ZeekJS: JavaScript support in Zeek

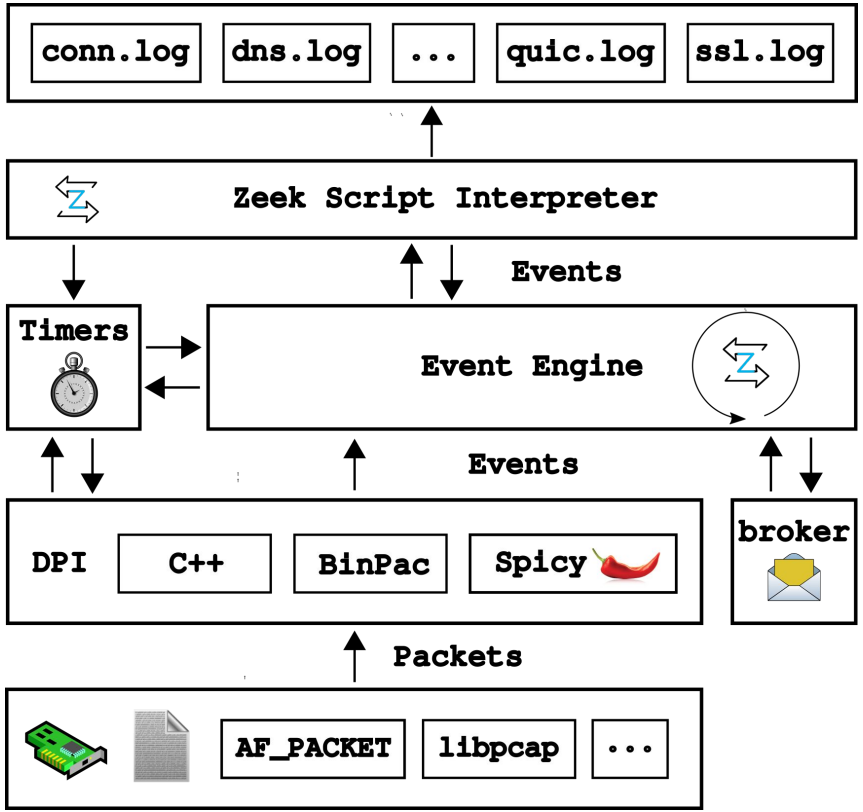
Arne Welzel, FOSDEM 2024

What is Zeek?

Network Security Monitor

- 1995
- Open-source (BSD)
- Bro until 2018

- Passive
- Protocol Logs
- Extensible
- Scriptable

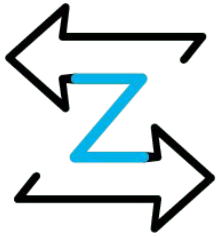


Example Logs

```
# quic.log (since Zeek 6.1)
{
  "ts": 1692198832.625294,
  "uid": "Cv5q4e3YZwQqbxfXD5",
  "id.orig_h": "82.239.54.117",
  "id.orig_p": 44174,
  "id.resp_h": "250.58.23.113",
  "id.resp_p": 443,
  "version": "1",
  "client_initial_dcid": "c5a5015ae8f479784a",
  "client_scid": "34696c",
  "server_scid": "01275b138ee...47cf7773f",
  "server_name": "blog.cloudflare.com",
  "client_protocol": "h3",
  "history": "ISiishIhhhHHhHH"
}
```

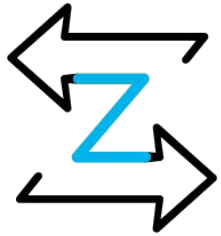
```
# conn.log
{
  "ts": 1692198832.625294,
  "uid": "Cv5q4e3YZwQqbxfXD5",
  "id.orig_h": "82.239.54.117",
  "id.orig_p": 44174,
  "id.resp_h": "250.58.23.113",
  "id.resp_p": 443,
  "proto": "udp",
  "service": "ssl,quic",
  "duration": 1.2880840301513672,
  "orig_bytes": 30627,
  "resp_bytes": 502113,
  "conn_state": "SF",
  "local_orig": false,
  "local_resp": true,
  "missed_bytes": 0,
  "history": "Dd",
  "orig_pkts": 134,
  "orig_ip_bytes": 34379,
  "resp_pkts": 474,
  "resp_ip_bytes": 515385
}
```

Zeek Scripting Language



```
event QUIC::initial_packet(c: connection, is_orig: bool, version: count,
                           dcid: string, scid: string) {
        c$quic = Info(
                $ts=network_time(),
                $uid=c$uid,
                $id=c$id,
                $version=version_strings[version]);

        Conn::register_removal_hook(c, finalize_quic);
        ...
}

event ssl_extension_server_name(c: connection, is_client: bool,
                                names: string_vec) &priority=5 {
        if ( is_client && c?$quic && |names| > 0 )
                c$quic$server_name = names[0];
}

hook finalize_quic(c: connection) {
        Log::write(c$quic);
}
```

Zeek Scripting Language



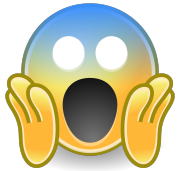
```
event QUIC::initial_packet(c: connection, is_orig: bool, version: count,
                           dcid: string, scid: string) {
        c$quic = Info(
                $ts=network_time(),
                $uid=c$uid,
                $id=c$id,
                $version=ver

        Conn::register_remov
        ...
}

event ssl_extension_server_n

        if ( is_client && c?
                c$quic$serve
}

hook finalize_quic(c: connection) {
        Log::write(c$quic);
}
```

Zeek's asynchronous ActiveHTTP module

Spawn an input reader thread executing roughly:

```
system(fmt("curl -o %s %s", "/tmp/1234_body", url))
```

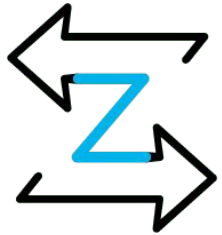
Read /tmp/1234_body and return to caller



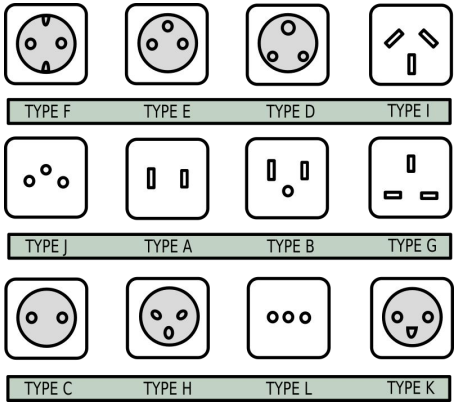
Adding JavaScript to Zeek…

…as a plugin!

Zeek Plugins



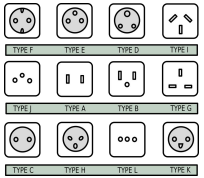
- Shared libraries (.so, .dylib)
- Can access Zeek's C++ API
- Can *hook* into Zeek's execution
  - InitPreScript(), InitPostScript(), Done()
  - HookLoadFile(), HookCallFunction(), HookDrainEvents(), HookSetupAnalyzerTree()
- Can add Components to Zeek
  - Log writers (Kafka, …)
  - Packet sources (PF_RING, Napatech, …)
  - Protocol, packet and file analyzers
  - Opaque script types
  - IO sources



What it might look like…


```
// quic-http.js
const http = require("node:http");

zeek.on("QUIC::initial_packet", (c, is_orig, version, dcid, scid) => {
  console.log(`QUIC::initial_packet: ${c.uid} dcid=${hexlify(dcid)}`);
});

zeek.on("ssl_extension_server_name", (c, is_client, names) => {
  console.log(`ssl_extension_server_name: ${c.uid} ${names[0]}`);

  let req = http.request("http://localhost:8080/server_names",
                         {method: "POST"});
  req.write(JSON.stringify({
    uid: c.uid,
    names: names,
  }));

  req.end();
});
```

Step 1: Intercept loading of .js files

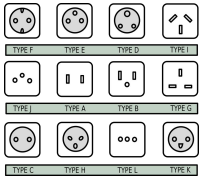


- *HookLoadFile()*

```
$ zeek ./quic-ssl.js

# main.zeek
@load ./quic-ssl.js
```

```
int Plugin::HookLoadFile(const zeek::plugin::Plugin::LoadType,
                         const std::string& file,
                         const std::string& resolved) {
  if (file.find(".js", file.size() - 3) != std::string::npos) {
    load_files.emplace_back(resolved);
    return 1;
  }

  return 0;
}
```

Step 2: Initialize V8 / Node.js environment

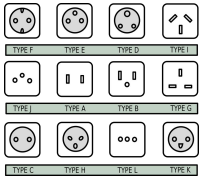


- *InitPostScript()*
- zeek.on() calls are executed!

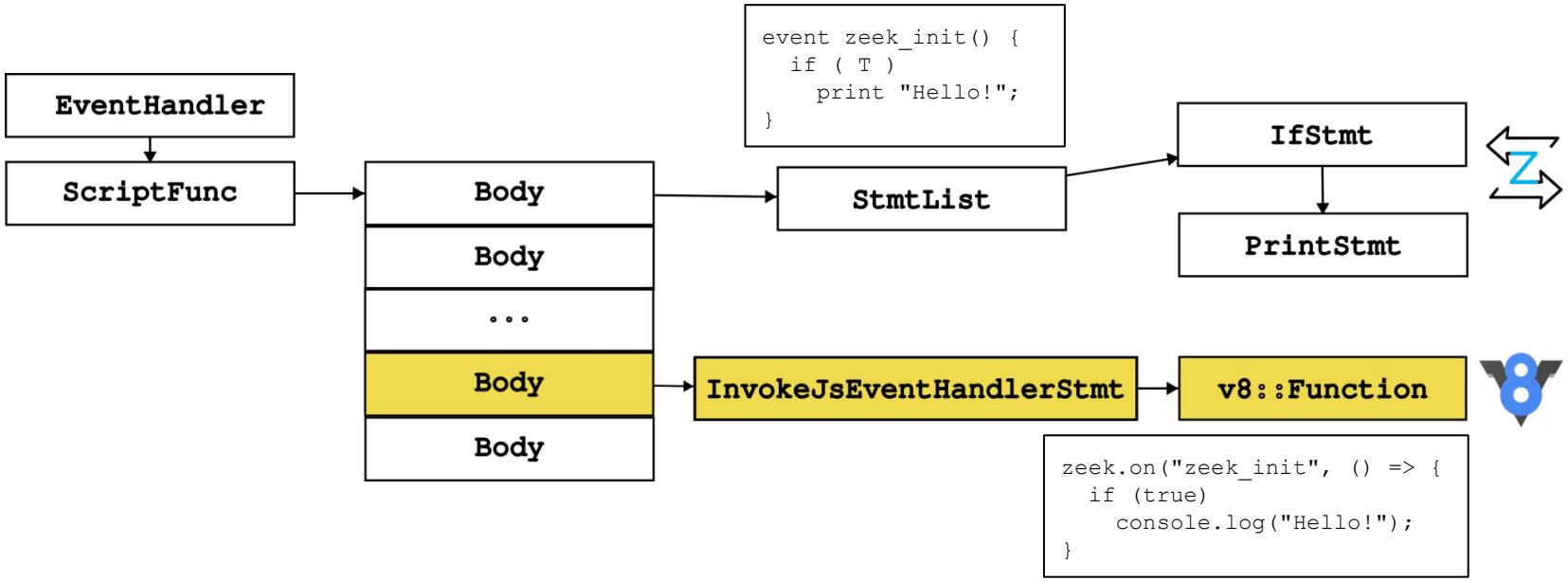
```
node::InitializeOncePerProcess(...);
node::MultiIsolatePlatform::Create(...);
v8::V8::InitializePlatform(...);
v8::V8::Initialize();
node::CreateEnvironment(...);

# main_script_source actually imports all .js files
node::LoadEnvironment(env_.get(), main_script_source.c_str());
```

https://nodejs.org/api/embedding.html

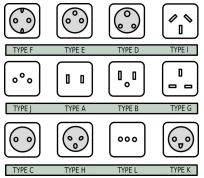
Step 3: Implement zeek.on() callback



- Install JavaScript functions as Zeek event handlers

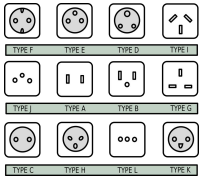


Step 4: Zeek to JavaScript Type Conversions

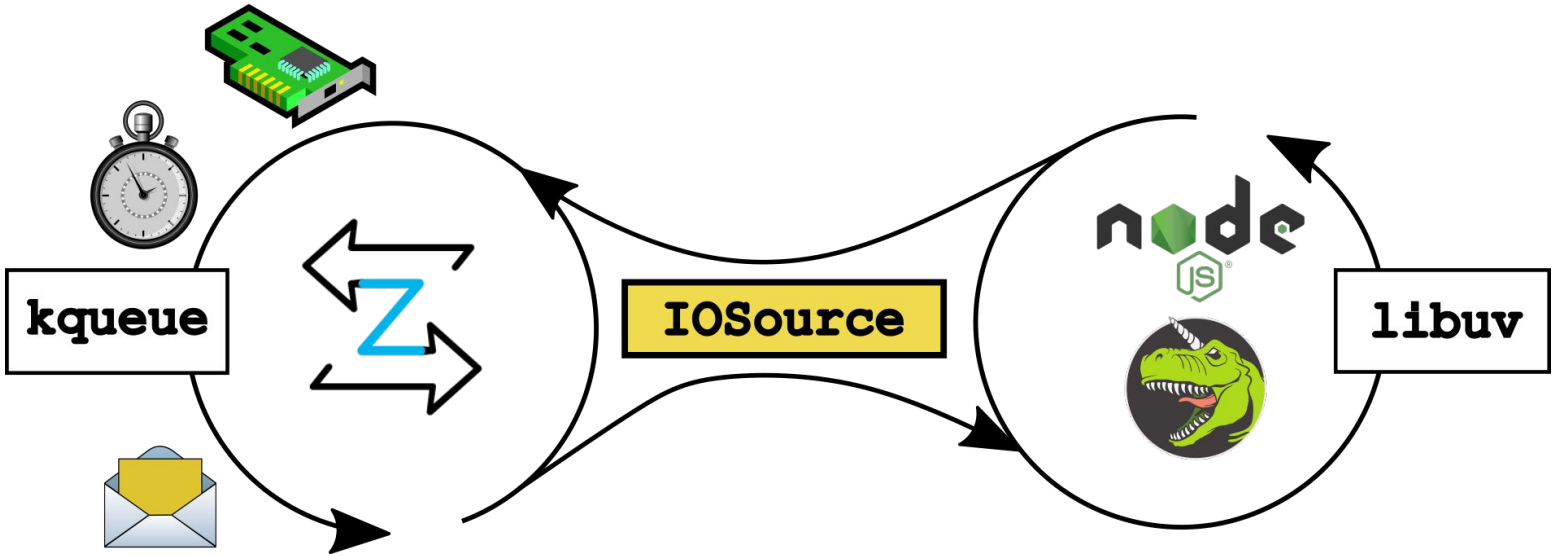


string	string
addr and subnet	string
time and interval	number
count	bigint
record	object wrapping Zeek record value
…	…

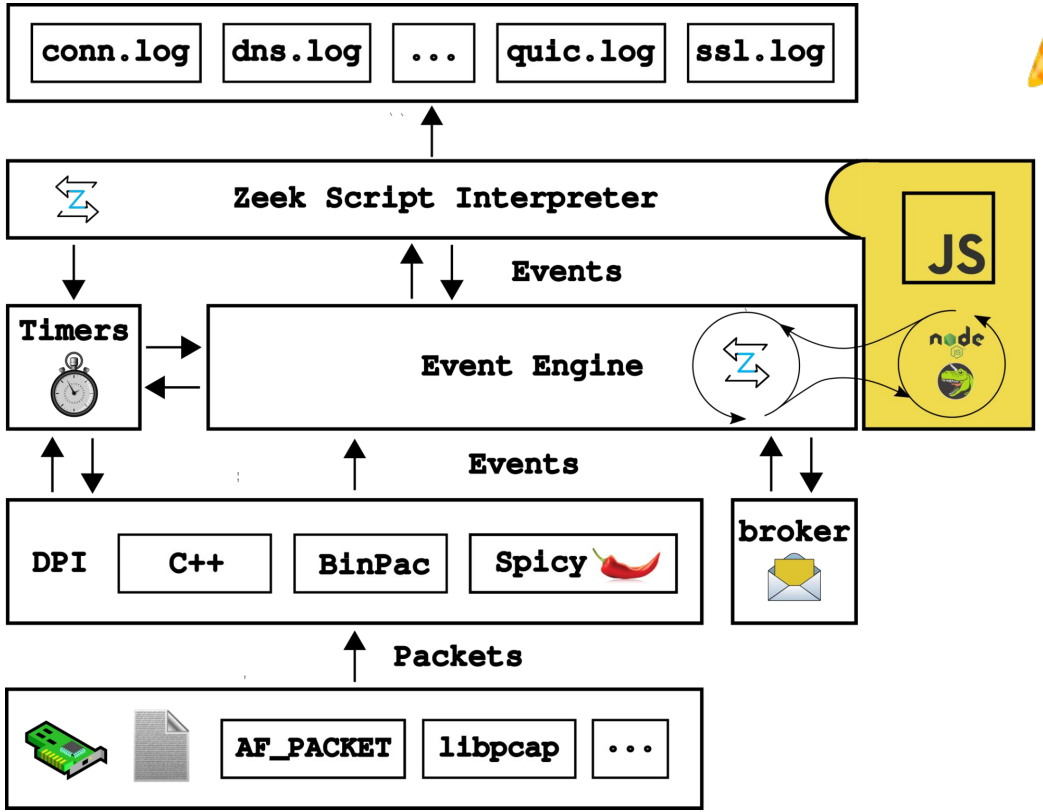
Step 5: IO Loop Integration



- The libuv event loop is registered as just another IO source
- RegisterFd(uv_backend_fd(&loop), plugin)



ZeekJS


conn.log
dns.log
...
quic.log
ssl.log
Zeek Script Interpreter
JS
Events
Timers
Event Engine
node
Events
broker
DPI
C++
BinPac
Spicy
Packets
AF_PACKET
libpcap
...

Summary, Outlook and Questions

- Zeek is extensible enough to add a whole scripting language
- JavaScript is not going to replace the Zeek Scripting Language
- Prototyping of integrations without C++. But, anything, really!
- ZeekJS is part of Zeek 6.0

https://github.com/corelight/zeekjs

https://github.com/zeek/zeek