LuaJIT FFI

Kit Chan (kichan@apache.org)

Simple Example

```
local ffi= require("ffi")
ffi.cdef[[
typedef struct db DB;
typedef DB ca_cert_db;
ca_cert_db *ca_open_cert_db();
char *ca_get_cert(ca_cert_db *db, const char *appid);
int ca_close_cert_db(ca_cert_db *db);
]]
local ca= ffi.load("/usr/local/lib/libca.so.1")
```

Simple Example

```
function do_global_read_request()
  local cert_db= ca.ca_open_cert_db()
  local cert = ca.ca_get_cert(cert_db, “myapp”)
  local close = ca.ca_close_cert_db(cert_db)
  if(cert == nil) then
    ts.error(‘Certificate cannot be retrieved, just return ')
    return 0
  end
  local cert_str = ffi.string(cert)
  ts.client_request.header[‘X-Cert’] = cert_str
end
```

LuaJIT FFI

- Why?
  - Reuse existing code
  - Easy to develop & deploy
- Why Not?
  - Safety issue

Advanced Use - Modsecurity

- Opensource WAF - e.g. IronBee
- OWASP Core Rule Set
- Was closely Integrated with Apache (with Nginx support as well)

ModSecurity Example

GET /index.php?testparam=test HTTP/1.1

SecRule ARGS:testparam “@contains test” “id:1234,deny,status:403”

ModSecurity V3

- No more dependency on Apache or APR
- Generic library

Integrating with ATS

- We can build C / C++ Plugins
- Lua (WIP) - https://github.com/shukitchan/ats-luajit-modsecurity/blob/master/ats-luajit-modsecurity.lua
- Concerns
  - Performance
  - Request and response body examination
  - Lua scripting allowed inside Modsecurity V3 engine

```
local ffi = require("ffi")

ffi.cdef[[

typedef struct ModSecurity ModSecurity;
ModSecurity* msc_init();
void msc_set_connector_info(ModSecurity *msc, const char *connector);
void msc_cleanup(ModSecurity *msc);

typedef struct Rules Rules;
Rules* msc_create_rules_set();
int msc_rules_add_file(Rules *rules, const char *file, const char **error);
int msc_rules_cleanup(Rules *rules);

typedef struct Transaction Transaction;
Transaction *msc_new_transaction(ModSecurity *ms, Rules *rules, void *logCbData);
int msc_process_connection(Transaction *transaction, const char *client, int cPort, const char *server, int sPort);
int msc_process_uri(Transaction *transaction, const char *uri, const char *protocol, const char *http_version);
int msc_add_request_header(Transaction *transaction, const unsigned char *key, const unsigned char *value);
int msc_process_request_headers(Transaction *transaction);
int msc_add_response_header(Transaction *transaction, const unsigned char *key, const unsigned char *value);
int msc_process_response_headers(Transaction *transaction, int code, const char* protocol);
int msc_process_logging(Transaction *transaction);
void msc_transaction_cleanup(Transaction *transaction);

typedef struct ModSecurityIntervention_t {
    int status;
    int pause;
    char *url;
    char *log;
    int disruptive;
} ModSecurityIntervention;
int msc_intervention(Transaction *transaction, ModSecurityIntervention *it);

]]

local msc = ffi.load("/usr/local/modsecurity/lib/libmodsecurity.so")
```

Latest on ATS Lua Plugin

Kit Chan (kichan@apache.org)

Latest Features on 8.0.0

- Promoted to stable
- Option for global plugin script to be reloaded without restart
- Support more API functions
  - TSRemapFromUrlGet / TSRemapToUrlGet
  - TSHttpTxnOutgoingAddrSet
  - TSHttpTxnNextHopAddrGet
  - TSHttpTxnServerFdGet / TSHttpTxnClientFdGet
  - TSHostLookup
  - TSInstallDirGet / TSConfigDirGet / TSRuntimeDirGet / TSPluginDirGet
  - TSTrafficServerVersionGet
  - TSHttpSsnIdGet
  - TSIOBufferWaterMarkSet / TSIOBufferWaterMarkGet
  - TSHttpTxnAborted

Future Plan

- Support for building with Lua as alternate choice for LuaJIT
- Support for SSL session API, Lifecycle API, early intervention API, TLS User agent hooks
- ???

Backup slides

Example - Defining functions, data structures, data types to be used

```
local ffi = require("ffi")
ffi.cdef[[
static const int IPPROTO_TCP = 6;
static const int TCP_INFO = 11;

typedef struct tcp_info {
   ..... // not listing the whole structure for fitting into this slide
} info;

typedef uint32_t socklen_t;

int getsockopt(int sockfd, int level, int optname, void *optval, socklen_t *optlen);
]]
```

Example - Using the data structures, data types and functions

```
local C = ffi.C

function do_global_read_request()
   local fd = ts.http.get_client_fd() or ''
   local inf = ffi.new("info")
   local inf_size = ffi.sizeof(inf)
   local siz = ffi.new("socklen_t[1]", inf_size)
   local rc = C.getsockopt(fd, C.IPPROTO_TCP, C.TCP_INFO, inf, siz)
   ts.client_request.header['X-Tcp-Rtt'] = inf.tcpi_rtt
end
```

Example - MaxMind GeoIP and binding library

- Download and install GeoIP 1.6.12 - https://github.com/maxmind/geoip-api-c/releases
- Download and install legacy country database - https://dev.maxmind.com/geoip/legacy/install/country/
- Download and install luajit-geoip binding - https://github.com/leafo/luajit-geoip

Example - MaxMind GeoIP and binding library

```
ts.add_package_path('/usr/local/share/lua/5.1/?.lua')

local geoip = require 'geoip'

function do_global_send_response()
 local res = geoip.lookup_addr("8.8.8.8")
 ts.client_response.header['X-Country'] = res.country_code
end
```

FFI Performance

- https://github.com/dyu/ffi-overhead
- LuaJIT calling to C function is faster than C calling to C function
- https://nullprogram.com/blog/2018/05/27/
  - C Program calling C function in another shared object
    - Route through the Procedure Link Table (PLT)
    - PLT fetches address from Global Offset Table (GOT) and jumps there
  - JIT compiled (i.e LuaJIT) code can do direct C function calls instead