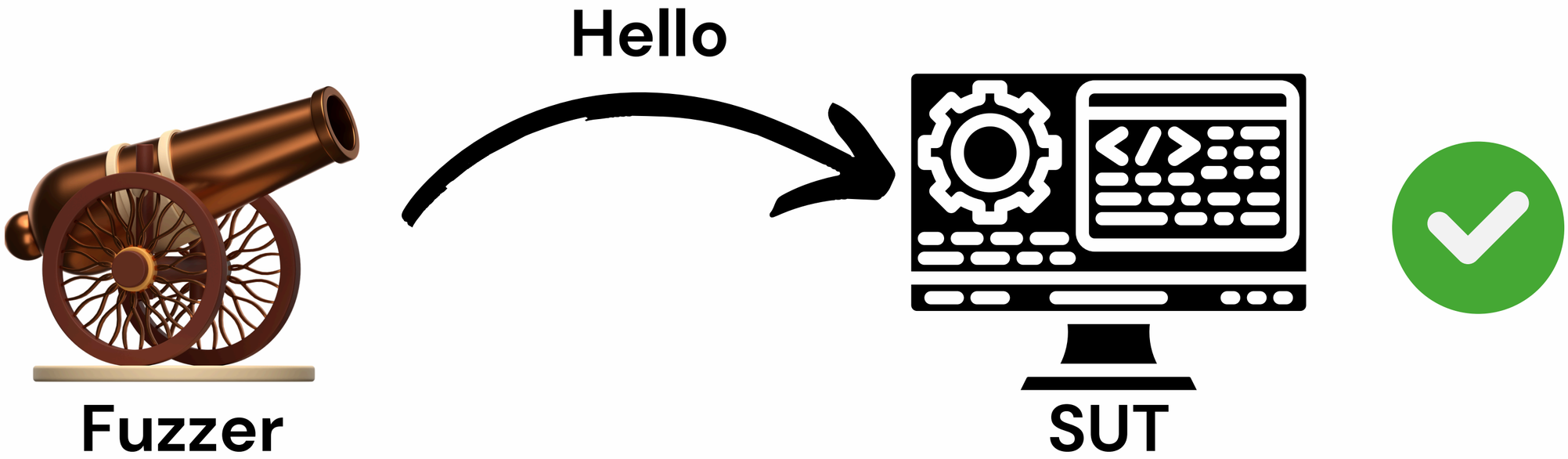
Informatica (L-31)

Relatore: Prof. Giampaolo Bella

Correlatore: Dott. Marcello Maugeri

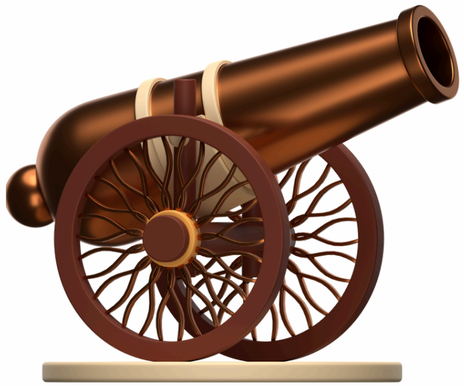
Correlatore: Dott. Cristian Daniele

Fuzzing

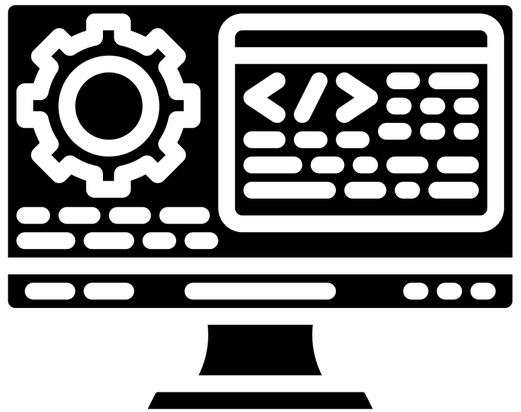


Fuzzing

AAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAA



Fuzzer



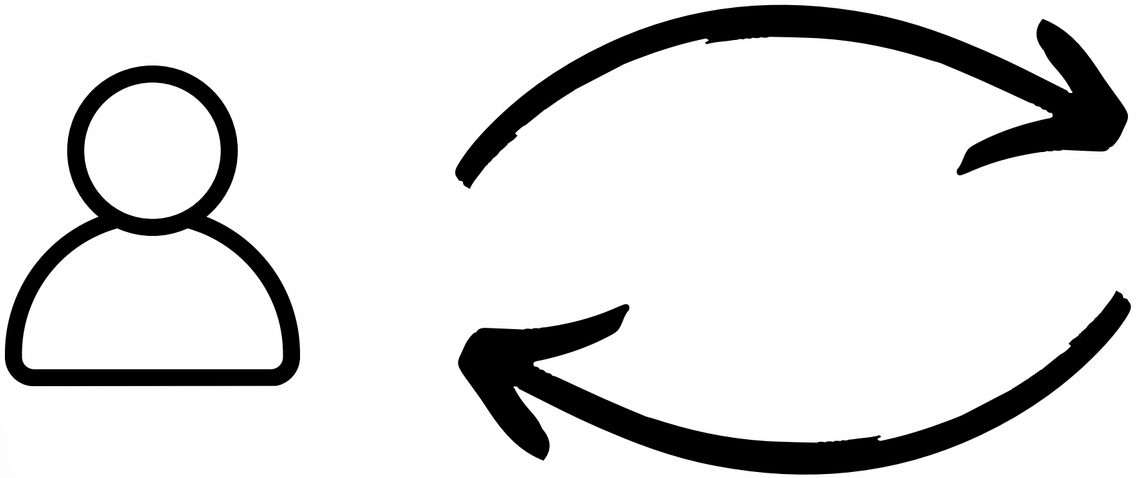
SUT

Stateless Fuzzing

Stateful Fuzzing

GET /hello.txt

Q = [“GET …”, “PUT …”, “GET …”, …]



Client

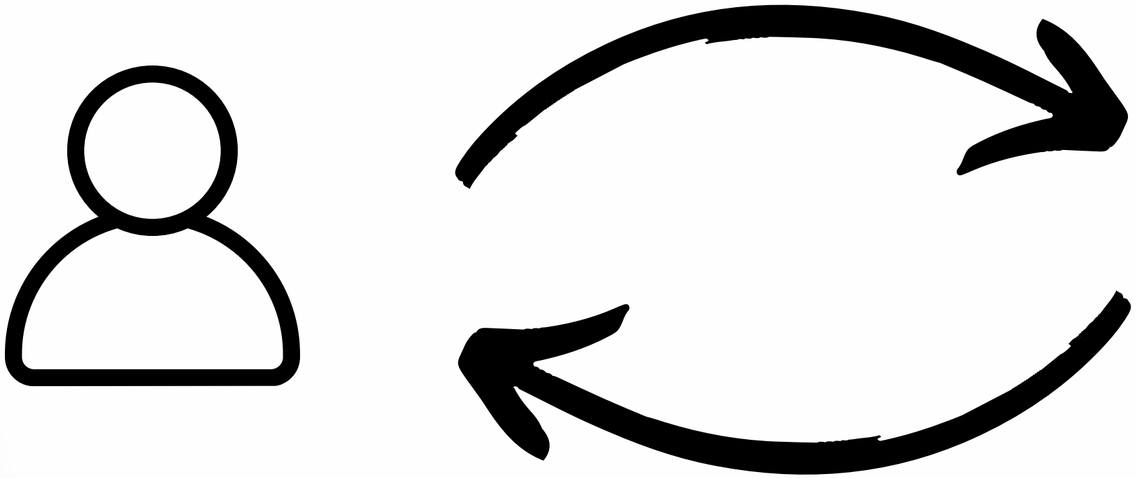
404 – Not Found

Web Server

Stateful Fuzzing

PUT /hello.txt

Q = [“PUT …”, “GET …”, …]



Client

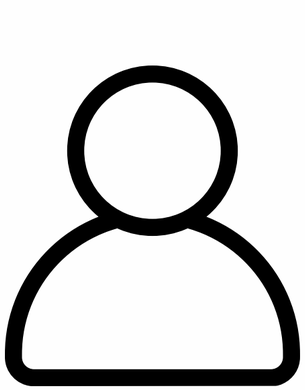
200 – OK

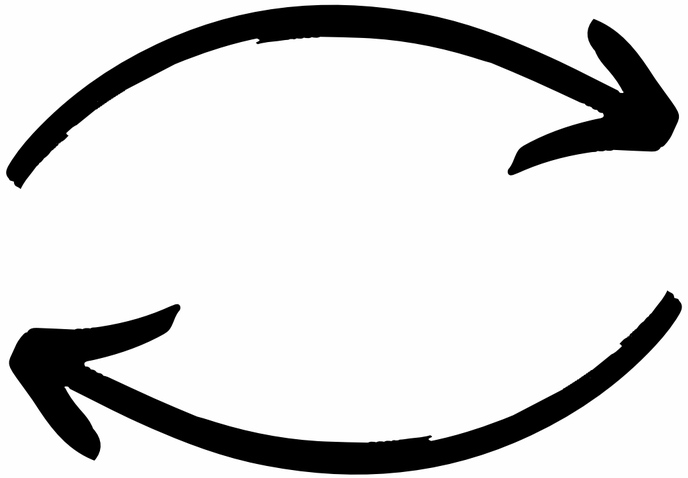
Web Server

Stateful Fuzzing

GET /hello.txt

Q = [“GET …”, …]





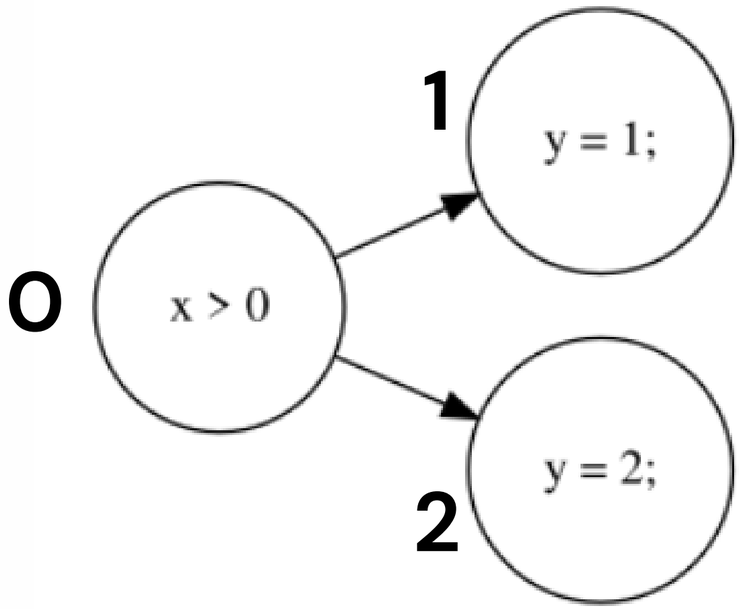
Client

200 – OK

Web Server

Edge Coverage

```
if (x > 0) {
    y = 1;
else {
    y = 2;
}
```



	0	1	2
0	0	1	0
1	0	0	0
2	0	0	0

Code **Edge Graph** **Coverage Map**

Lighttpd

- **Server web.**
- **Molte connessioni simultanee.**
- **Basso consumo di CPU e memoria** (architettura asincrona basata su eventi).
- **Sicurezza** (SSL/TLS).
- **Moduli per estendere le funzionalità** (gestione del traffico, compressione dinamica e riscrittura delle URL).
- **Leggero e flessibile** (usato in passato da Youtube e Wikipedia).

AFLNet and ChatAFL

AFLNet

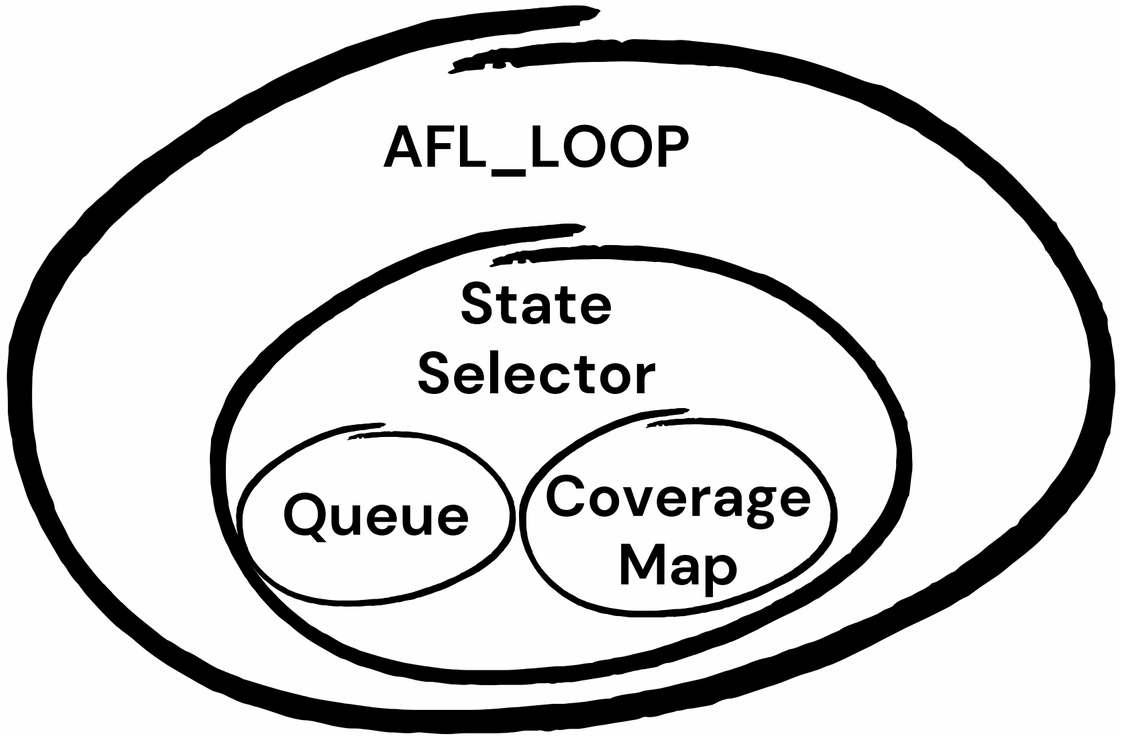
- **Primo fuzzer stateful**.
- Combina **codice di ritorno** e coverage per comprendere lo stato.
- Limiti: ”**coverage plateau**”, riavvio ad ogni traccia

ChatAFL

- Basato su **AFLNet**.
- Usa **LLM (ChatGPT3.5 Turbo)**.
- **Supera il** “coverage plateau” di AFLNet
- Limiti: riavvio ad ogni traccia

Fallaway

- **Persistent mode.**
- **AFL_LOOP.**
- **Code separate.**
- **Coverage map separate.**
- **Tempo di esecuzione** ed **efficienza.**



Setup

Enabling persistent mode

src/connections.c CHANGED

```
@@ -151,19 +151,25 @@ static void connection_handle_shutdown(connection *con) {
151 151          connection_reset(con);
152 152
153 153          /* close the connection */
154     -        if (con->fd >= 0
155     -            && (con->is_ssl_sock || 0 == shutdown(con->fd, SHUT_WR))) {
156     -            con->close_timeout_ts = log_monotonic_secs;
157     -
158     -            request_st * const r = &con->request;
159     -            connection_set_state(r, CON_STATE_CLOSE);
160     -            if (r->conf.log_state_handling) {
161     -                log_error(r->conf.errh, __FILE__, __LINE__,
162     -                  "shutdown for fd %d", con->fd);
163     -            }
164     -        } else {
165     -            connection_close(con);
166     -        }
    154 +        // if (con->fd >= 0
    155 +        //     && (con->is_ssl_sock || 0 == shutdown(con->fd, SHUT_WR))) {
    156 +        //     con->close_timeout_ts = log_monotonic_secs;
    157 +
    158 +        //     request_st * const r = &con->request;
    159 +        //     connection_set_state(r, CON_STATE_CLOSE);
    160 +        //     if (r->conf.log_state_handling) {
    161 +        //         log_error(r->conf.errh, __FILE__, __LINE__,
    162 +        //           "shutdown for fd %d", con->fd);
    163 +        //     }
    164 +        // } else {
    165 +        //     connection_close(con);
    166 +        // }
    167 +    request_reset_ex(&con->request); /*(r->conf.* is still valid below)*/
    168 +    chunkqueue_reset(con->read_queue);
    169 +    con->request_count = 0;
    170 +    con->is_ssl_sock = 0;
    171 +    con->revents_err = 0;
    172 +    connection_set_state(&con->request, CON_STATE_REQUEST_START);
167 173  }
```

Setup

Enabling persistent mode

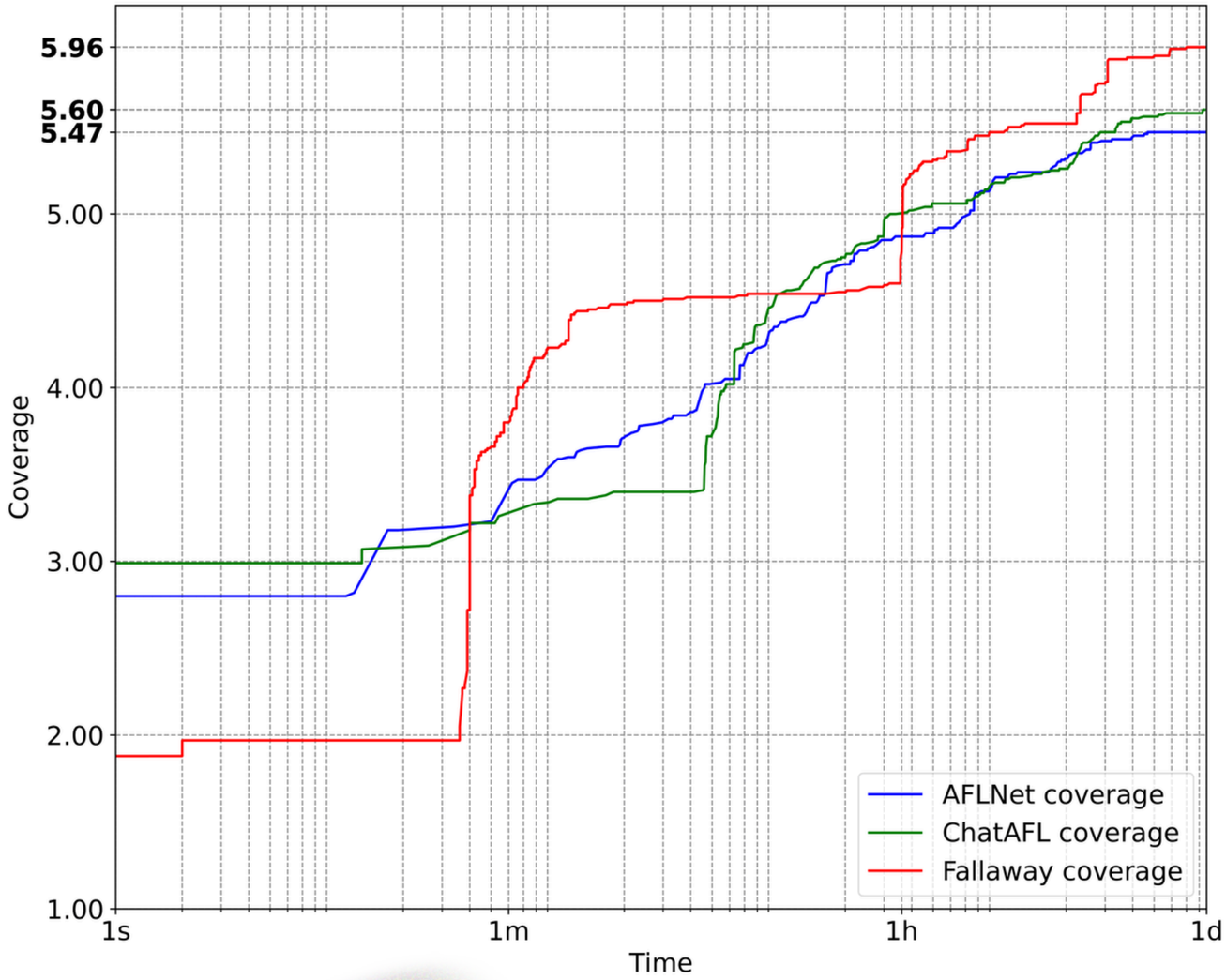
src/server.c CHANGED

```
@@ -2194,6 +2194,9 @@ static void server_main_loop (server * const srv) {
                        server_load_check(srv);
                }

+               while (__AFL_LOOP(INT64_MAX))
+               {
+                       fdevent_poll(srv->ev, -1);
        #ifndef _MSC_VER
                static
        #endif
@@ -2202,9 +2205,11 @@ static void server_main_loop (server * const srv) {
                connection * const joblist = log_con_jqueue;
                log_con_jqueue = sentinel;
                server_run_con_queue(joblist, sentinel);
+               }
+               srv_shutdown = 1;

-               if (fdevent_poll(srv->ev, log_con_jqueue != sentinel ? 0 : 1000) > 0)
-                       last_active_ts = log_monotonic_secs;
+               // if (fdevent_poll(srv->ev, log_con_jqueue != sentinel ? 0 : 1000) > 0
+               //      last_active_ts = log_monotonic_secs;
        }
  }
```

Results


5.96
5.60
5.47
5.00
4.00
3.00
2.00
1.00
Coverage
1s
1m
1h
1d
Time
AFLNet coverage
ChatAFL coverage
Fallaway coverage

Conclusions

Sono stati valutati tre fuzzer:

- **Fallaway** ha ottenuto i migliori risultati grazie all'elevato throughput, reso possibile dalla persistent mode, nonostante un approccio più semplice.
- **ChatAFL**, pur utilizzando LLM, ha superato leggermente **AFLNet**, senza però distaccarsi in modo significativo.

Fallaway rappresenta la base per un progetto più ampio, frutto della collaborazione tra l'Università di Catania, Radboud e Twente. Un possibile sviluppo futuro riguarda il fuzzing di sistemi multiprocesso, che presentano la sfida di gestire lo stato e il feedback tra i processi figli e quindi uno studio approfondito sulla intercomunicazione tra processi (**IPC**).

Thank you for your attention !