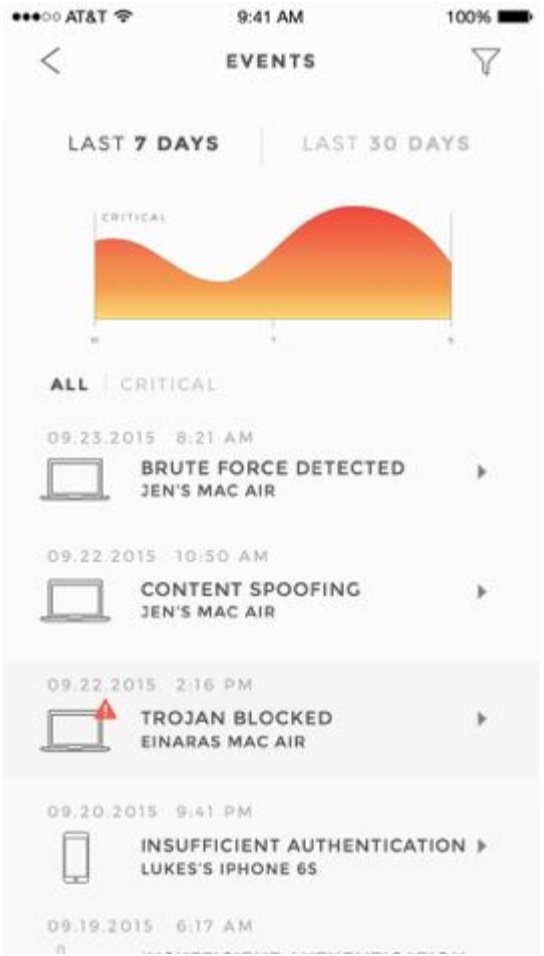
# CUJO - Safe Browsing with Lua

Lourival Vieira Neto <lourival.neto@getcujo.com>

# Introduction

- CUJO
  - Smart Firewall
  - Safe Browsing
  - Parental Controls



# Introduction

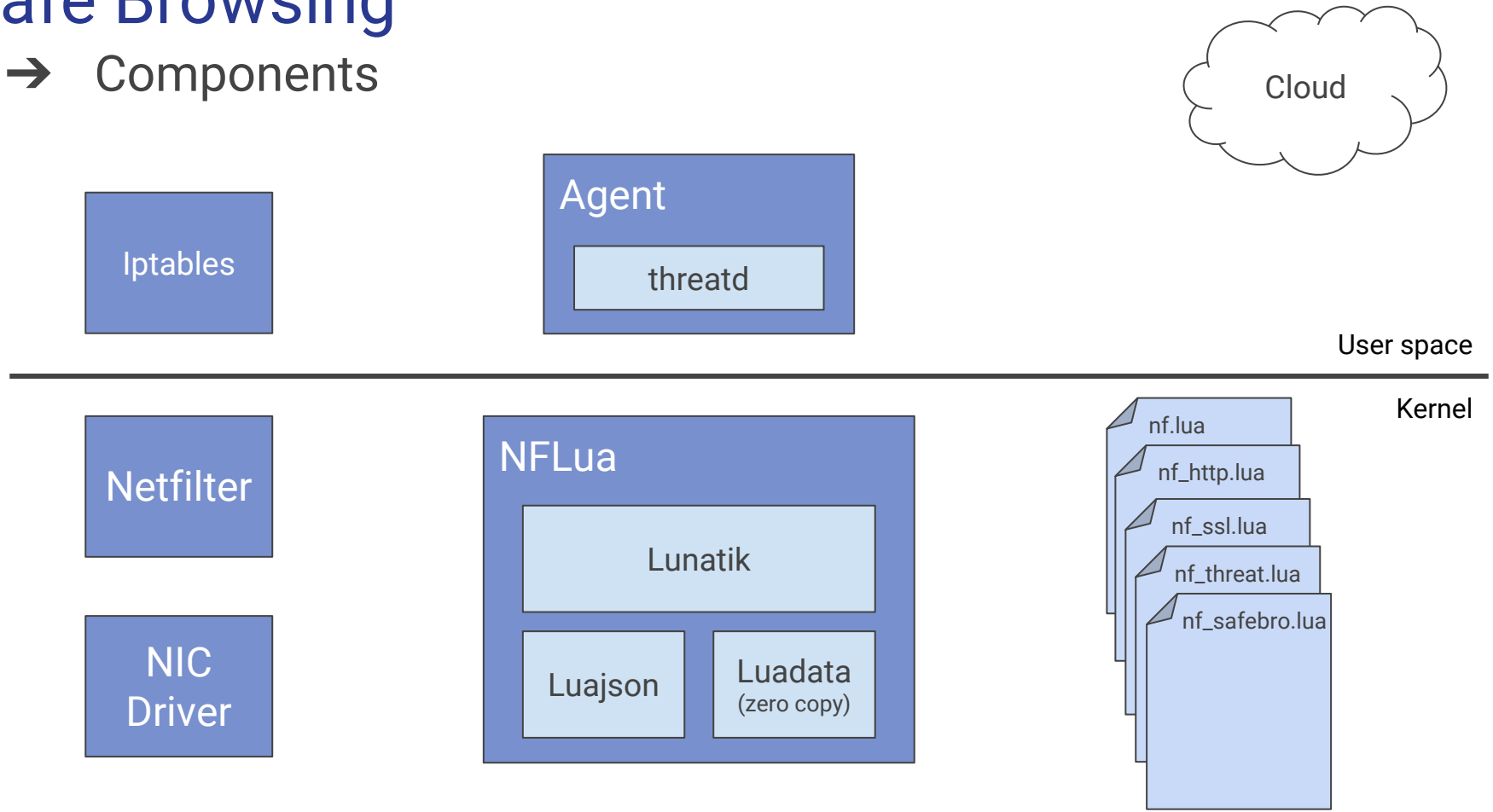
- CUJO Firmware Team
  - Gabriel Ligneul
  - Iruatã Souza
  - Katia Fernandes
  - Linas Nenorta
  - Lourival Vieira Neto
  - Marcel Moura
  - Savio Barbosa
  - Tadeu Bastos
  - Pedro Tammela

# Introduction

- Lunatik
  - Lua in the Linux Kernel
  - "Scriptable Operating Systems with Lua"
    - Vieira Neto, L., Ierusalimschy, R., de Moura, A.L. and Balmer, M.
- Luadata
  - “Zero-copy”
- NFLua
  - Netfilter Binding

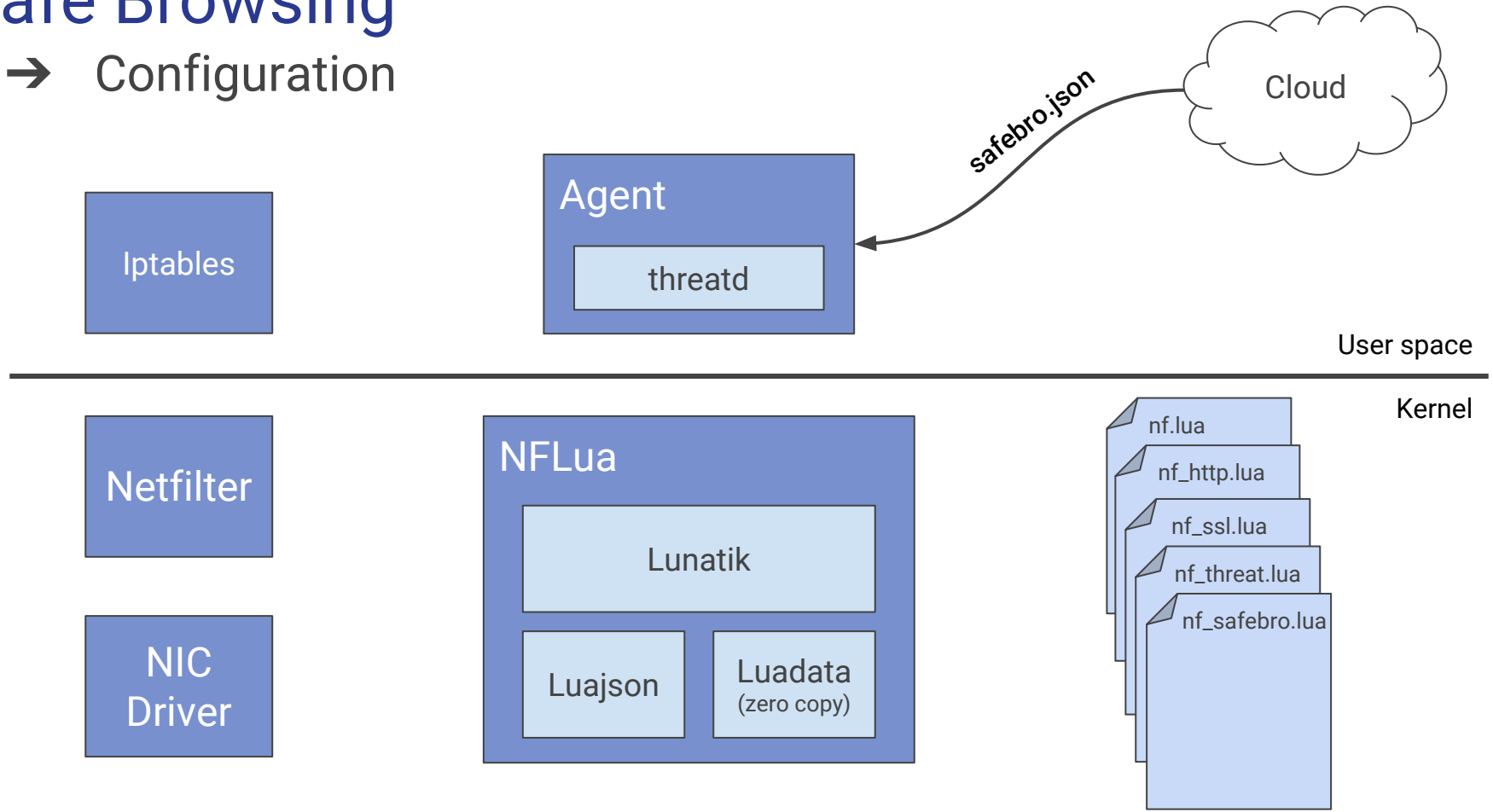
# Safe Browsing

➔ Components



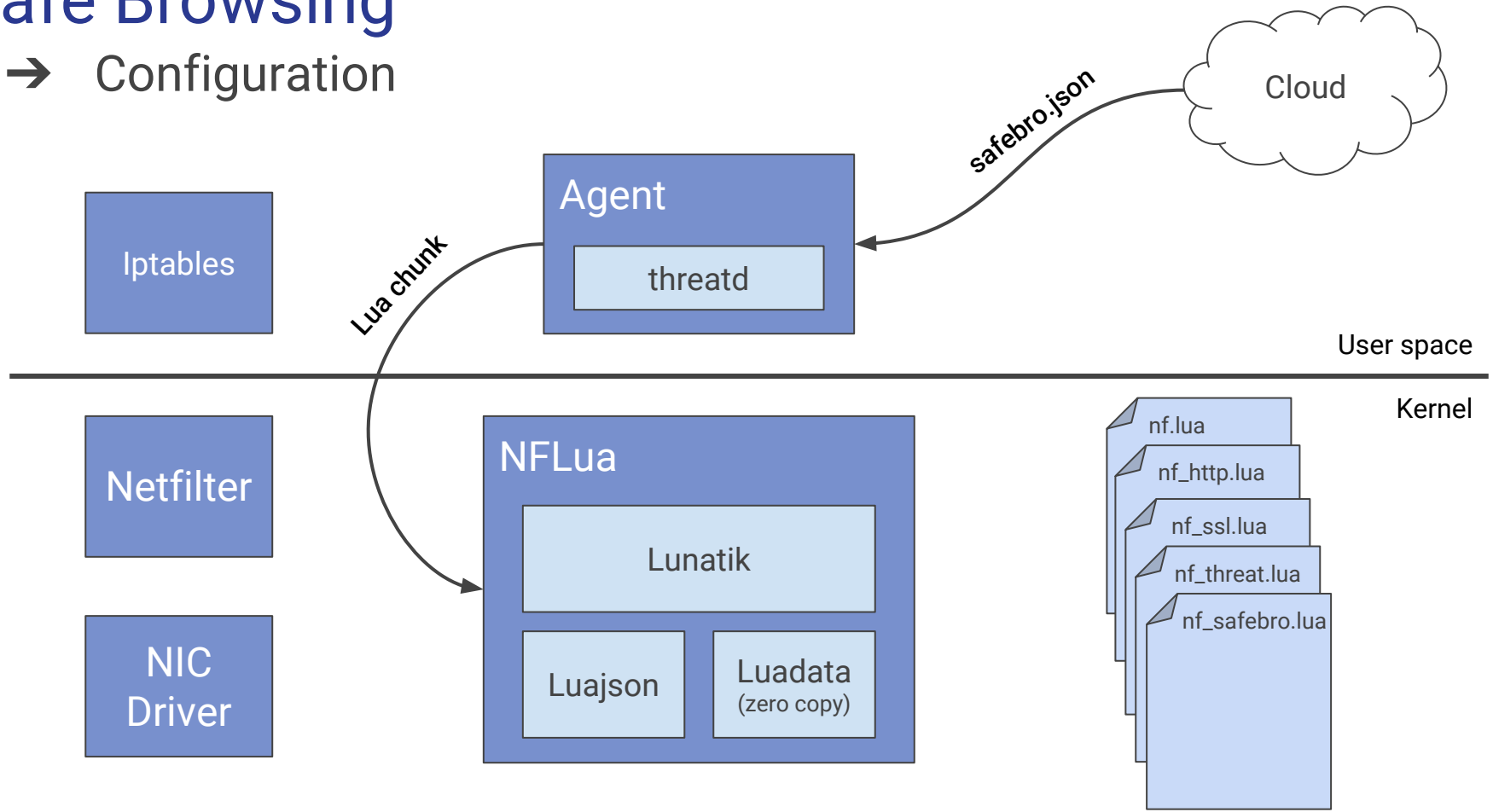
# Safe Browsing

➔ Configuration



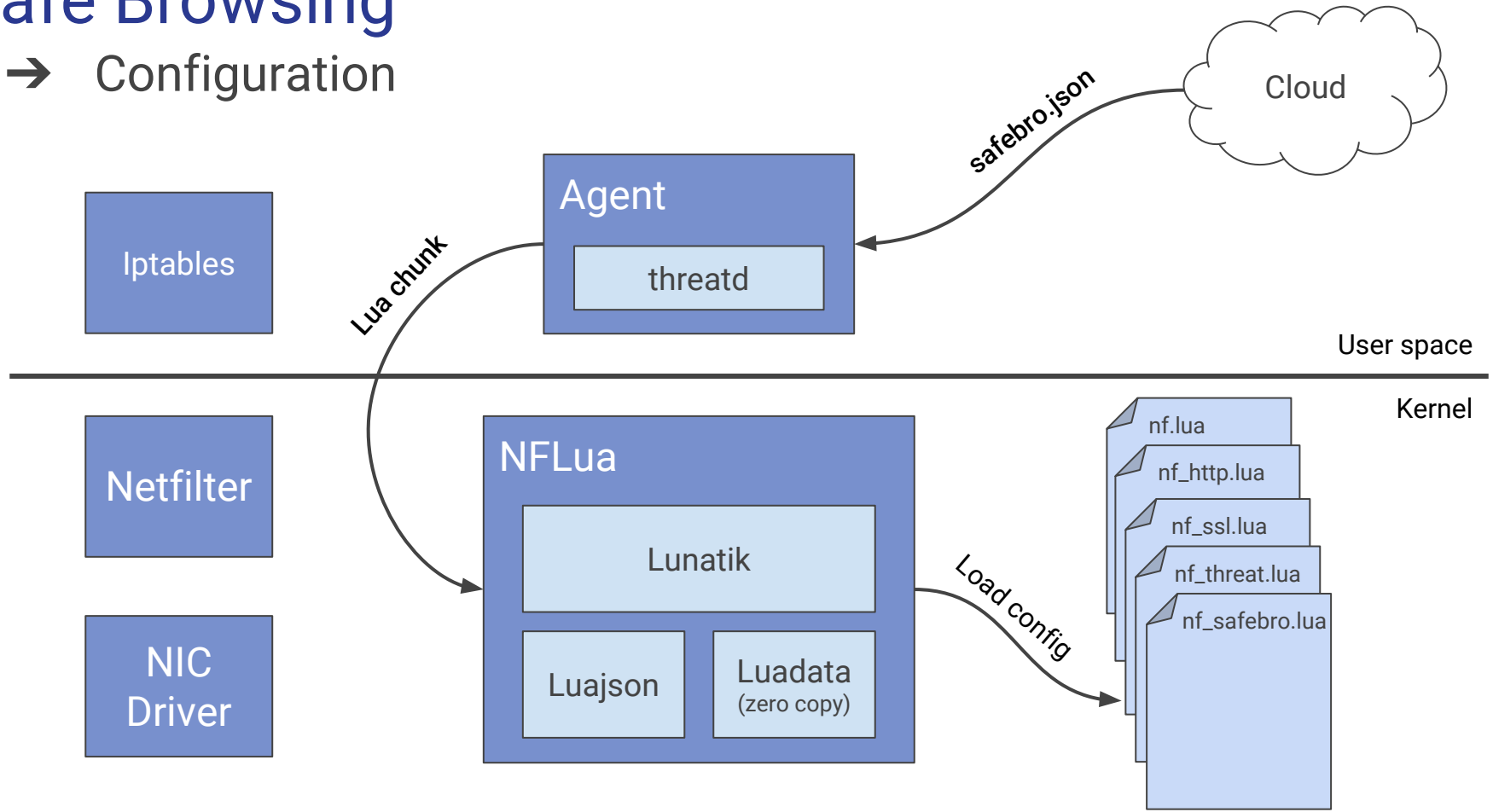
# Safe Browsing

➔ Configuration



# Safe Browsing

➔ Configuration



# Safe Browsing

➔ Configuration

```
# cat nf_{threat,safebro,http,ssl}.lua > /proc/nf_lua

# iptables -A FORWARD -p tcp --dport 80 --tcp-flags PSH PSH \
  -m lua --function nf_http -j DROP

# iptables -A FORWARD -p tcp --dport 443 --tcp-flags PSH PSH \
  -m lua --function nf_ssl -j REJECT --reject-with tcp-reset
```

# Safe Browsing

➔ Configuration

```
103  local function init()
104          local file = assert(io.open('/var/config/safebro/safebro.json', 'r'))
105          local conf = file:read'a'
106          local params = json.decode(conf)
107
108          webroot.init(params.deviceId, params.oem, params.uid, params.server)
109
110          local nflua = assert(io.open('/proc/nf_lua', 'w+'))
111          nflua:write(string.format('safebro.config[[%s]]', conf))
112          nflua:flush()
113
114          daemon(params)
115  end
```

threatd.lua

# Safe Browsing

➔ Configuration

```
203  static ssize_t nflua_write(struct file *file, const char __user *buf,
204                  size_t size, loff_t *ppos)
205  {
206          char *script = NULL;
207          int err_exec = 0;
208
209          if (size == 0)
210                  return 0;
211
212          script = (char *)kmalloc(size, GFP_KERNEL);
213          if (script == NULL)
214                  return -ENOMEM;
215
216          if (copy_from_user(script, buf, size) < 0)
217                  return -EIO;
218
219          spin_lock_bh(&lock);
220          luaU_setenv(L, NULL, struct nflua_ctx);
221
222          if (nflua_dostring(L, script, size) != 0) {
223                  pr_err("%s\n", lua_tostring(L, -1));
224                  lua_pop(L, 1); /* error */
225                  err_exec = -ENOEXEC;
226          }
227          spin_unlock_bh(&lock);
228
229          kfree(script);
230
231          return err_exec ? err_exec : size;
232  }
```

xt_lua.c

# Safe Browsing

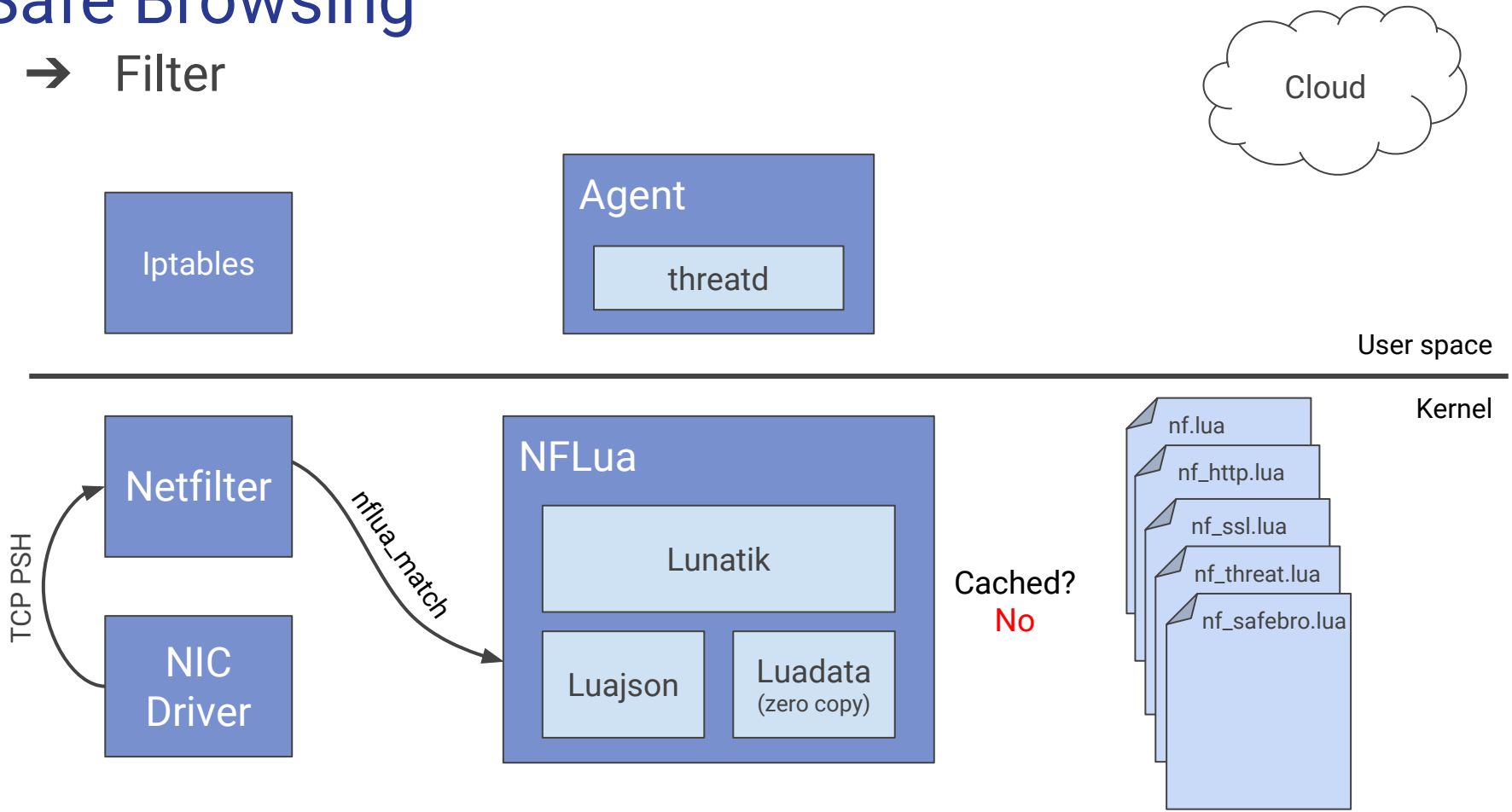
➔ Configuration

```
117  function safebro.config(settings)
118          local _conf = json.decode(settings)
119          conf.profiles = load_profiles(_conf.profiles or {})
120          conf.reputation = _conf.reputation or conf.reputation
121          conf.categories = load_list(_conf.categories or conf.categories)
122          conf.whitelist = load_list(_conf.whitelist)
123  end
```

nf_safebro.lua

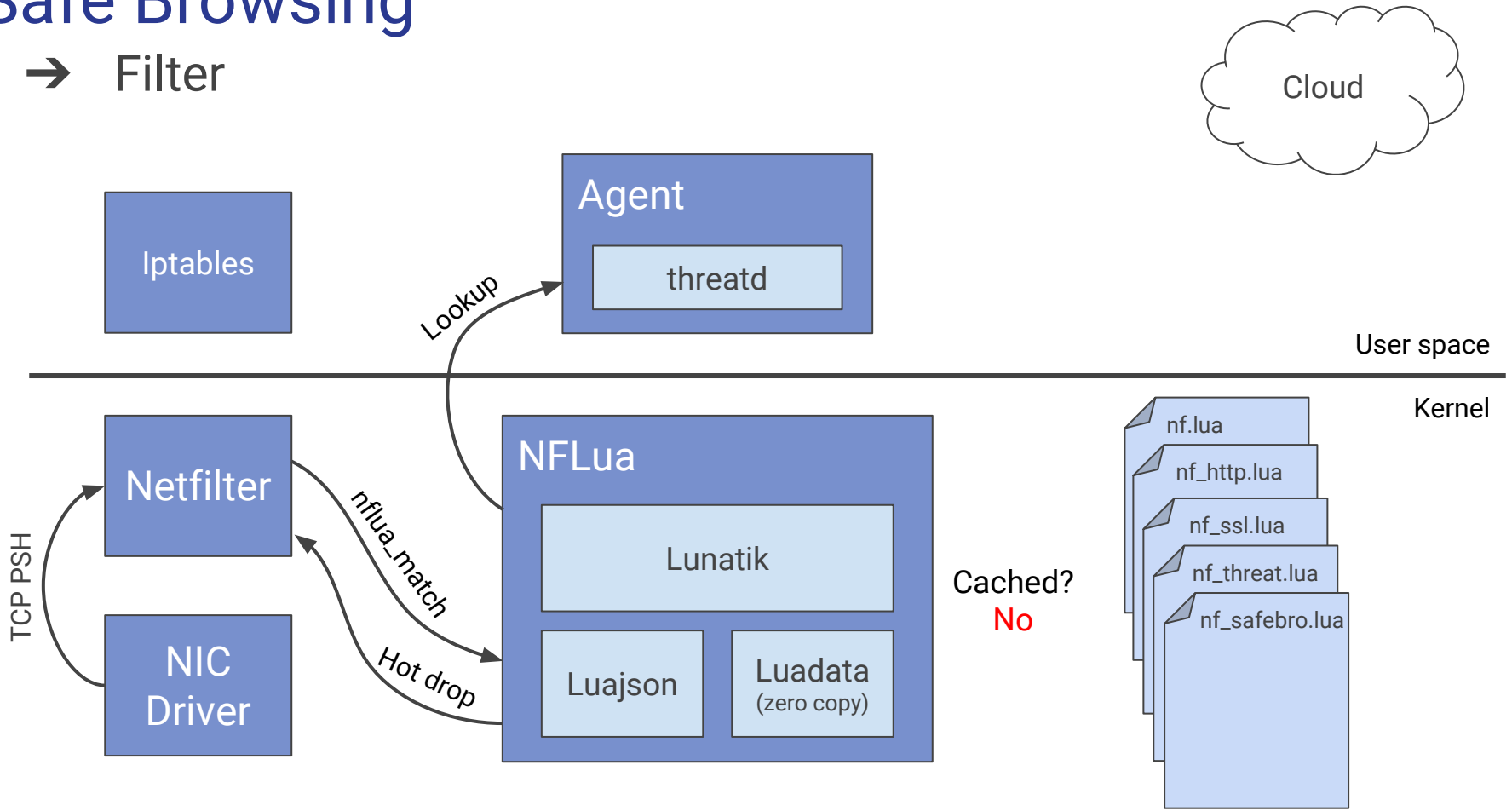
# Safe Browsing

➔ Filter



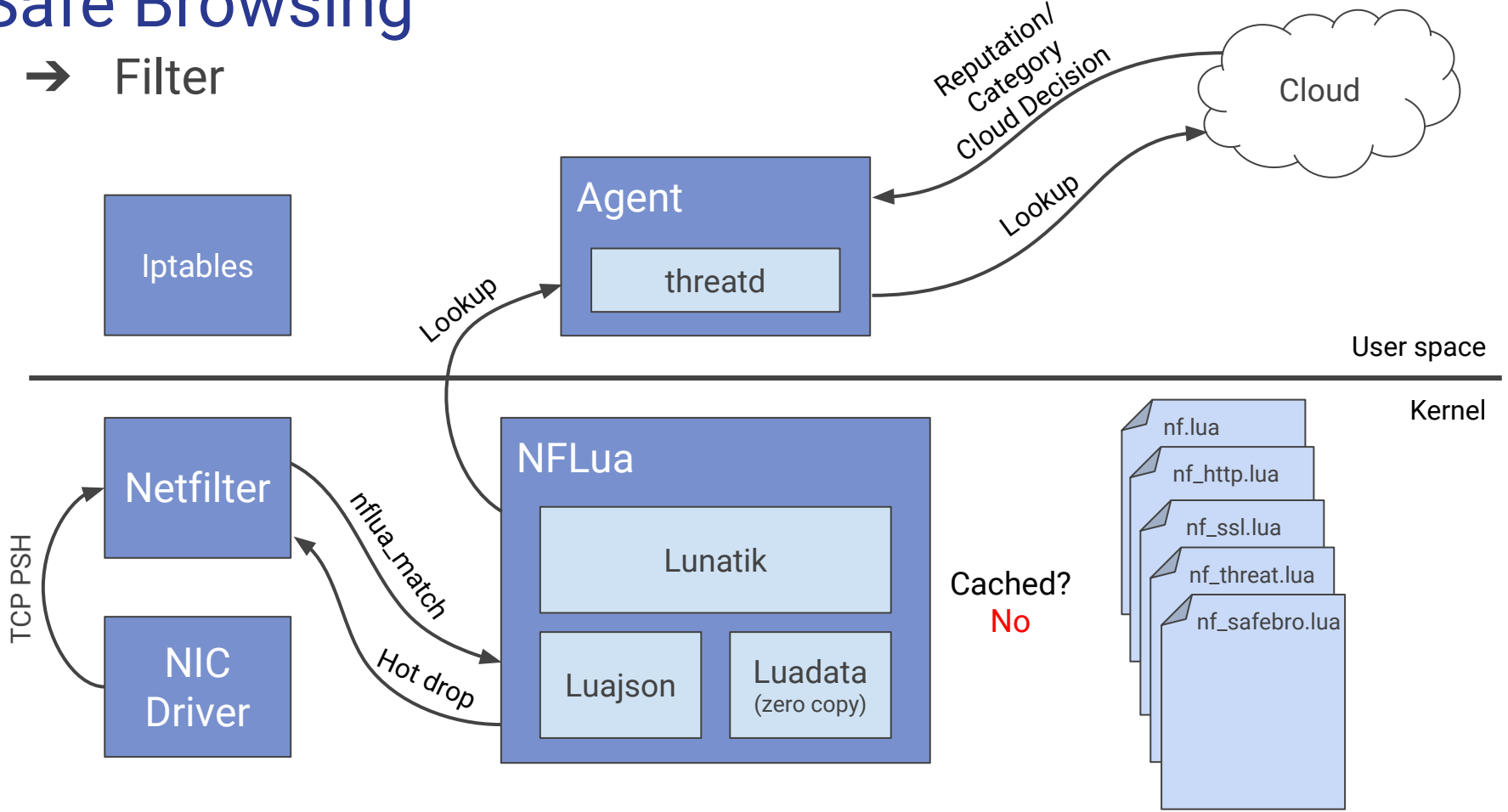
# Safe Browsing

➔ Filter



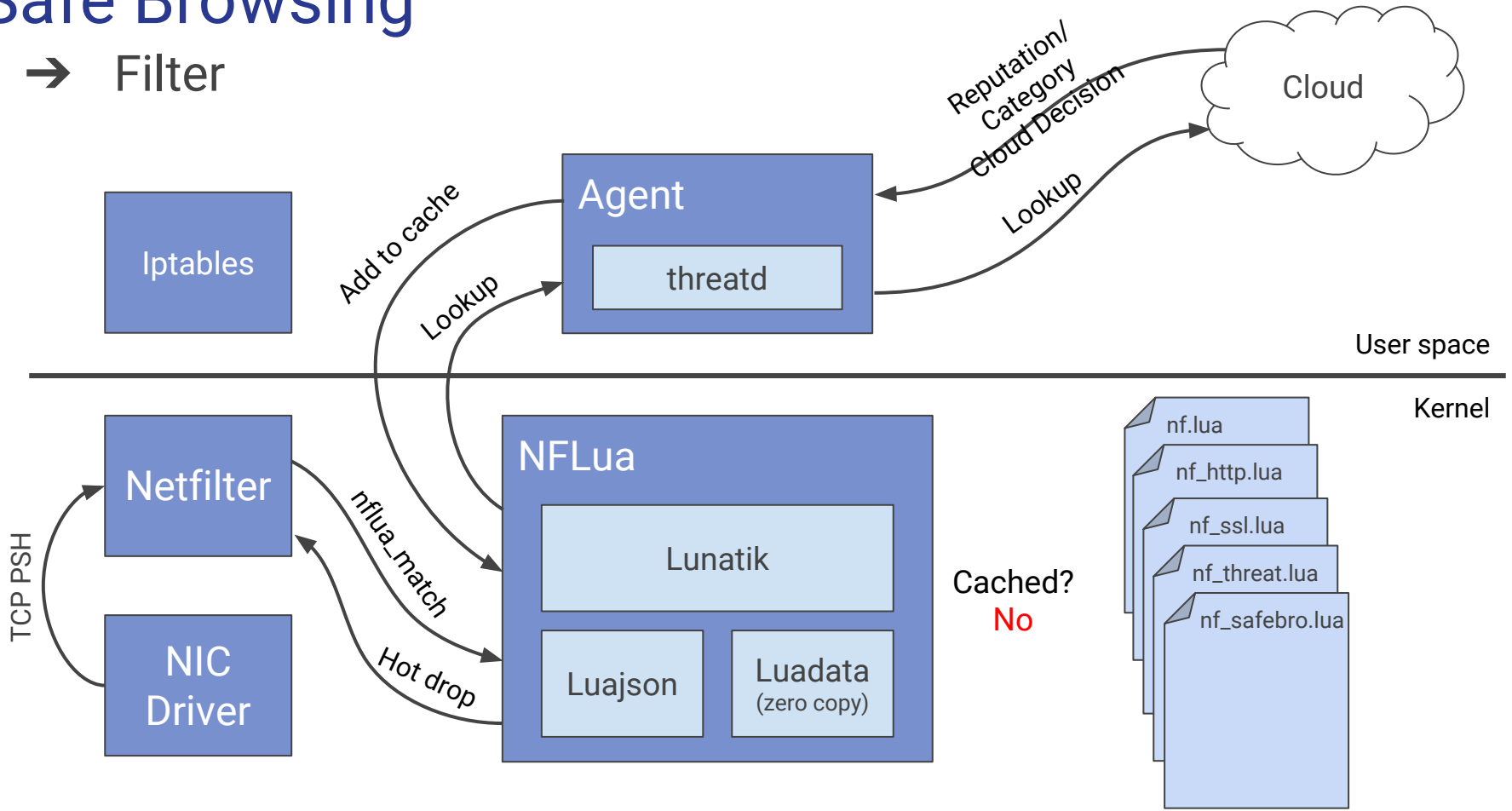
# Safe Browsing

➔ Filter



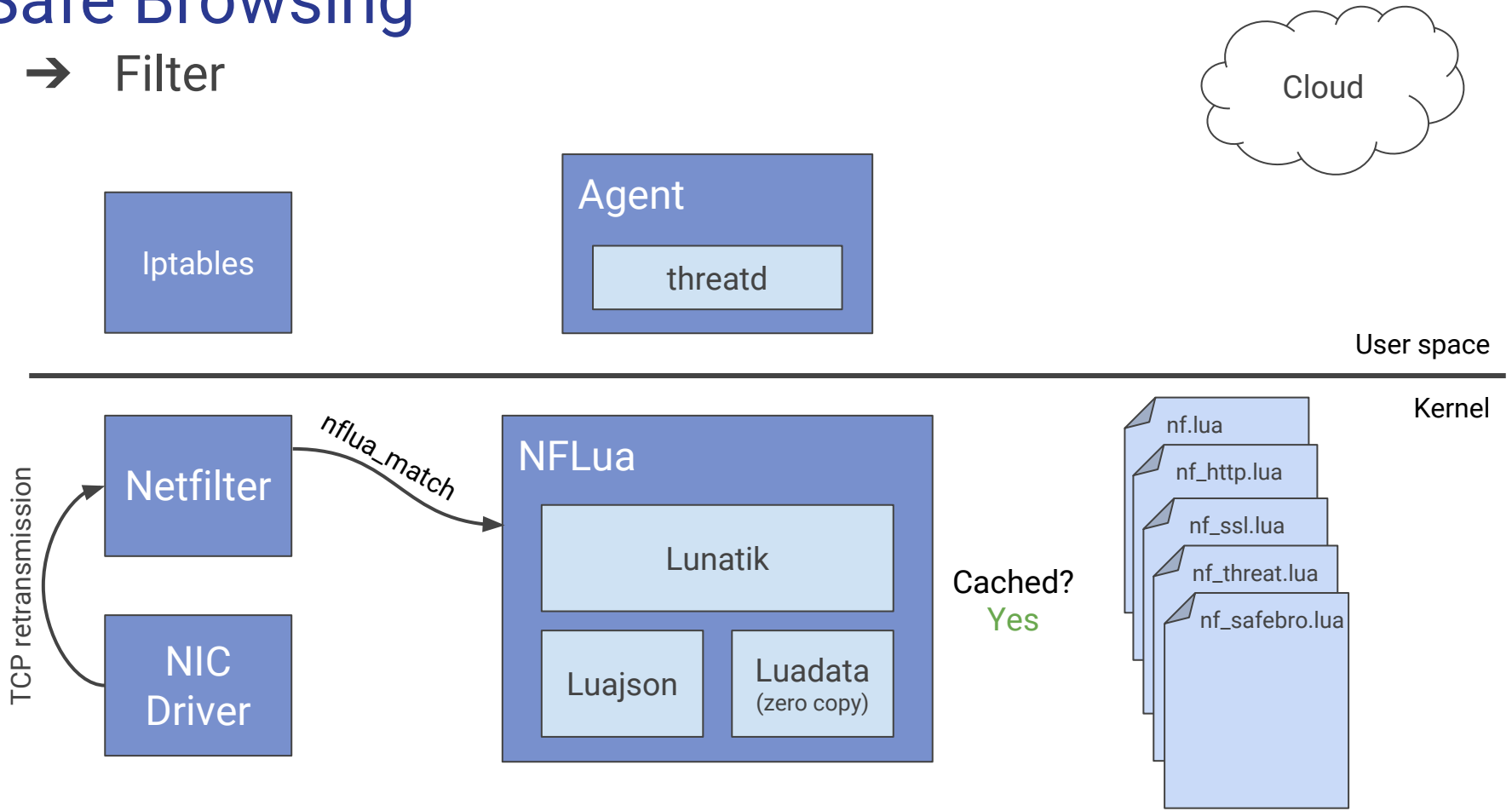
# Safe Browsing

➔ Filter



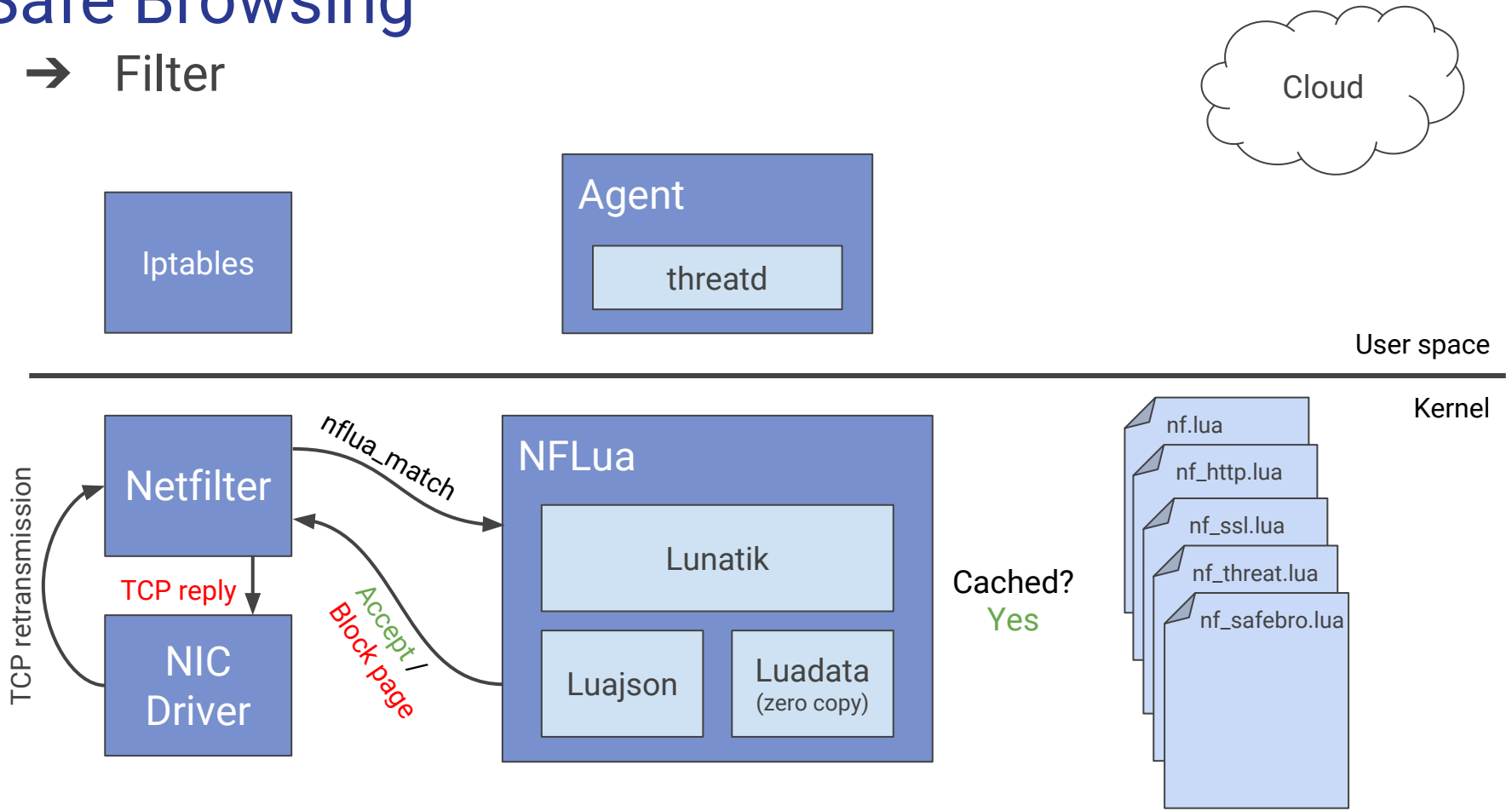
# Safe Browsing

➔ Filter



# Safe Browsing

➔ Filter



# Safe Browsing

➔ Filter

```
62  static bool nflua_match(const struct sk_buff *skb, struct xt_action_param *par)
63  {
64          const struct xt_lua_mtinfo *info = par->matchinfo;
65          struct nflua_ctx ctx = {.skb = skb, .par = par};
66          bool match = false;
67          int error  = 0;
68          int frame = LUA_NOREF;
69          int packet = LUA_NOREF;
70
71          spin_lock(&lock);
72          luaU_setenv(L, &ctx, struct nflua_ctx);
73
74          if (lua_getglobal(L, info->func) != LUA_TFUNCTION) {
75                  pr_err("%s: %s\n", "couldn't find match function", info->func);
76                  goto out;
77          }
78
79          frame = ldata_newref(L, skb_mac_header(skb), skb->mac_len);
80          packet = ldata_newref(L, skb->data, skb->len);
81
82          error = lua_pcall(L, 2, 1, 0);
83
84          ldata_unref(L, frame);
85          ldata_unref(L, packet);
86
87          if (error) {
88                  pr_err("%s\n", lua_tostring(L, -1));
89                  goto out;
90          }
91
92          par->hotdrop = (bool) lua_isnil(L, -1); /* cache miss? */
93          match = par->hotdrop ? false : (bool) lua_toboolean(L, -1);
94  out:
95          lua_pop(L, 1); /* result, info->func or error */
96          spin_unlock(&lock);
97          return match;
98  }
```

xt_lua.c

# Safe Browsing

➔ Filter

```
74          if (lua_getglobal(L, info->func) != LUA_TFUNCTION) {
75                  pr_err("%s: %s\n", "couldn't find match function", info->func);
76                  goto out;
77          }
78
79          frame = ldata_newref(L, skb_mac_header(skb), skb->mac_len);
80          packet = ldata_newref(L, skb->data, skb->len);
81
82          error = lua_pcall(L, 2, 1, 0);
83
84          ldata_unref(L, frame);
85          ldata_unref(L, packet);
86
87          if (error) {
88                  pr_err("%s\n", lua_tostring(L, -1));
89                  goto out;
90          }
91
92          par->hotdrop = (bool) lua_isnil(L, -1); /* cache miss? */
93          match = par->hotdrop ? false : (bool) lua_toboolean(L, -1);
```

xt_lua.c

# Safe Browsing

➔ Filter

```
78  function nf_http(frame, packet)
79          local mac = nf.mac(frame)
80          local ip = nf.ipv4(packet)
81          local tcp, payload = nf.tcp(ip)
82
83          if payload and not threat.bypass[mac.src] then
84                  local request = tostring(payload)
85                  local path, host = string.match(request, '[A-Z]+ (%g+).*Host: (%g+)')
86
87                  if host then
88                          local uri = host .. path
89                          local urikey = threat.key(mac, uri)
90                          local hostkey = threat.key(mac, host)
91
92                          return not whitelist(hostkey, urikey)
93                                  and not unblock(host, hostkey, uri, urikey, request)
94                                  and block(mac, ip, tcp, host, uri) -- DROP
95                  end
96          end
97
98          return false -- ALLOW
99  end
```

nf_http.lua

# Safe Browsing

➔ Filter

```
22  function nf.ipv4(packet)
23          local layout = data.layout{
24                  version  = {   0,   4},
25                  ihl      = {   4,   4},
26                  tos      = {   8,   6},
27                  ecn      = {  14,   2},
28                  tot_len  = {  16,  16, 'net'},
29                  id       = {  32,  16, 'net'},
30                  flags    = {  48,   3},
31                  frag_off = {  51,  13, 'net'},
32                  ttl      = {  64,   8},
33                  protocol = {  72,   8},
34                  check    = {  86,  16, 'net'},
35                  src      = {  96,  32, 'net'},
36                  dst      = { 128,  32, 'net'},
37          }
38          return segment(packet, layout)
39  end
```

nf.lua

# Safe Browsing

➔ Filter

```
63  function nf_ssl(frame, packet)
64          local mac = nf.mac(frame)
65          local ip = nf.ipv4(packet)
66          local tcp, payload = nf.tcp(ip, ssl)
67
68          if threat.bypass[mac.src] then return false end
69
70          local host = is_client_hello(payload) and extract_hostname(payload)
71          return host and not threat.whitelist[threat.key(mac, host)] and
72                  safebro.filter(mac.src, ip.src, host)
73  end
```

nf_ssl.lua

# Safe Browsing

➔ Filter

```
31  local function extract_hostname(payload)
32          local ssl_info = payload:segment(43)
33
34          ssl_info:layout{sid = BYTE}
35          ssl_info:layout{sid = BYTE, cis = {8 + (ssl_info.sid * 8), 16, 'net'}}
36
37          if not ssl_info.sid or not ssl_info.cis then return false end
38          local noise_len = 50 + ssl_info.sid + ssl_info.cis
39
40          local extension = payload:segment(noise_len)
41          if not extension then return false end
42
43          repeat
44                  extension:layout(server_name)
45
46                  if extension.id == 0 and extension.ext_id == 0 then
47                          local len = extension.ext_len - 3
48                          local hostname = extension:segment(9, len)
49                          return tostring(hostname)
50                  end
51
52                  extension = extension:segment(extension.len + 4)
53          until not extension
54
55          return false
56  end
```

nf_ssl.lua

# Safe Browsing

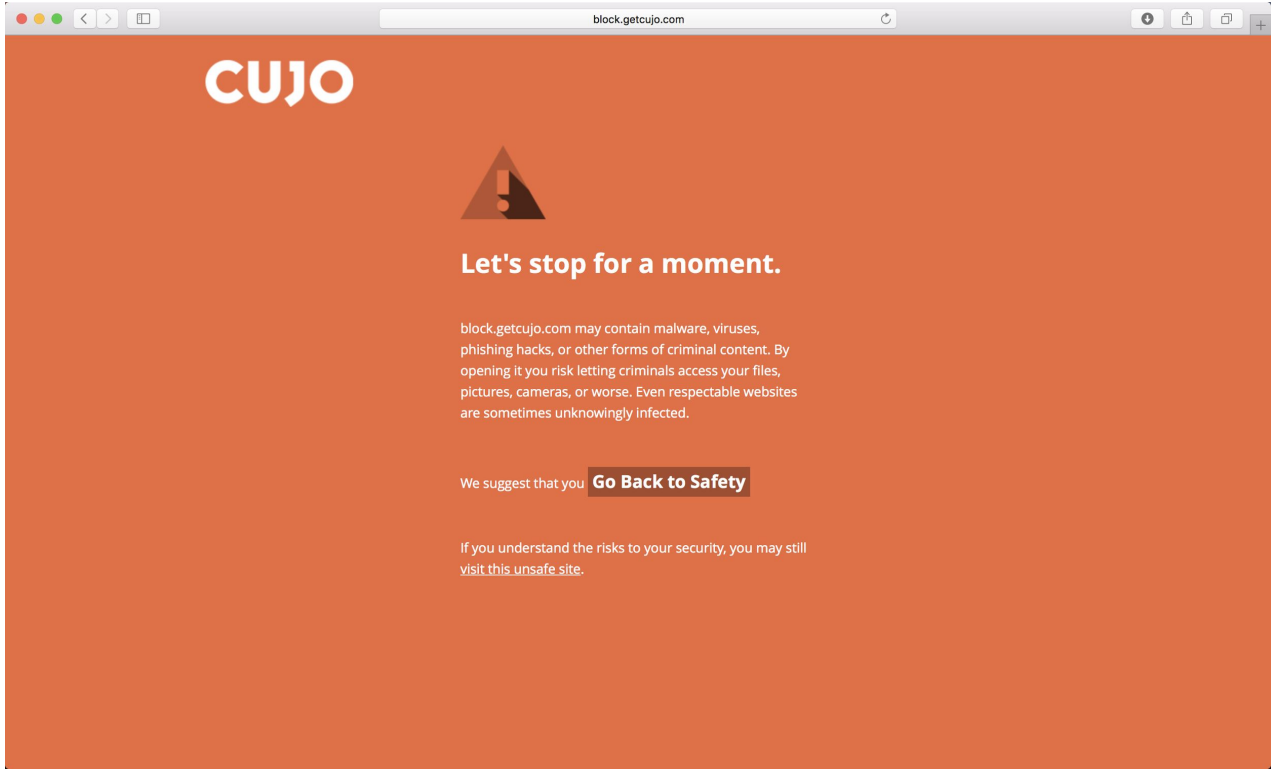
➔ Filter

```
39  local function block(mac, ip, tcp, host, uri)
40          local block, reason = safebro.filter(mac.src, ip.src, host, uri)
41
42          if block then
43                  local salt
44                  local page = blockpage
45
46                  if reason ~= 'access' then
47                          local uri = reason == 'db' and uri or host
48
49                          page = warnpage
50                          salt = math.random()
51                          blocked[threat.key(mac, uri)] = salt
52                  end
53
54                  nf.reply('tcp', response(page, uri, salt))
55                  finished[source(ip, tcp)] = true
56          end
57
58          return block
59  end
```

nf_http.lua

# Safe Browsing

➔ Filter




Block page

# Why Lua?

- ➔ Extensible Extension Language
  - ◆ Embeddable and Extensible
  - ◆ C Library
- ➔ Almost Freestanding
- ➔ Small Footprint
  - ◆ ~250 KB
- ➔ Fast
- ➔ MIT License

# Why Lua?

- ➔ Ease of Development
  - ➔ High-level Language
  - ➔ Dynamically Typed
  - ➔ Domain-specific API

# Why Lua?

- ➔ Safety
  - ➔ Automatic Memory Management
  - ➔ Protected Call
  - ➔ Fully Isolated States
  - ➔ Cap the Number of Executed Instructions
  - ➔ Test Suite

# Why Lua?

## ➔ Security

- A single vulnerability disclosed since 1993

**CVE Details**

*The ultimate security vulnerability datasource*

(e.g.: CVE-2009-1234 or 2010-1234 or 20101234)

Log In Register

Vulnerability Feeds & Widge

Switch to https://
Home
**Browse :**
Vendors
Products
Vulnerabilities By Date
Vulnerabilities By Type
**Reports :**
CVSS Score Report
CVSS Score Distribution
**Search :**
Vendor Search
Product Search
Version Search
Vulnerability Search
By Microsoft References

## LUA : Vulnerability Statistics

Products (1) Vulnerabilities (1) Search for products of LUA CVSS Scores Report Possible matches for this vendor Related Metasploit Modules
Vulnerability Feeds & Widgets

### Vulnerability Trends Over Time

| Year | # of Vulnerabilities | DoS | Code Execution | Overflow | Memory Corruption | Sql Injection | XSS | Directory Traversal | Http Response Splitting | Bypass something | Gain Information | Gain Privileges | CSRF | File Inclusion | # of exploits |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2014 | 1 | 1 | | 1 | | | | | | | | | | | |
| Total | 1 | 1 | | 1 | | | | | | | | | | | |
| % Of All | | 100.0 | 0.0 | 100.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | |

Warning : Vulnerabilities with publish dates before 1999 are not included in this table and chart. (Because there are not many of them and they make the page look bad; and they may not be years.)

# Benchmarks

- Tinyproxy
  - ~150 Mbps
  - CPU Bound
- NFLua
  - Slow Path: ~500 Mbps
  - Fast Path: ~750 Mbps
  - Not CPU Bound
- Bypass
  - ~890 Mbps
- Online Units: ~5.5 k