Supporting Secure Coding with RefactorErl

Brigitta Baranyai
Eötvös Loránd University
Budapest, Hungary

SZÉCHENYI 2020
HUNGARIAN GOVERNMENT
European Union
European Social Fund
INVESTING IN YOUR FUTURE

RESEARCH BACKGROUND

- Growing number of cyber threats in the era of Internet.
- In order to improve the security of the systems, there are several standards and static analyser tools.
- The lack of security analyser tools in case of Erlang.

```
-module(injection).
-export([run_cmd/1]).

run_cmd(Input) ->
    os:cmd("cat "++ Input).
```

→

```
1>injection:run_cmd("test.txt;ls").
"Hello World! test.txt"
```

SECURE CODING

Distributed processes run isolated with their own resources.

Immutable data structures.

Pure functions, modularity.

Fault tolerance as a core language concept.

04

SECURE CODING IN ERLANG

Interoperability mechanism related vulnerabilities

Concurrent programming related issues

Distributed programming related issues

Injection

Memory overload related attacks

INTEROPERABILITY MECHANISM RELATED VULNERABILITIES

Using Erlang ports:

```
-module(complex1).
-export([start/1, init/1]).

start(ExtPrg) ->
    spawn(?MODULE, init, [ExtPrg]).

init(ExtPrg) ->
    register(complex, self()),
    process_flag(trap_exit, true),
    loop(open_port({spawn, ExtPrg}),
            [{packet, 2}])).
```

Using dynamically loaded libraries (erl_ddll) or NIF:

```
-module(complex2).
-export([foo/1]).
-on_load(init/0).

init() ->
    {ok, ExtPrg} = io:read("Provide a program..."),
    ok = erlang:load_nif(ExtPrg, 0).

foo(_X) -> exit(nif_library_not_loaded).
```

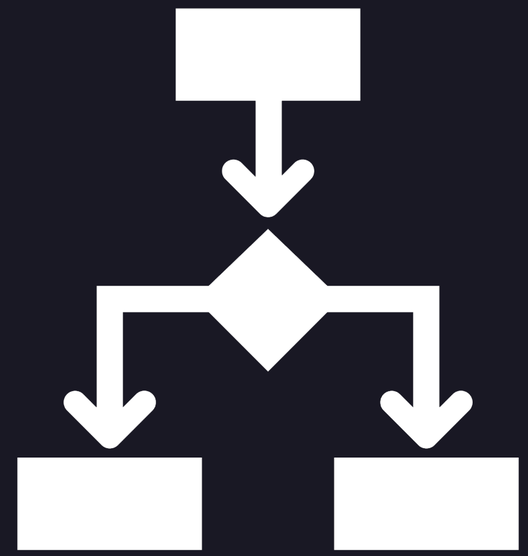
06

CONCURRENT PROGRAMMING RELATED ISSUES

Not connecting processes in an atomic way

Modifying process priority

ETS traversal without table fixes



DISTRIBUTED PROGRAMMING RELATED ISSUES

Using the network kernel related functions:

```
net_kernel:allow/1,
net_kernel:connect_node/1,
net_kernel:start/1
```

SSL-3.0 and TLS-1.0 protocol configuration options for communication over sockets via the ssl module which can call forft Man-in-the-middle attacks:

```
ssl:connect("example.net", 443, [
    {padding_check, false},
    {beast_mitigation, disabled},
    {fallback, true}
]).
```

With the evolution of the OpenSSL package some of the functions of the crypto module became obsolete:

```
crypto:block_encrypt/3/4,
crypto:block_decrypt/3/4,
crypto:cmac/3/4, crypto:hmac/3/4, ...
```

INJECTION

OS commands called with unknown input:

```
-module(injection).
-export([run_cmd/1]).

run_cmd(Input) ->
    os:cmd("cat " ++ Input).
```

File related operations with unknown input data:

```
-module(injection).
-export([eval/1]).

eval(Input) ->
    file:eval(Input).
```

Dynamically loaded program code coming from unknown data source:

```
-module(injection).
-export([load/1]).

load(Input) ->
    code:load_file(Input).
```

MEMORY OVERLOAD RELATED ATTACKS

Dynamic atom creation related functions:

```
parse_uri(Input) ->
    http_uri:parse(Input, [{ipv6_host_with_brackets, true}]).
```

XML parsing related functions without the usage of proper event handlers for preventing the internal or external entity expansion:

```
parse_xml(Input) ->
    xmerl_sax_parser:stream(Input, []).
```

REFACTORERL

Static source code analyser tool.

Source code transformations without behaviour change.

Helps in understanding huge code bases, their maintenance or even investigating bugs by tracing back their origin.

Integrates well with editors like Emacs, Vim, Visual Studio Code and Eclipse.

Provides a web interface or command line tool through interactive shell.

THE SECURITY CHECKER OF REFACTORERL

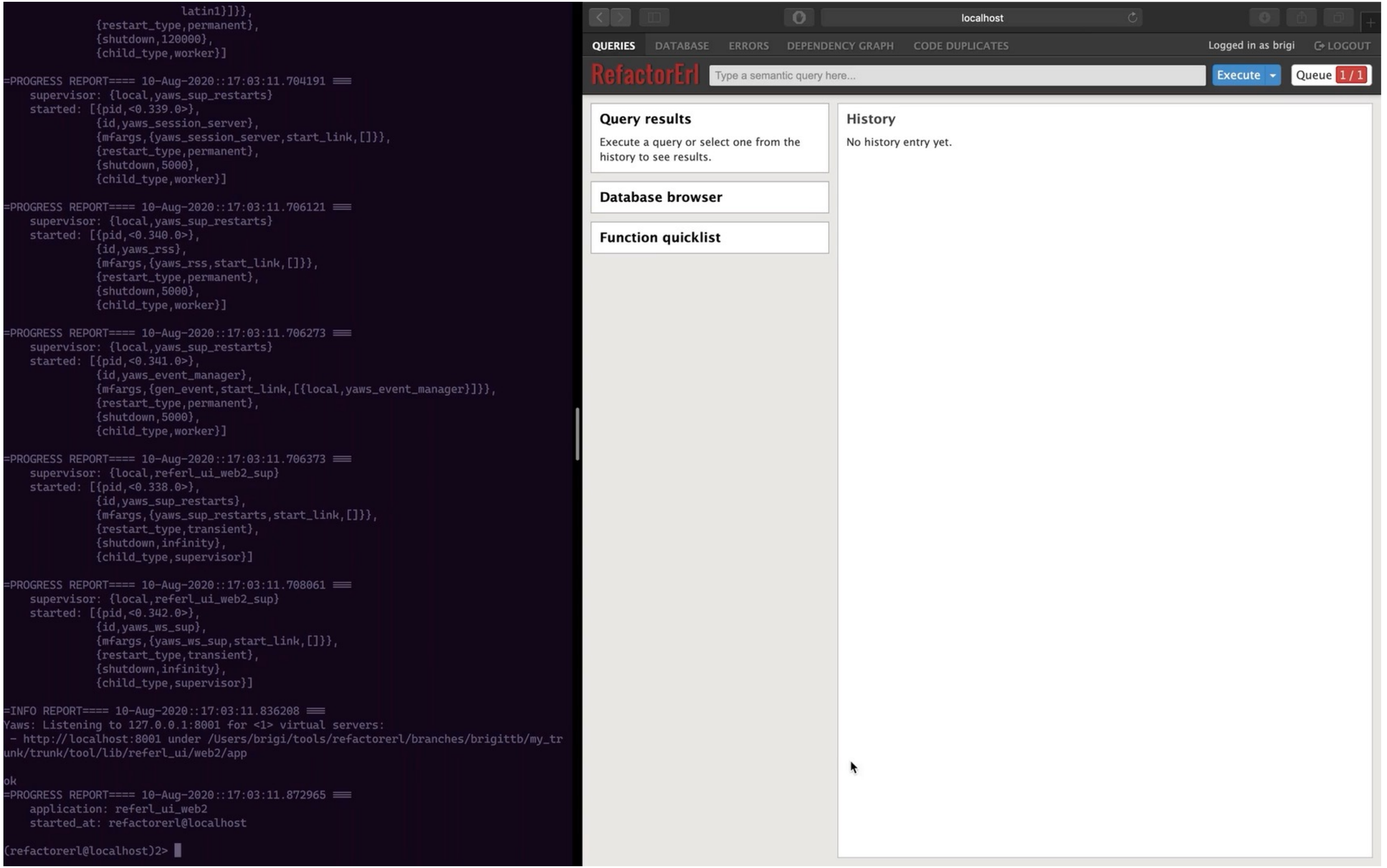
Similar algorithm for all the attack types.

- Determine the function call locations which are associated with unsecure operations.
- Select the functions parameters that can be associated with potential vulnerabilities.
- Run dataflow analysis on the sensitive parameters.
- Flag parameters with unknown source.
- Filter out functions provided by the users for input validation.

SEMANTIC QUERY LANGUAGE OF REFACTORERL

Provides syntactic and semantic information about Erlang programs by querying the call chains, function calls appearing in expressions, etc.

The units of the query language correspond to the semantic language elements of Erlang, which include the following: files, functions, function parameters, expressions, variables, etc.

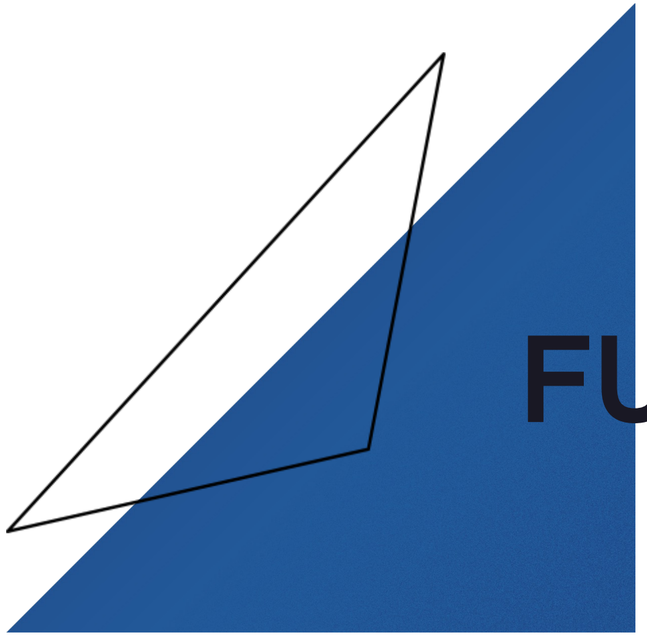

latin1}]}},
{restart_type,permanent},
{shutdown,120000},
{child_type,worker}]
=PROGRESS REPORT==== 10-Aug-2020::17:03:11.704191 ===
supervisor: {local,yaws_sup_restarts}
started: [{pid,<0.339.0>},
{id,yaws_session_server},
{mfargs,{yaws_session_server,start_link,[]}},
{restart_type,permanent},
{shutdown,5000},
{child_type,worker}]
=PROGRESS REPORT==== 10-Aug-2020::17:03:11.706121 ===
supervisor: {local,yaws_sup_restarts}
started: [{pid,<0.340.0>},
{id,yaws_rss},
{mfargs,{yaws_rss,start_link,[]}},
{restart_type,permanent},
{shutdown,5000},
{child_type,worker}]
=PROGRESS REPORT==== 10-Aug-2020::17:03:11.706273 ===
supervisor: {local,yaws_sup_restarts}
started: [{pid,<0.341.0>},
{id,yaws_event_manager},
{mfargs,{gen_event,start_link,[{local,yaws_event_manager}]}},
{restart_type,permanent},
{shutdown,5000},
{child_type,worker}]
=PROGRESS REPORT==== 10-Aug-2020::17:03:11.706373 ===
supervisor: {local,referl_ui_web2_sup}
started: [{pid,<0.338.0>},
{id,yaws_sup_restarts},
{mfargs,{yaws_sup_restarts,start_link,[]}},
{restart_type,transient},
{shutdown,infinity},
{child_type,supervisor}]
=PROGRESS REPORT==== 10-Aug-2020::17:03:11.708061 ===
supervisor: {local,referl_ui_web2_sup}
started: [{pid,<0.342.0>},
{id,yaws_ws_sup},
{mfargs,{yaws_ws_sup,start_link,[]}},
{restart_type,transient},
{shutdown,infinity},
{child_type,supervisor}]
=INFO REPORT==== 10-Aug-2020::17:03:11.836208 ===
Yaws: Listening to 127.0.0.1:8001 for <1> virtual servers:
- http://localhost:8001 under /Users/brigi/tools/refactorerl/branches/brigittb/my_trunk/trunk/tool/lib/referl_ui/web2/app
ok
=PROGRESS REPORT==== 10-Aug-2020::17:03:11.872965 ===
application: referl_ui_web2
started_at: refactorerl@localhost
(refactorerl@localhost)2>
localhost
QUERIES
DATABASE
ERRORS
DEPENDENCY GRAPH
CODE DUPLICATES
Logged in as brigi
LOGOUT
RefactorErl
Type a semantic query here...
Execute
Queue 1 / 1
Query results
Execute a query or select one from the history to see results.
Database browser
Function quicklist
History
No history entry yet.

RESULTS OF THE SECURITY CHECKER OF REFACTORERL

```
(refactorerl@localhost)31> ri:q("mods.funs.unsecure_calls").
coap_client:resolve_uri/1
    {ok, {Scheme, _UserInfo, Host, PortNo, Path, Query}} =
        http_uri:parse(Uri, [{scheme_defaults, [{coap, ?DEFAULT_COAP_PORT},
 {coaps, ?DEFAULT_COAPS_PORT}]}])
coap_server_content:filter/2
    filter(
        case binary:split(Search, <<$=>>) of
            [Name0, Value0] ->
                Name = list_to_atom(binary_to_list(Name0)),
                Value = wildcard_value(Value0),
                lists:filter(
                    fun (Link) -> match_link(Link, Name, Value) end,
                    Links);
            _Else ->
                Links
        end,
        Query)
ok
(refactorerl@localhost)32>
```

RESULTS OF THE SECURITY CHECKER OF PEST

```
brigi@debVM:~/Projects/erlang$ ~/Projects/pest/pest/pest.erl
-r -s 0 relayr/gen_coap/_build/default/lib/gen_coap/ebin/
 15: Keep OpenSSL updated for crypto module use (run with "-V
 crypto")
     coap_dtls_listen.beam:19 (ssl:_/_)
     coap_dtls_socket.beam:[32,43,47,60,64] (ssl:_/_)
brigi@debVM:~/Projects/erlang$
```



FUTURE WORK

Additional rules to identify race conditions, obsolete cypher algorithms from the crypto module.

Add security level related settings, configurable analysis to further enhance the user experience.

THANK YOU FOR YOUR ATTENTION!

This work was supported by the construction EFOP-3.6.3-VEKOP-16-2017-00002. The project was supported by the European Union, co-financed by the European Social Fund.

SZÉCHENYI 2020

HUNGARIAN GOVERNMENT

European Union
European Social Fund

INVESTING IN YOUR FUTURE