HAKA

UN LANGAGE ORIENTÉ RÉSEAUX ET SÉCURITÉ

Arkoon, OpenWide et Telecom ParisTech

CONTEXTE

- Attaques réseaux de plus en plus complexes
- Développements en C long et coûteux
- Besoin d'analyses jusqu'au pseudo niveau 8

PROJET

- Définition d'un langage
  - Pour des règles de sécurité
  - Pour de la dissection protocolaire
- Définition d'une API

TELECOM ParisTech

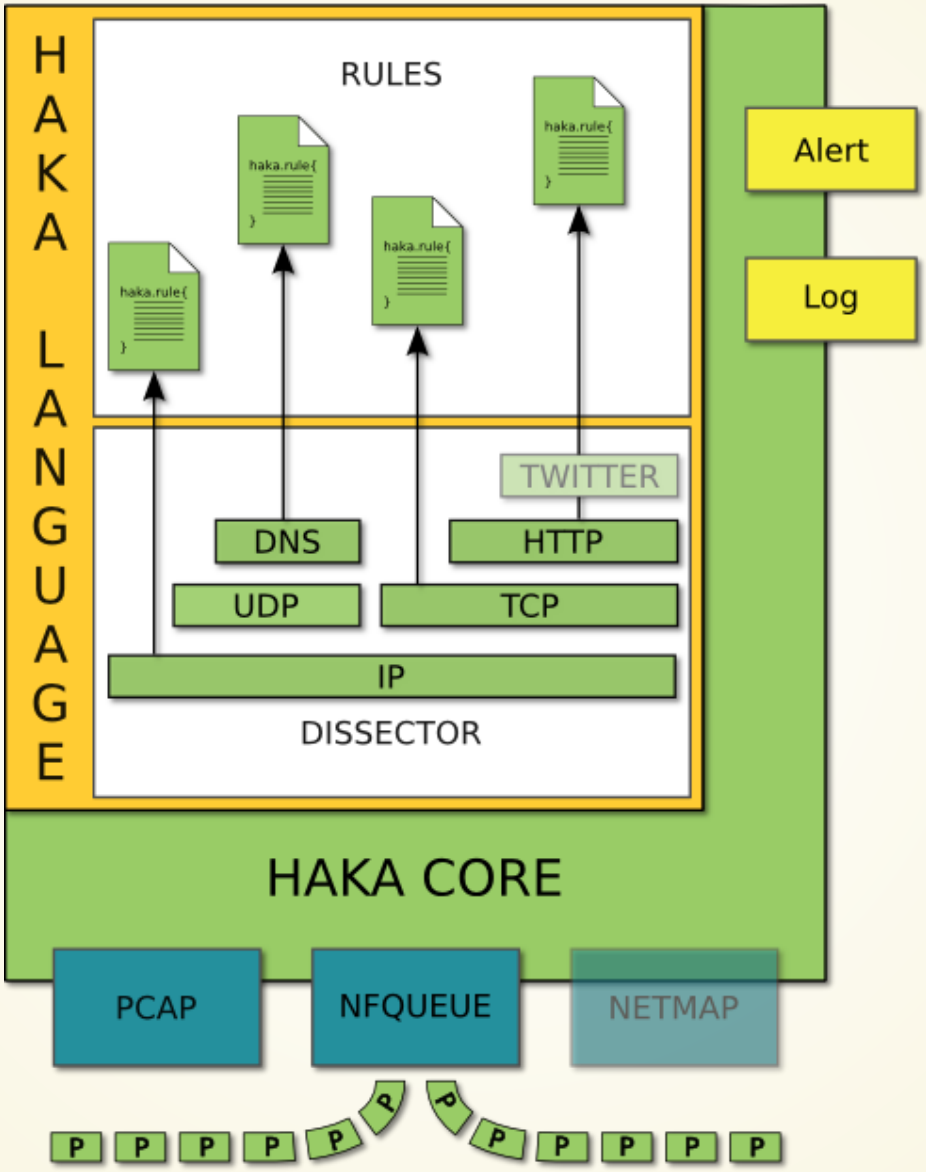
open wide
ARCHITECTE OPEN SOURCE

ARKOON + NETASQ

Fond national pour la Société Numérique

ARCHITECTURE

- Un core en C
- Une construction modulaire
- Des extensions Lua


HAKA LANGUAGE
RULES
haka.rule{
}
DNS
UDP
TWITTER
HTTP
TCP
IP
DISSECTOR
Alert
Log
HAKA CORE
PCAP
NFQUEUE
NETMAP
P

RÈGLES DE SÉCURITÉ

- Filtrage de flux
- Alertes
- Modification de flux à la volée
- Injection de paquets

RÈGLES DE SÉCURITÉ

```
local http = require('protocol/http')

haka.rule {
    hook = http.events.request,
    eval = function(http, request)
        request.headers["Haka"] = "Modified by Haka"
        if request.method == "HEAD" then
            haka.alert{
                description = string.format("HEAD method from %s",
                    http.flow.srcip);
                severity = 'low'
            }
            http:drop()
        end
    end
}
```

STREAM

```
local rem = require('regexp/pcre')
local re = rem.re:compile('<script.*</script>')

haka.rule {
    hook = http.events.response_data,
    options = { streamed = true },
    eval = function (http, iter)
        local match
        repeat
            match = re:match(iter)
            if match then
                match:erase()
            end
        until not match
    end
}
```

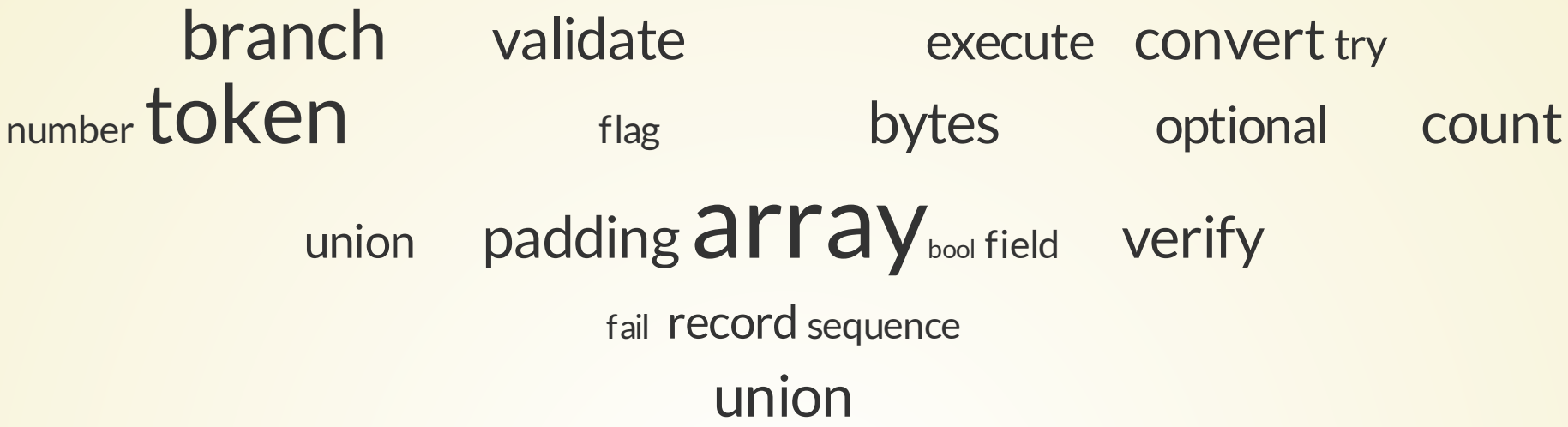
DISSECTION

- Analyse du protocole
- Extraction des champs
- Définition d'évènements

GRAMMAIRE

```
icmp_dissector.grammar = haka.grammar.new("icmp", function ()
    packet = record{
        field('type',     number(8)),
        field('code',     number(8)),
        field('checksum', number(16))
            :validate(function (self)
                self.checksum = 0
                self.checksum = ipv4.inet_checksum_compute(self._payload)
            end),
        field('payload',  bytes())
    }

    export(packet)
end)
```


branch
validate
execute
convert
try
number
token
flag
bytes
optional
count
union
padding
array
bool
field
verify
fail
record
sequence
union

MACHINE À ÉTATS

- Définition d'états
- Création des transitions entre ces états

MACHINE À ÉTATS

```
response = state()
connect = state()

response:on{
    event = events.down,
    check = function (self, response)
        return self.request.method:lower() == 'connect'
    end,
    jump = connect
}
```

RÉSULTATS

- Publication open source
  - v0.1: règles de sécurité
  - v0.2: orientée écriture de dissecteurs
- Perspectives
  - v0.3 axée sur les performances
  - Animation de la communauté

http://haka-security.com

@hakasecurity

github.com/haka-security/haka

QUESTIONS?