Chimera

A Declarative Language for Streaming Network Traffic Analysis

Kevin Borders*, Jonathan Springer†, Matthew Burnside*

*National Security Agency, †Reservoir Labs

Motivation

Pattern-Based Signatures

`/^evil$/`

Behavioral Analytics

`Session Cookie from > 1`
`Client IP/User-Agent`

("side-jacking")

Problem: Semantic Gap

Session Cookie from > 1 Client IP/ User-Agent

Alice

Chimera

```
~50 lines of code

…
event http_all_headers(c: connection,
  is_orig: bool, hlist:mime_header_list)
{
  if (! is_orig)
    return;
  if (! c$http?$cookie ||
      c$http$cookie == "" )
    return;

  local cookie = "";
  local service = c$http$host;

…
~50 more lines of code
```

Bro

Chimera Goals

1. Concise
2. Declarative
3. Interoperable
4. Efficient

Start From SQL

Beyond SQL:
Structured Data (Lists and Maps)

- `SPLIT` Operator
  - One row for each item
- Iteration + First class functions, e.g.:
  - `<list>.foreach{<expr>}`
    - Replace each item with `<expr>`
  - `<list>.find{<expr>}`
    - Return first item where `<expr> == true`

Beyond SQL:
Dynamic Window Boundaries

- Steaming SQL:
  - `FROM x RANGE 60 MINUTES, SLIDE 1 MINUTE`
- Chimera:
  - `GROUP BY … UNTIL <expr>`
  - `FROM <a> JOIN <b> … WINDOW <expr>`

+ Increased flexibility

+ Immediate results

Chimera in Action

Scenario 1: Side-jacking

Description

- Session ID in HTTP ‘Cookie’ header used by more than one IP/User-Agent client

Chimera Query

```
SELECT
 list_agg(distinct(concat(
    [packets].[srcip], ':',
    [headers].[User-Agent])))
    AS clientlist
 [headers].[Cookie].split(';').
    foreach{$.split('=')}.
    find{$.first() == 'SID'}.last()
    AS sessionid
FROM http
WHERE [sessionid] != NULL
GROUP BY [sessionid]
UNTIL [clientlist].size() > 1
```

Chimera in Action

Scenario 2: DNS TTL Value Changes

Description

- Count the number of changes in DNS TTL value per domain per day

  (From EXPOSURE paper)

Chimera Query

```
SELECT count(), [name]
FROM (
 SELECT [name]
 FROM dns_rr
 WHERE [ttl] != NULL
 GROUP BY [name]
 UNTIL
  last([ttl]) != last([ttl], 2,
   true) &&
  last([ttl], 2, true) != NULL
)
GROUP BY [name]
UNTIL GLOBAL nextwindow(
   [packets].[time], 86400)
```

Chimera in Action

Scenario 3: DNS Tunnel Detection

Description

- Count DNS A-record replies
- Where no client connects to the A-record IP
- Aggregate by DNS client IP
- Thresholds:
  - 100 replies
  - Unanswered for 5 minutes
  - In < 1 Hour

Chimera Query

```
SELECT
 [dns].[packets].[dstip] AS client,
 first([dns].[packets].[time]) AS
   start
 last([dns].[packets].[time]) AS end,
FROM dns EXCLUSIVE LEFT SINGLE JOIN
 ip_packet
 ON [answers].[aip] EQUALS [dstip]
WINDOW [new].[packets].[time] -
 [old].[packets].[time] < 300
WHERE [dns].[answers].[aip] != NULL
GROUP BY [client]
UNTIL count() > 100
HAVING [end] - [start] < 3600
```

The Compiler

Chimera Language

1. Parsing

Abstract Syntax Tree

2. Translation

Chimera Core (Rel. Algebra)

3. Code Generation

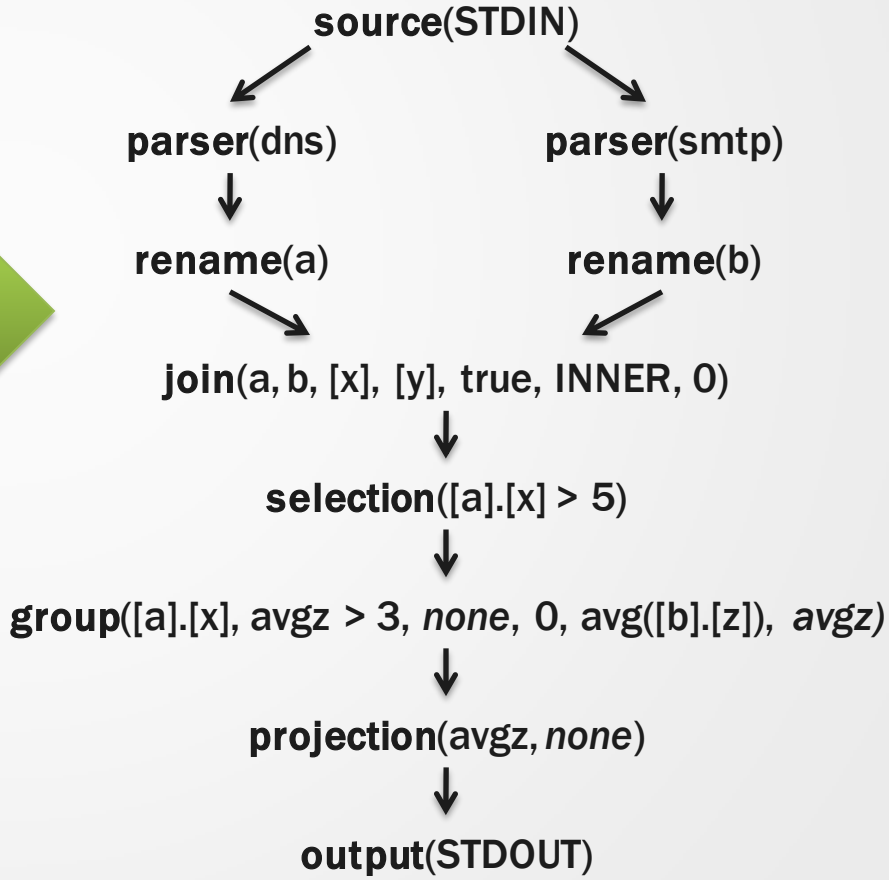
Bro Event Code

Translation

Chimera Query

```
SOURCE STDIN
SELECT avg([b].[z])
  AS avgz
FROM dns AS a
  JOIN
  smtp AS b
  ON [x] EQ [y]
WHERE [a].[x] > 5
GROUP BY [a].[x]
UNTIL avgz > 3
INTO STDOUT
```

Data Flow Graph



Code Generation

Chimera Query

```
SELECT [path]
FROM http-request
WHERE [method] == "GET"
```

Data-flow Graph

l0: **source**(STDIN)

l1: **parser**(http-request)

l2: **selection**([method] == “GET")

l3: **projection**([path], none)

l4: **output**()

Type Definitions

```
@http-reply
type l1_type:
    record {
  method: string;
  path: string;
  headers: listmap;
  body: string;
  packetlist:
    packetlist_type;
};
type l3_type: record {
  v1: string;
};
```

Event Handlers

```
# 14
event l3(t: l3_type) {
  print t$v1;
}
# 13
event l2(t: l1_type) {
  local out: l3_type;
  out$v1 = t$path;
  event l3(t);
}
# 12
event l1(t: l1_type) {
  if (!(t$method == "GET"))
    return;
  event l2(t);
}
# 11
event http_message_done(
    c:connection, ...) {
  local t = http_trans(c);
  event l1(t);
}
```

Is It Fast Enough?

Type Definitions

```
@http-reply
type l1_type:
    record {
  method: string;
  path: string;
  headers: listmap;
  body: string;
  packetlist:
    packetlist_type;
};
type l3_type: record {
  v1: string;
};
```

Event Handlers

```
# 14
event l3(t: l3_type) {
  print t$v1;
}
# 13
event l2(t: l1_type) {
  local out: l3_type;
  out$v1 = t$path;
  event l3(t);
}
# 12
event l1(t: l1_type) {
  if (!(t$method == "GET"))
    return;
  event l2(t);
}
# 11
event http_message_done(
    c:connection, ...) {
  local t = http_trans(c);
  event l1(t);
}
```

Hand-Written

```
@http-reply
event http_request(
    c: connection,
    method: string,
    original_URI: string,
    unescaped_URI: string,
    version: string) {
  if (method == "GET")
    print original_URI;
}
```

1. No data copying
2. Inline evaluation
3. Earlier event type

Optimization Impact

1 = No data copying, 2 = Inline evaluation, 3 = Earlier event type

Configuration	Base	Opt-1	Opt-1+2	Opt-1+2+3
Avg. Time(s)	14.21	14.00	14.01	13.79
Std. dev (s)	0.084	0.083	0.074	0.081
Sped-up(%)	n/a	1.5%	1.4%	3.0%

Is there room for optimization? Yes

Is it significant? Probably Not

Real Scenarios

Chimera v. Hand-Written Script

Scenario	Chimera (s)	Hand-Written (s)	Speed-up (%)
Sidejacking (4.1)	15.64	15.48	1.1%
IPs/domain (4.2.1)	8.81	8.72	0.96%
Phishing (4.4)	2.77	2.75	0.79%

Is Chimera fast enough in real-world scenarios? Yes

Future Work

- Optimizations
  - Relational Algebra (extensive prior work)
  - Analysis Logic – e.g., Bloom filter for EXCLUSIVE RIGHT JOIN
  - Parallel Processing
- Additional Targets
  - IBM Streams, others?

Availability

www.chimera-query.org

Now

- Language specification

Pending Pre-Publication Review

- Source code

Logistics

- BSD-style license
- Maintained independently from Bro
- (Hopefully) Bro compiler packaged with Bro distribution in the future

Thank You!

- Questions?