# Drilling Security Data

Charles S. Givre, CISSP
charles.givre@gtkcyber.com
www.thedataist.com

APACHE DRILL

DataDistillr

# 2 Huge Problems in Data Analytics

- Data is often stored in multiple disparate systems, and querying these systems requires understanding multiple query languages
- Joining data from these multiple disparate systems is difficult, costly and time consuming

**Que feriez-vous si toutes vos données parlaient la même langue?**

**Què faria vostè si totes les seves dades parlaven el mateix idioma?**

**ሁሉም የእርስዎ ውሂብ ተመሳሳይ ቋንቋ ሲጠቀም ምን ያደርጉ ነበር?**

# What would you do if all your data spoke the same language?

**Что бы вы сделали, если все ваши данные говорили на одном языке?**

**מה היית עושה אם כל הנתונים שלך מדברים באותה שפה?**

**اگر تمام داده های شما به همان زبان صحبت می کنند چه کاری انجام می دهید؟**

**Was würden Sie tun, wenn alle Ihre Daten die gleiche Sprache sprechen würden?**

# The Problem: Analyzing Security Data is Hard

# Security Data Analysis is Hard...Really Hard

# Why?

# Security Data comes in Many Forms

# Security Data Comes in Many Forms

- Standard Types such as JSON/CSV/XML
- Log Files: Event Logs, Database, Web Server etc.
- Syslog
- PCAP / PCAP-NG:  Binary, raw network traffic

# Security Data Lives In Many Places

- File systems
- Cloud Storage: Azure / S3
- Databases
- Real Time Event Streams
- Other sources

# Few tools can effectively analyze these data types effectively

Even fewer tools can effectively analyze **ALL** these data types effectively

# Splunk and ELK are solid SIEM platforms

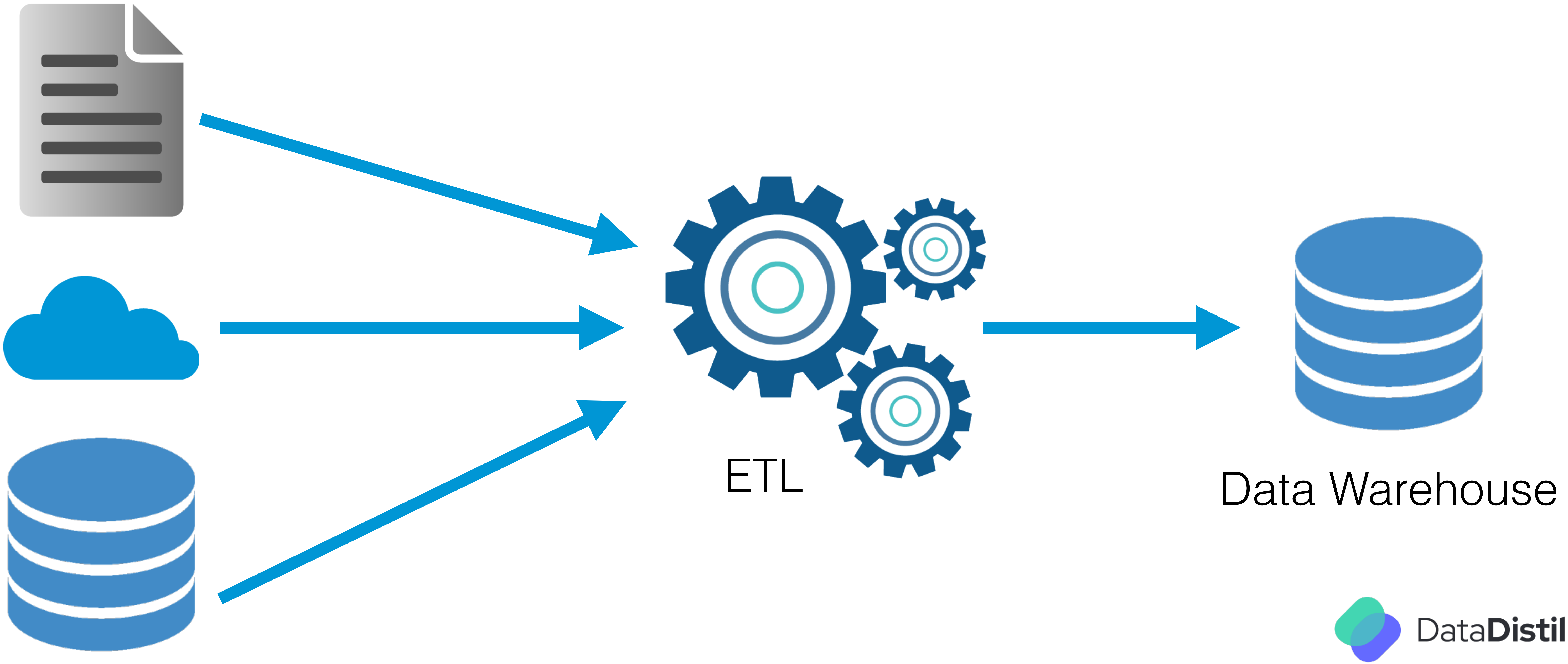
splunk>®

elasticsearch

# Wireshark is good for PCAP analysis

Security data is not arranged in an optimal way for ad-hoc analysis

# Security data is not arranged in an optimal way for ad-hoc analysis



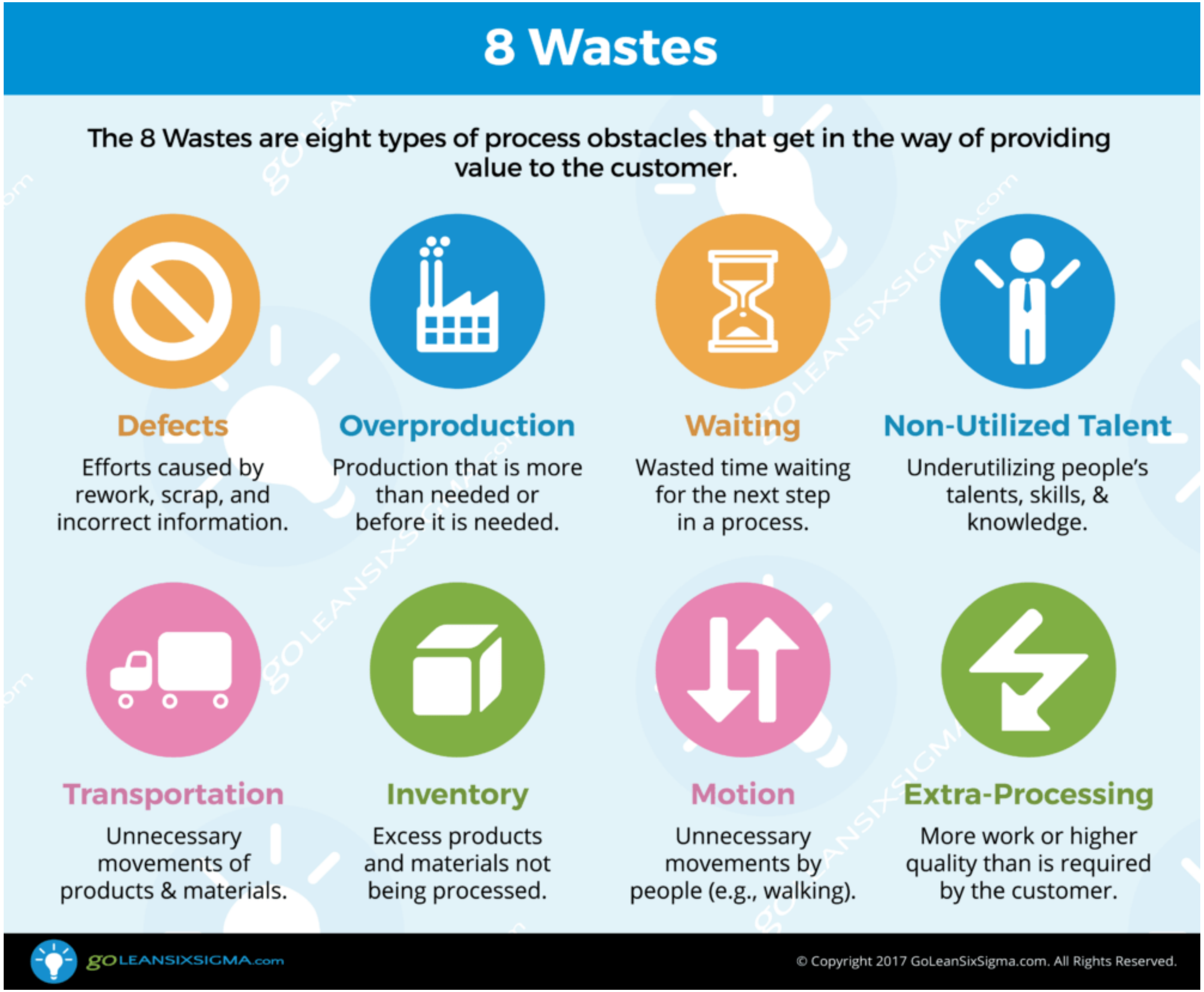
DataDistillr

# ETL is expensive and wasteful

Analytics teams spend between 50%-90% of their time preparing their data.

76% of Data Scientists say this is the **least enjoyable part** of their job.

The ETL Process **consumes the most time** and **contributes almost no value** to the end product.



# 8 Wastes

The 8 Wastes are eight types of process obstacles that get in the way of providing value to the customer.

### Defects

Efforts caused by rework, scrap, and incorrect information.

### Overproduction

Production that is more than needed or before it is needed.

### Waiting

Wasted time waiting for the next step in a process.

### Non-Utilized Talent

Underutilizing people's talents, skills, & knowledge.

### Transportation

Unnecessary movements of products & materials.

### Inventory

Excess products and materials not being processed.

### Motion

Unnecessary movements by people (e.g., walking).

### Extra-Processing

More work or higher quality than is required by the customer.

goLEANSIXSIGMA.com

*https://goleansixsigma.com/8-wastes/*

# So where does Drill fit in?

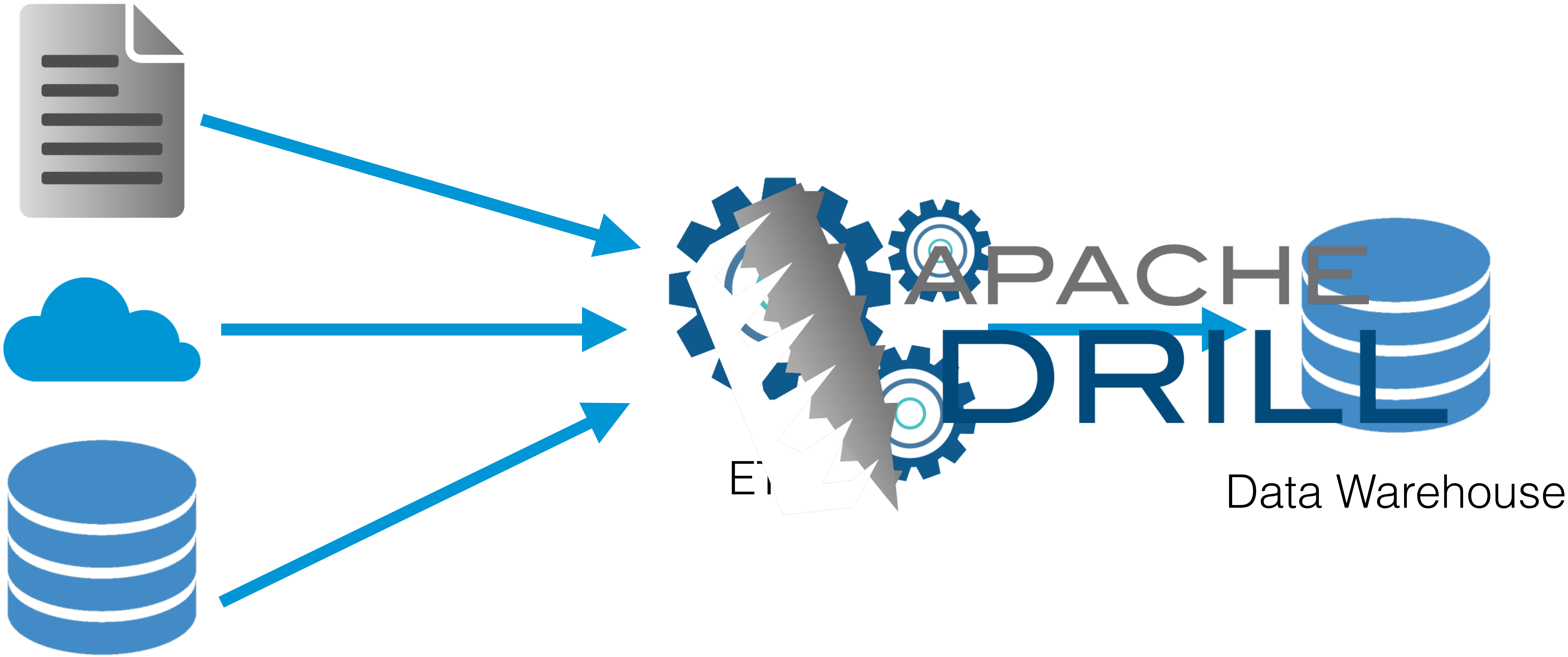
Apache Drill is an SQL Engine for self-describing data.

APACHE DRILL

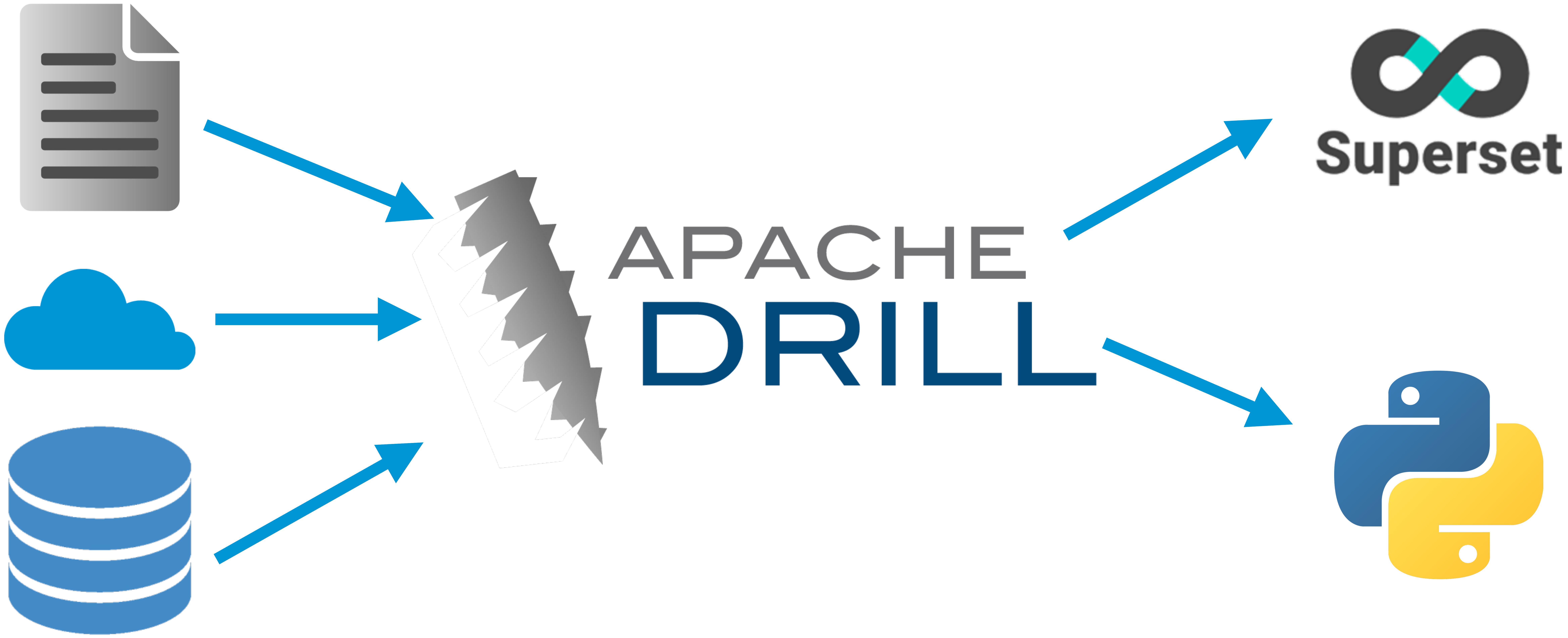
Drill lets you query anything*, wherever it is*, no matter its size** using standard SQL.

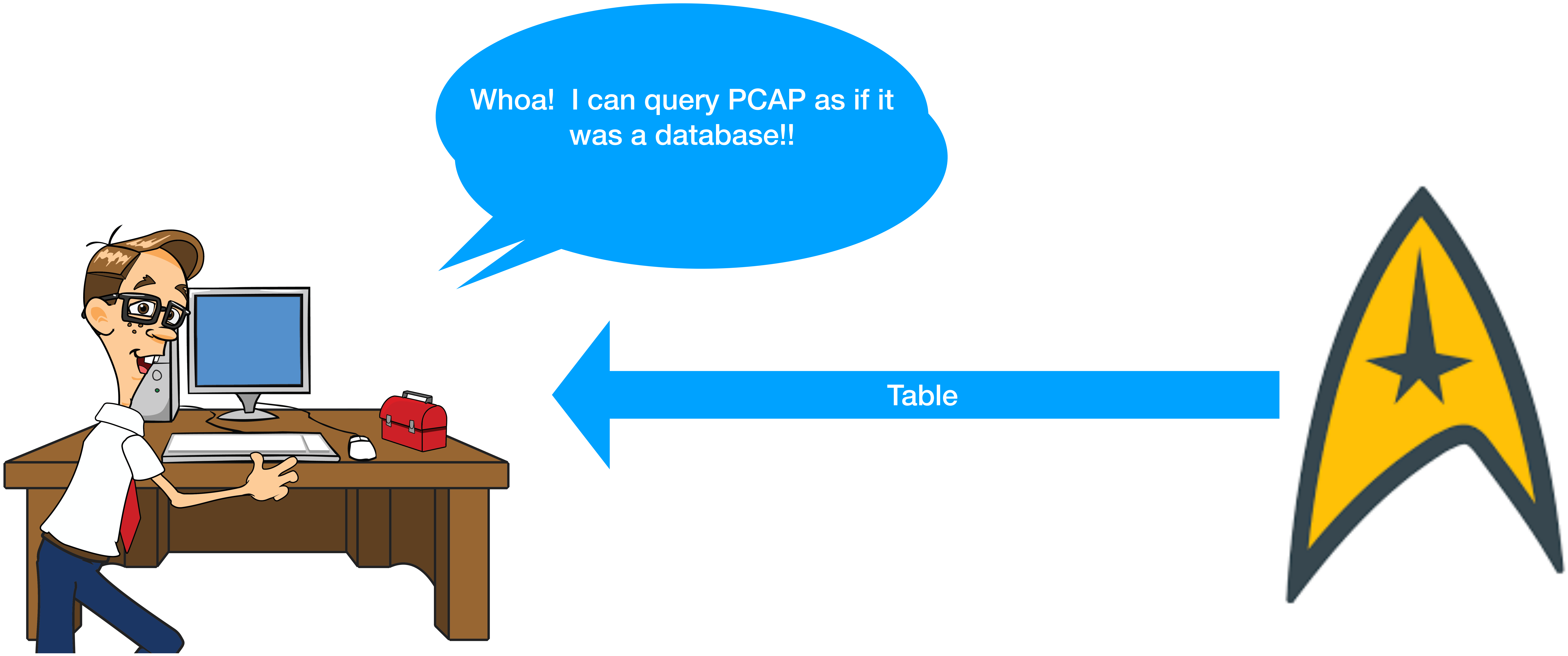
* well.. almost anything
** within reason

Drill acts as a universal translator for data.



DataDistillr


APACHE
DRILL
Superset

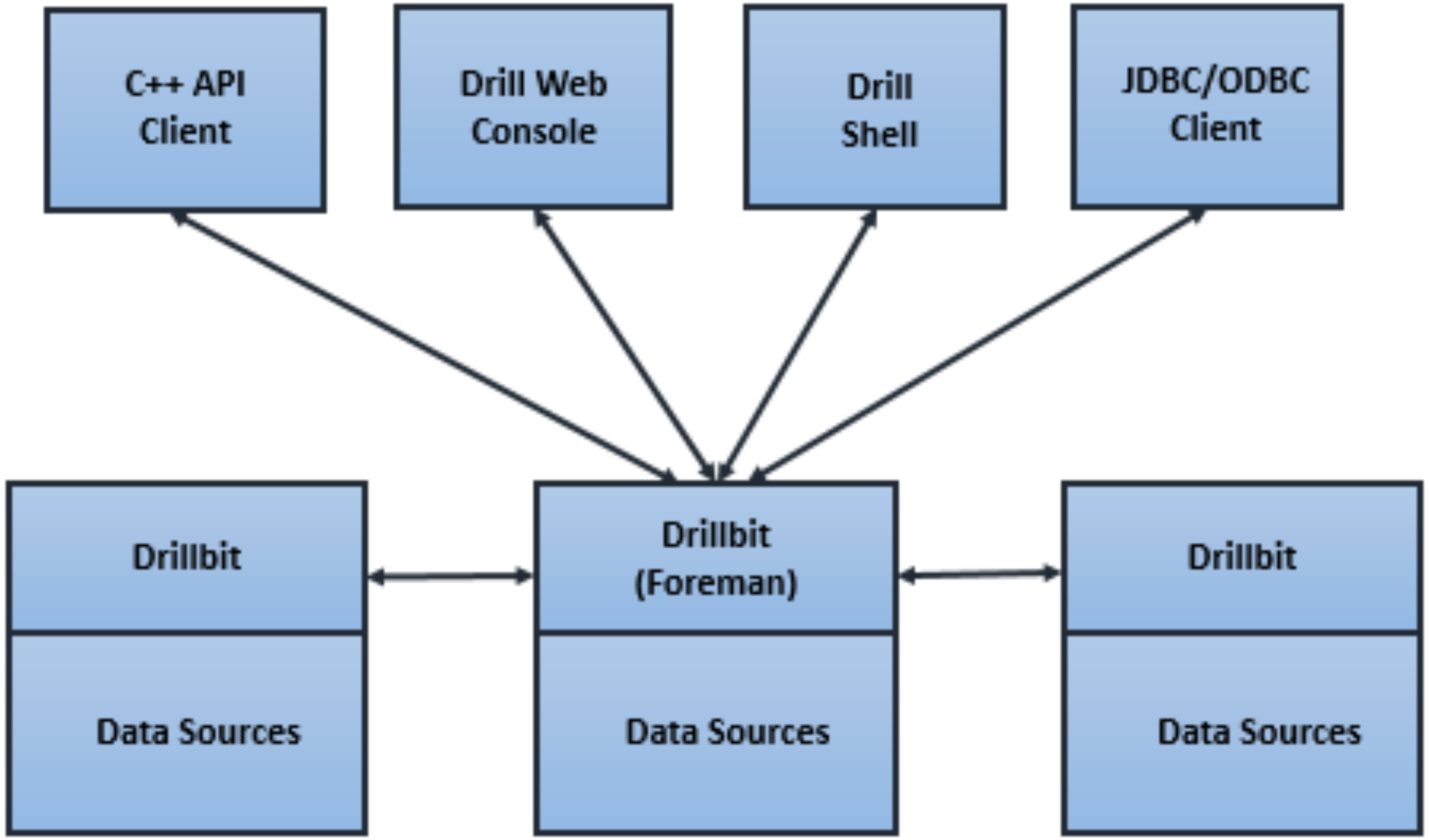


Whoa! I can query PCAP as if it was a database!!

Table

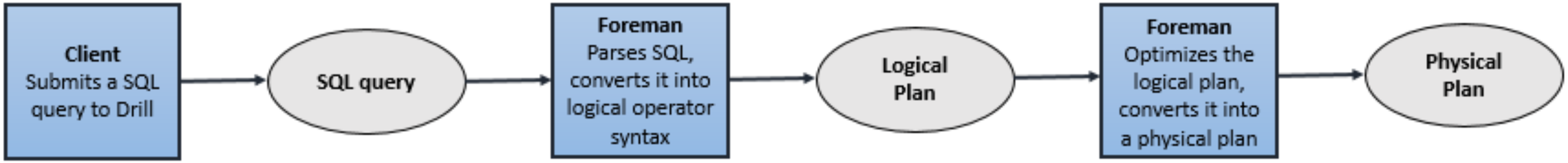
# How Drill Works

# How Drill Works



*https://drill.apache.org/docs/drill-query-execution/*

# How Drill Works



**Client**
Submits a SQL query to Drill

**SQL query**

**Foreman**
Parses SQL, converts it into logical operator syntax

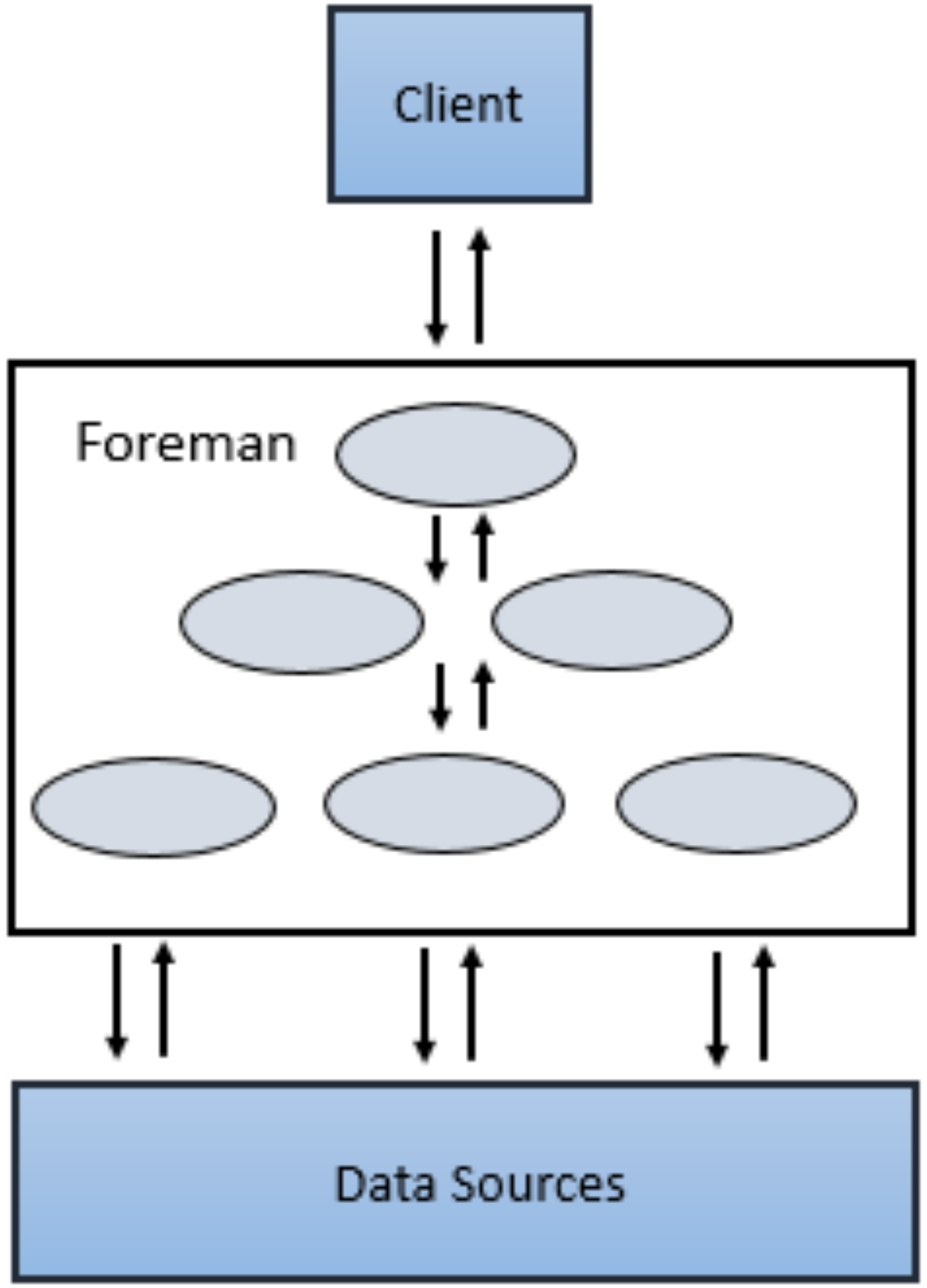
**Logical Plan**

**Foreman**
Optimizes the logical plan, converts it into a physical plan

**Physical Plan**

*https://drill.apache.org/docs/drill-query-execution/*

# How Drill Works

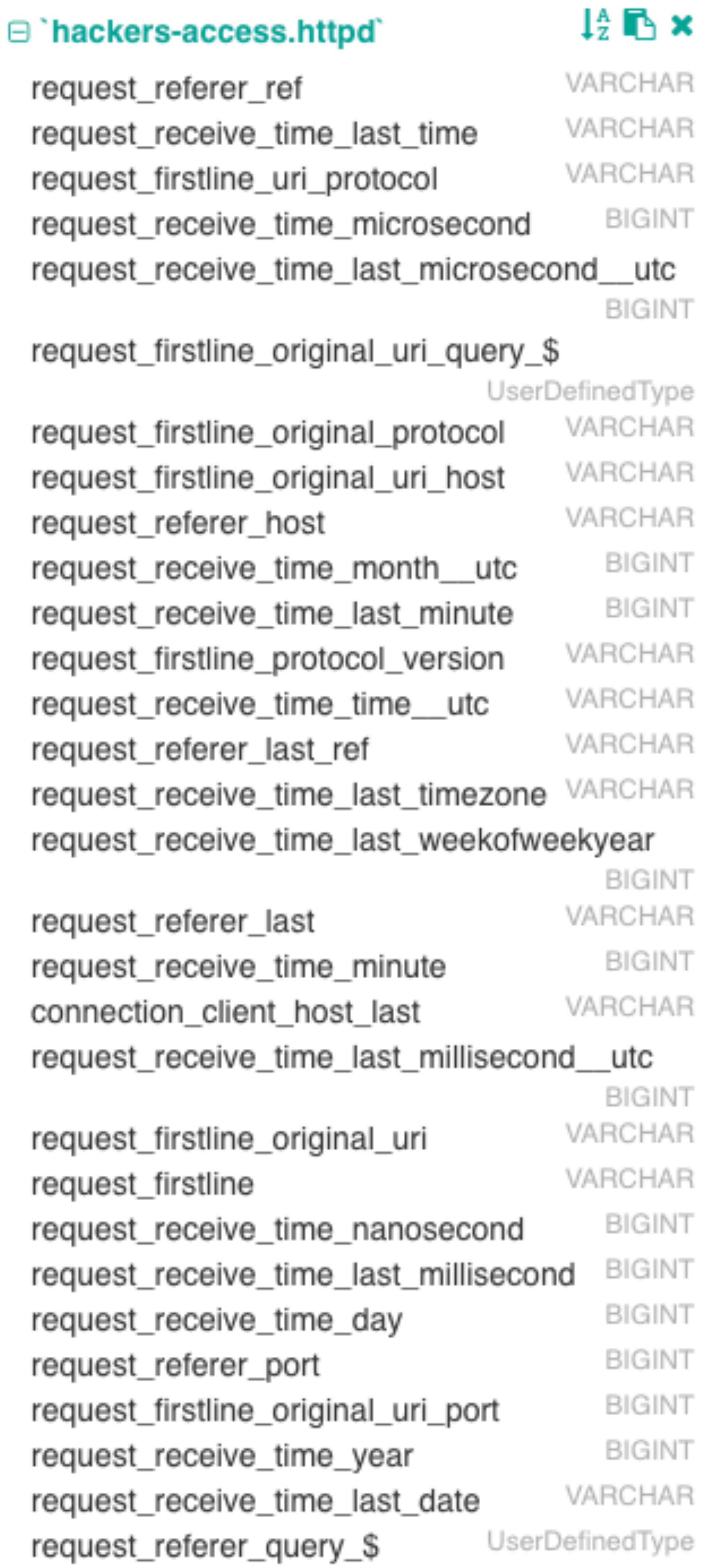


A parallelizer in the Foreman transforms the physical plan into multiple fragments that form an execution tree.

*https://drill.apache.org/docs/drill-query-execution/*

So let's take a look...

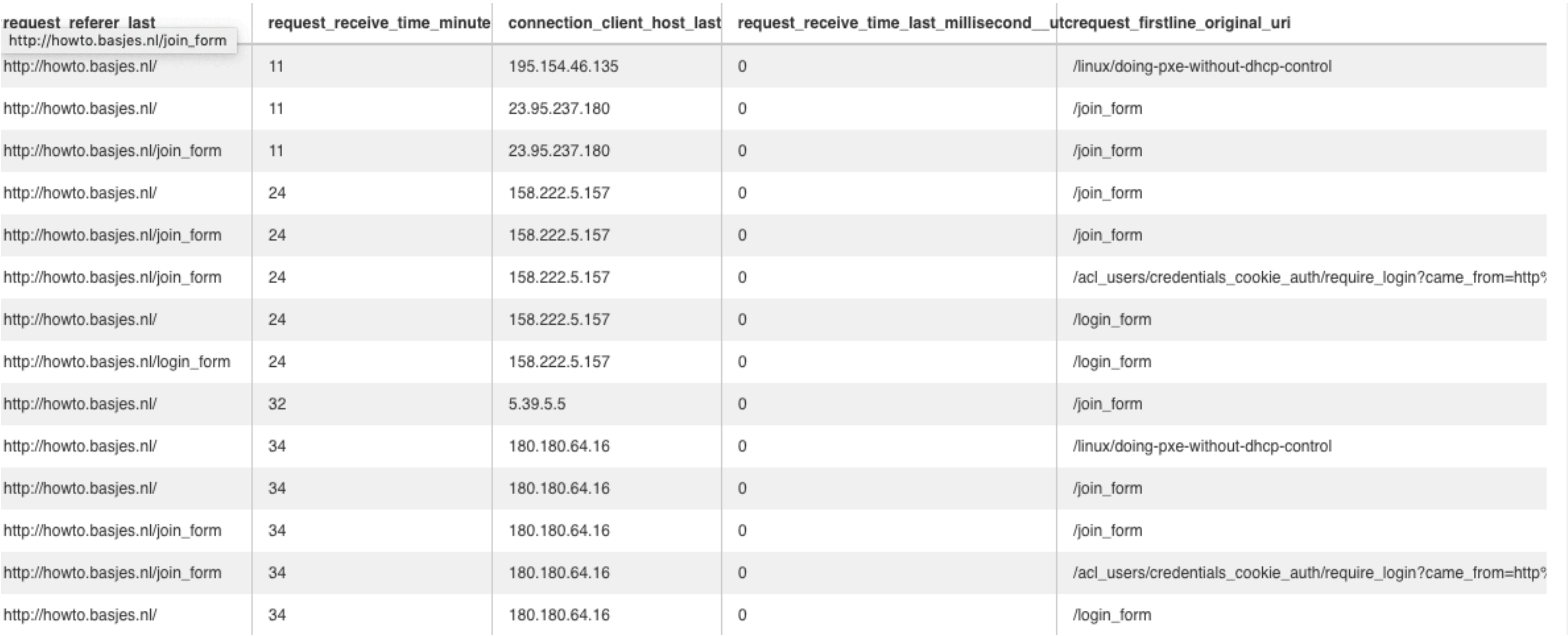
```
158.222.5.157 - - [25/Oct/2015:04:24:37 +0100] "GET /acl_users/
credentials_cookie_auth/require_login?came_from=http%3A//howto.basjes.nl/
join_form HTTP/1.1" 200 10716 "http://howto.basjes.nl/join_form" "Mozilla/5.0
(Windows NT 6.3; WOW64; rv:34.0) Gecko/20100101 Firefox/34.0 AlexaToolbar/
alxf-2.21"

158.222.5.157 - - [25/Oct/2015:04:24:39 +0100] "GET /login_form HTTP/1.1" 200
10543 "http://howto.basjes.nl/" "Mozilla/5.0 (Windows NT 6.3; WOW64; rv:34.0)
Gecko/20100101 Firefox/34.0 AlexaToolbar/alxf-2.21"
```



⊟ `hackers-access.httpd`

| | |
|---|---|
| request_referer_ref | VARCHAR |
| request_receive_time_last_time | VARCHAR |
| request_firstline_uri_protocol | VARCHAR |
| request_receive_time_microsecond | BIGINT |
| request_receive_time_last_microsecond__utc | BIGINT |
| request_firstline_original_uri_query_$ | UserDefinedType |
| request_firstline_original_protocol | VARCHAR |
| request_firstline_original_uri_host | VARCHAR |
| request_referer_host | VARCHAR |
| request_receive_time_month__utc | BIGINT |
| request_receive_time_last_minute | BIGINT |
| request_firstline_protocol_version | VARCHAR |
| request_receive_time_time__utc | VARCHAR |
| request_referer_last_ref | VARCHAR |
| request_receive_time_last_timezone | VARCHAR |
| request_receive_time_last_weekofweekyear | BIGINT |
| request_referer_last | VARCHAR |
| request_receive_time_minute | BIGINT |
| connection_client_host_last | VARCHAR |
| request_receive_time_last_millisecond__utc | BIGINT |
| request_firstline_original_uri | VARCHAR |
| request_firstline | VARCHAR |
| request_receive_time_nanosecond | BIGINT |
| request_receive_time_last_millisecond | BIGINT |
| request_receive_time_day | BIGINT |
| request_referer_port | BIGINT |
| request_firstline_original_uri_port | BIGINT |
| request_receive_time_year | BIGINT |
| request_receive_time_last_date | VARCHAR |
| request_referer_query_$ | UserDefinedType |

*https://github.com/nielsbasjes/logparser/tree/master/examples/demolog*



http://howto.basjes.nl/join_form

| request_referer_last | request_receive_time_minute | connection_client_host_last | request_receive_time_last_millisecond__utc | request_firstline_original_uri |
|---|---|---|---|---|
| http://howto.basjes.nl/ | 11 | 195.154.46.135 | 0 | /linux/doing-pxe-without-dhcp-control |
| http://howto.basjes.nl/ | 11 | 23.95.237.180 | 0 | /join_form |
| http://howto.basjes.nl/join_form | 11 | 23.95.237.180 | 0 | /join_form |
| http://howto.basjes.nl/ | 24 | 158.222.5.157 | 0 | /join_form |
| http://howto.basjes.nl/join_form | 24 | 158.222.5.157 | 0 | /join_form |
| http://howto.basjes.nl/join_form | 24 | 158.222.5.157 | 0 | /acl_users/credentials_cookie_auth/require_login?came_from=http% |
| http://howto.basjes.nl/ | 24 | 158.222.5.157 | 0 | /login_form |
| http://howto.basjes.nl/login_form | 24 | 158.222.5.157 | 0 | /login_form |
| http://howto.basjes.nl/ | 32 | 5.39.5.5 | 0 | /join_form |
| http://howto.basjes.nl/ | 34 | 180.180.64.16 | 0 | /linux/doing-pxe-without-dhcp-control |
| http://howto.basjes.nl/ | 34 | 180.180.64.16 | 0 | /join_form |
| http://howto.basjes.nl/join_form | 34 | 180.180.64.16 | 0 | /join_form |
| http://howto.basjes.nl/join_form | 34 | 180.180.64.16 | 0 | /acl_users/credentials_cookie_auth/require_login?came_from=http% |
| http://howto.basjes.nl/ | 34 | 180.180.64.16 | 0 | /login_form |

*https://github.com/nielsbasjes/logparser/tree/master/examples/demolog*

# Who is Trying to Hack This Site?

- The url /join_form is not public so anyone attempting to access this site, so almost anyone trying to access this probably a hacker…
- Let's see who is looking…

*https://github.com/nielsbasjes/logparser/tree/master/examples/demolog*

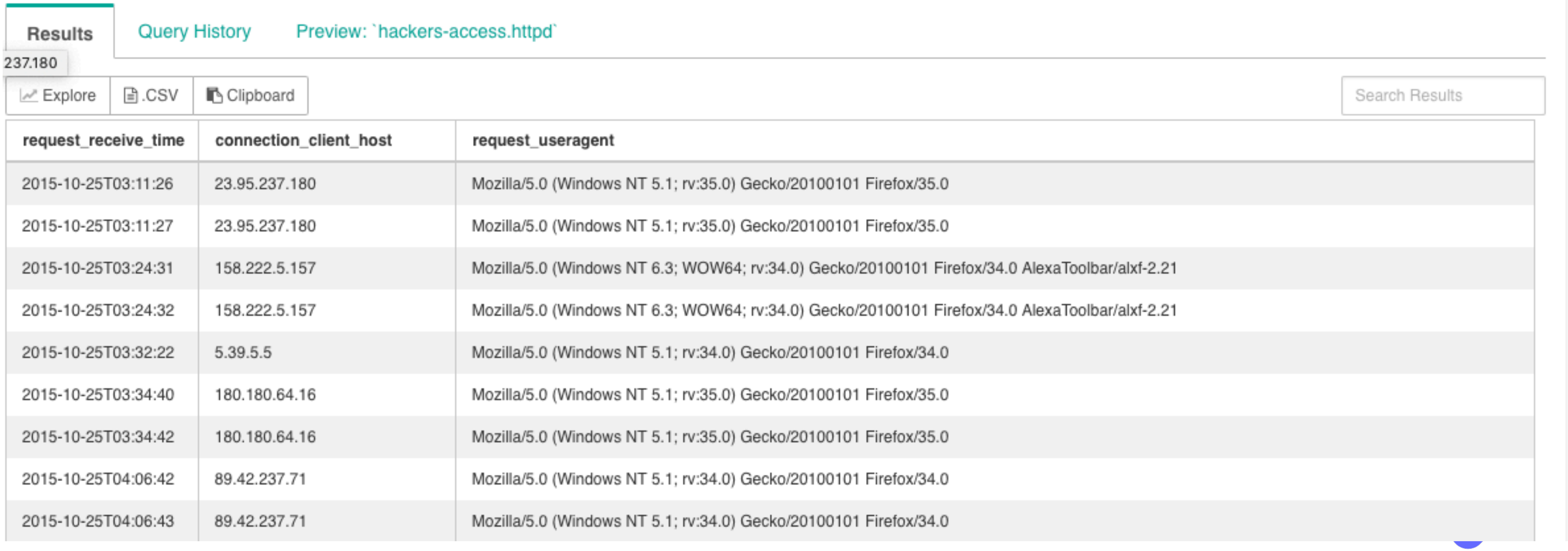
# Who is Trying to Hack This Site?

```
SELECT request_receive_time,
connection_client_host,
request_useragent
FROM dfs.test.`hackers-access.httpd`
WHERE request_firstline_uri = '/join_form'
```

*https://github.com/nielsbasjes/logparser/tree/master/examples/demolog*

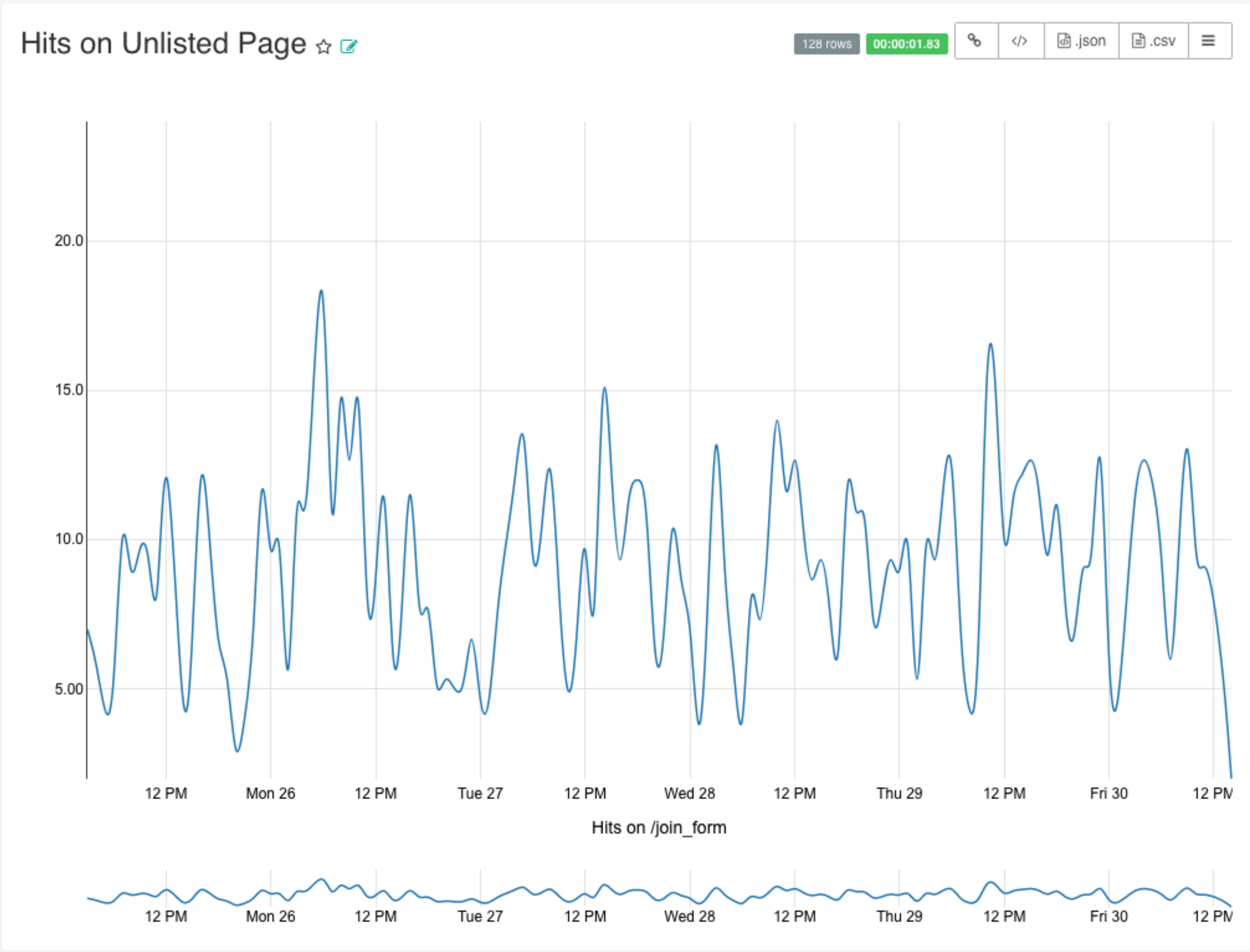
# Who is Trying to Hack This Site?

```
SELECT request_receive_time,
connection_client_host,
request_useragent
FROM dfs.test.`hackers-access.httpd`
WHERE request_firstline_uri = '/join_form'
```



| request_receive_time | connection_client_host | request_useragent |
|---|---|---|
| 2015-10-25T03:11:26 | 23.95.237.180 | Mozilla/5.0 (Windows NT 5.1; rv:35.0) Gecko/20100101 Firefox/35.0 |
| 2015-10-25T03:11:27 | 23.95.237.180 | Mozilla/5.0 (Windows NT 5.1; rv:35.0) Gecko/20100101 Firefox/35.0 |
| 2015-10-25T03:24:31 | 158.222.5.157 | Mozilla/5.0 (Windows NT 6.3; WOW64; rv:34.0) Gecko/20100101 Firefox/34.0 AlexaToolbar/alxf-2.21 |
| 2015-10-25T03:24:32 | 158.222.5.157 | Mozilla/5.0 (Windows NT 6.3; WOW64; rv:34.0) Gecko/20100101 Firefox/34.0 AlexaToolbar/alxf-2.21 |
| 2015-10-25T03:32:22 | 5.39.5.5 | Mozilla/5.0 (Windows NT 5.1; rv:34.0) Gecko/20100101 Firefox/34.0 |
| 2015-10-25T03:34:40 | 180.180.64.16 | Mozilla/5.0 (Windows NT 5.1; rv:35.0) Gecko/20100101 Firefox/35.0 |
| 2015-10-25T03:34:42 | 180.180.64.16 | Mozilla/5.0 (Windows NT 5.1; rv:35.0) Gecko/20100101 Firefox/35.0 |
| 2015-10-25T04:06:42 | 89.42.237.71 | Mozilla/5.0 (Windows NT 5.1; rv:34.0) Gecko/20100101 Firefox/34.0 |
| 2015-10-25T04:06:43 | 89.42.237.71 | Mozilla/5.0 (Windows NT 5.1; rv:34.0) Gecko/20100101 Firefox/34.0 |

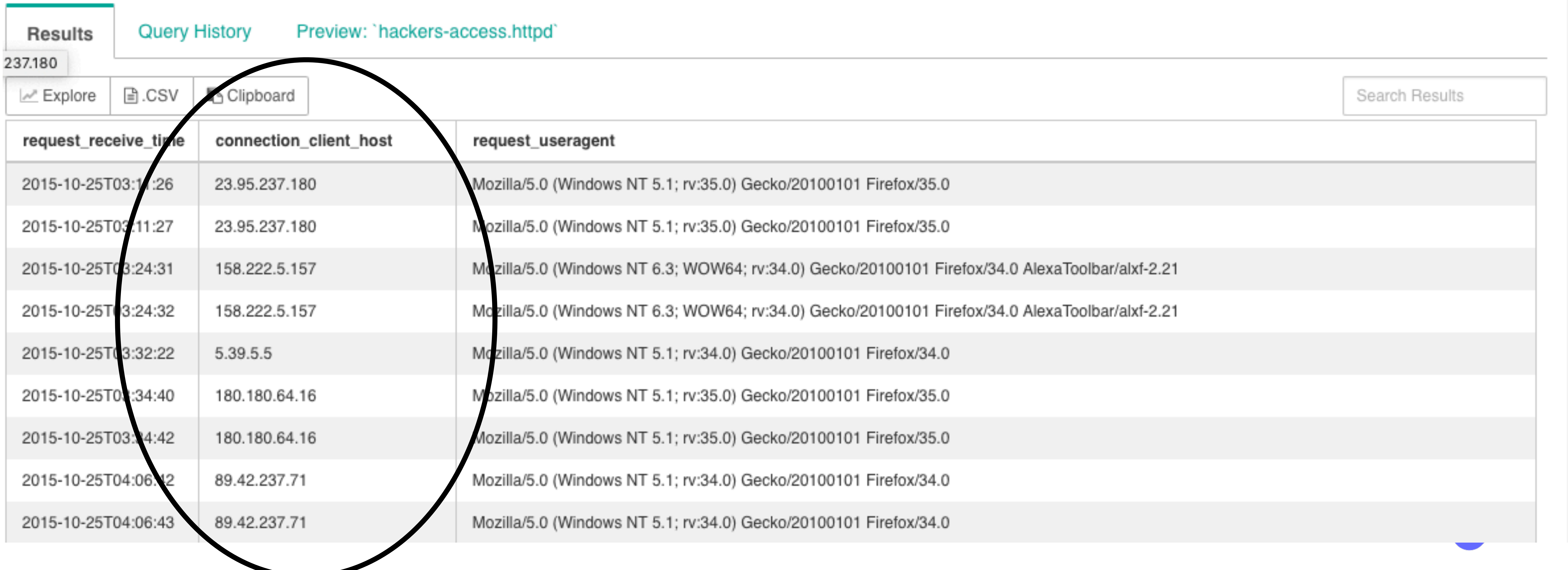
# Who is Trying to Hack This Site?



Let's take a look at those IP addresses shall we?

# Who is Trying to Hack This Site?

```
SELECT request_receive_time,
connection_client_host,
request_useragent
FROM dfs.test.`hackers-access.httpd`
WHERE request_firstline_uri = '/join_form'
```



| request_receive_time | connection_client_host | request_useragent |
|---|---|---|
| 2015-10-25T03:11:26 | 23.95.237.180 | Mozilla/5.0 (Windows NT 5.1; rv:35.0) Gecko/20100101 Firefox/35.0 |
| 2015-10-25T03:11:27 | 23.95.237.180 | Mozilla/5.0 (Windows NT 5.1; rv:35.0) Gecko/20100101 Firefox/35.0 |
| 2015-10-25T03:24:31 | 158.222.5.157 | Mozilla/5.0 (Windows NT 6.3; WOW64; rv:34.0) Gecko/20100101 Firefox/34.0 AlexaToolbar/alxf-2.21 |
| 2015-10-25T03:24:32 | 158.222.5.157 | Mozilla/5.0 (Windows NT 6.3; WOW64; rv:34.0) Gecko/20100101 Firefox/34.0 AlexaToolbar/alxf-2.21 |
| 2015-10-25T03:32:22 | 5.39.5.5 | Mozilla/5.0 (Windows NT 5.1; rv:34.0) Gecko/20100101 Firefox/34.0 |
| 2015-10-25T03:34:40 | 180.180.64.16 | Mozilla/5.0 (Windows NT 5.1; rv:35.0) Gecko/20100101 Firefox/35.0 |
| 2015-10-25T03:34:42 | 180.180.64.16 | Mozilla/5.0 (Windows NT 5.1; rv:35.0) Gecko/20100101 Firefox/35.0 |
| 2015-10-25T04:06:42 | 89.42.237.71 | Mozilla/5.0 (Windows NT 5.1; rv:34.0) Gecko/20100101 Firefox/34.0 |
| 2015-10-25T04:06:43 | 89.42.237.71 | Mozilla/5.0 (Windows NT 5.1; rv:34.0) Gecko/20100101 Firefox/34.0 |

Drill's Flexible UDF Interface Allows you to write your own functions.

# A collection of GeoIP Functions is available on GitHub.

*https://github.com/cgivre/drill-geoip-functions*

DataDistillr

# Drill GeoIP Functions

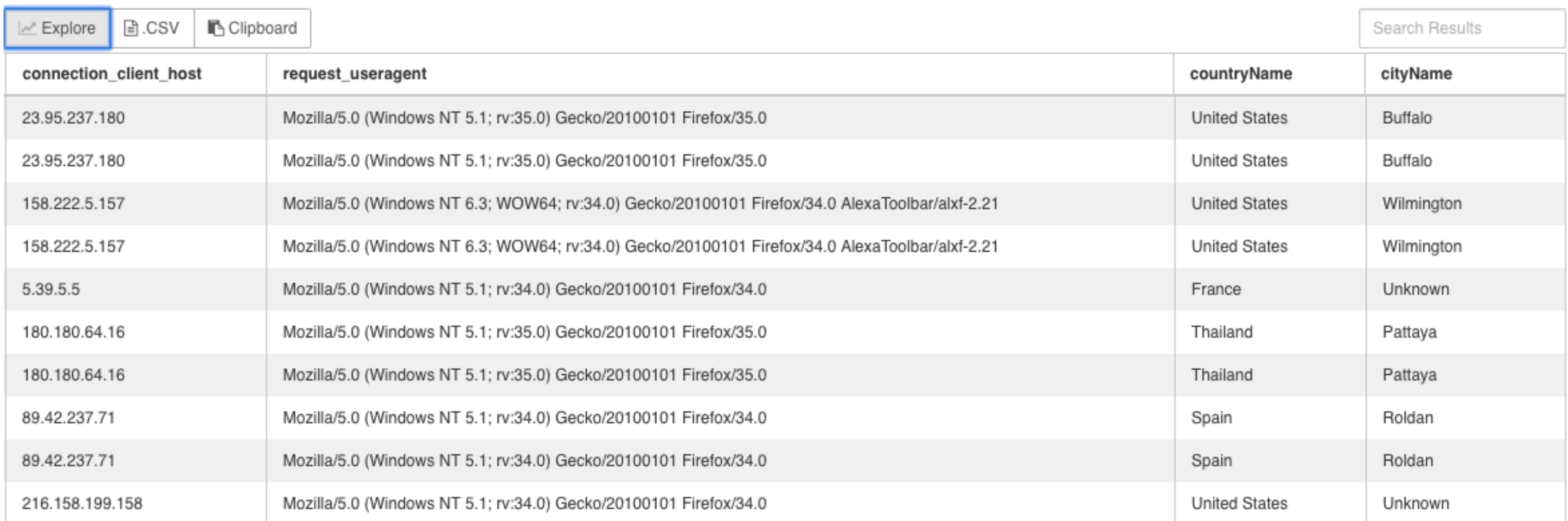
- **`getCountryName( <ip> )`**: This function returns the country name of the IP address, "Unknown" if the IP is unknown or invalid.
- **`getCountryConfidence( <ip> )`**: This function returns the confidence score of the country ISO code of the IP address.
- **`getCountryISOCode( <ip> )`**: This function returns the country ISO code of the IP address, "Unknown" if the IP is unknown or invalid.
- **`getCityName( <ip> )`**: This function returns the city name of the IP address, "Unknown" if the IP is unknown or invalid.
- **`getCityConfidence( <ip> )`**: This function returns confidence score of the city name of the IP address.
- **`getLatitude( <ip> )`**: This function returns the latitude associated with the IP address.
- **`getLongitude( <ip> )`**: This function returns the longitude associated with the IP address.
- **`getTimezone( <ip> )`**: This function returns the timezone associated with the IP address.
- **`getAccuracyRadius( <ip> )`**: This function returns the accuracy radius associated with the IP address, 0 if unknown.
- **`getAverageIncome( <ip> )`**: This function returns the average income of the region associated with the IP address, 0 if unknown.
- **`getMetroCode( <ip> )`**: This function returns the metro code of the region associated with the IP address, 0 if unknown.
- **`getPopulationDensity( <ip> )`**: This function returns the population density associated with the IP address.
- **`getPostalCode( <ip> )`**: This function returns the postal code associated with the IP address.
- **`getCoordPoint( <ip> )`**: This function returns a point for use in GIS functions of the lat/long of associated with the IP address.
- **`getASN( <ip> )`**: This function returns the autonomous system of the IP address, "Unknown" if the IP is unknown or invalid.
- **`getASNOrganization( <ip> )`**: This function returns the autonomous system organization of the IP address, "Unknown" if the IP is unknown or invalid.
- **`isEU( <ip> ), isEuropeanUnion( <ip> )`**: This function returns `true` if the ip address is located in the European Union, `false` if not.
- **`isAnonymous( <ip> )`**: This function returns `true` if the ip address is anonymous, `false` if not.
- **`isAnonymousVPN( <ip> )`**: This function returns `true` if the ip address is an anonymous virtual private network (VPN), `false` if not.
- **`isHostingProvider( <ip> )`**: This function returns `true` if the ip address is a hosting provider, `false` if not.
- **`isPublciProxy( <ip> )`**: This function returns `true` if the ip address is a public proxy, `false` if not.
- **`isTORExitNode( <ip> )`**: This function returns `true` if the ip address is a known TOR exit node, `false` if not.

*https://github.com/cgivre/drill-geoip-functions*

DataDistillr

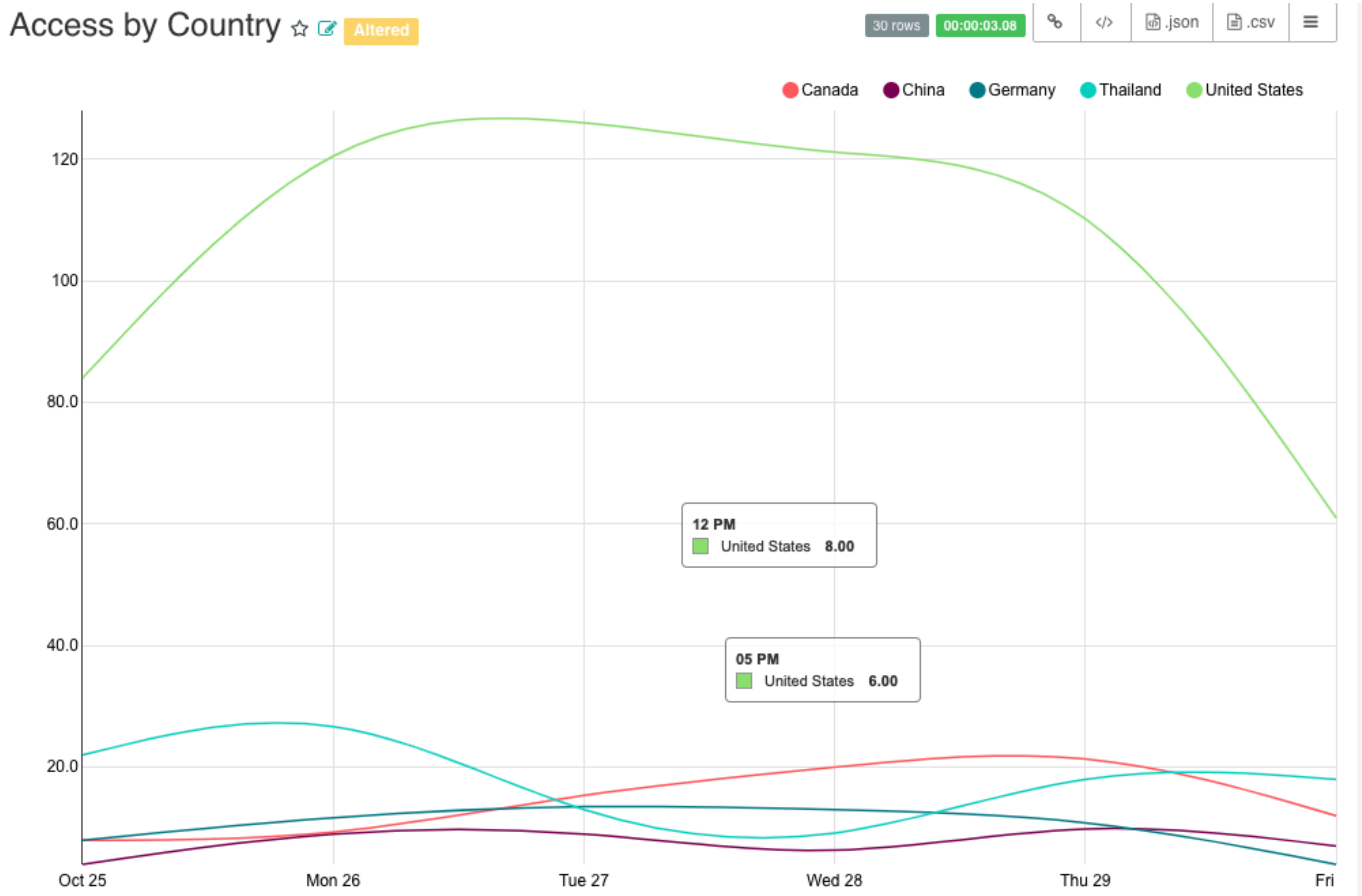
# Who is Trying to Hack This Site?

```
SELECT request_receive_time, connection_client_host, request_useragent,
getCountryName(connection_client_host ) as countryName,
getCityName(connection_client_host ) as cityName
FROM dfs.test.`hackers-access.httpd`
WHERE request_firstline_uri = '/join_form'
```



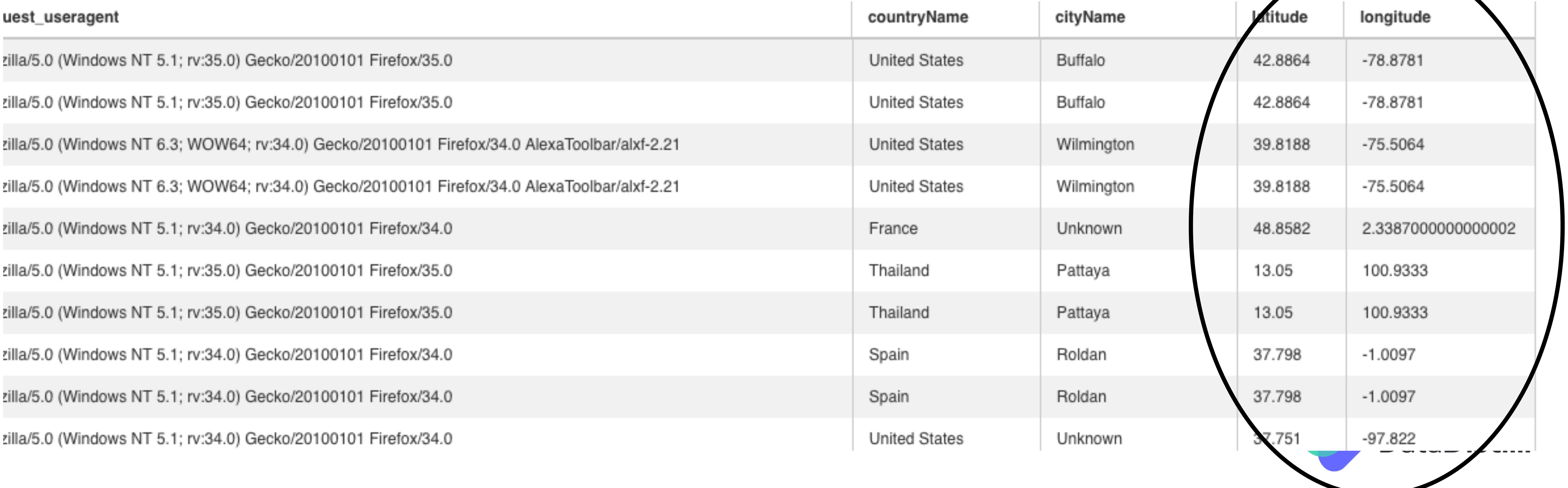
| connection_client_host | request_useragent | countryName | cityName |
|---|---|---|---|
| 23.95.237.180 | Mozilla/5.0 (Windows NT 5.1; rv:35.0) Gecko/20100101 Firefox/35.0 | United States | Buffalo |
| 23.95.237.180 | Mozilla/5.0 (Windows NT 5.1; rv:35.0) Gecko/20100101 Firefox/35.0 | United States | Buffalo |
| 158.222.5.157 | Mozilla/5.0 (Windows NT 6.3; WOW64; rv:34.0) Gecko/20100101 Firefox/34.0 AlexaToolbar/alxf-2.21 | United States | Wilmington |
| 158.222.5.157 | Mozilla/5.0 (Windows NT 6.3; WOW64; rv:34.0) Gecko/20100101 Firefox/34.0 AlexaToolbar/alxf-2.21 | United States | Wilmington |
| 5.39.5.5 | Mozilla/5.0 (Windows NT 5.1; rv:34.0) Gecko/20100101 Firefox/34.0 | France | Unknown |
| 180.180.64.16 | Mozilla/5.0 (Windows NT 5.1; rv:35.0) Gecko/20100101 Firefox/35.0 | Thailand | Pattaya |
| 180.180.64.16 | Mozilla/5.0 (Windows NT 5.1; rv:35.0) Gecko/20100101 Firefox/35.0 | Thailand | Pattaya |
| 89.42.237.71 | Mozilla/5.0 (Windows NT 5.1; rv:34.0) Gecko/20100101 Firefox/34.0 | Spain | Roldan |
| 89.42.237.71 | Mozilla/5.0 (Windows NT 5.1; rv:34.0) Gecko/20100101 Firefox/34.0 | Spain | Roldan |
| 216.158.199.158 | Mozilla/5.0 (Windows NT 5.1; rv:34.0) Gecko/20100101 Firefox/34.0 | United States | Unknown |

# Who is Trying to Hack This Site?



# Who is Trying to Hack This Site?

```
SELECT request_receive_time, connection_client_host, request_useragent,
getCountryName(connection_client_host ) as countryName,
getCityName(connection_client_host ) as cityName,
getLatitude(connection_client_host ) as latitude,
getLongitude(connection_client_host ) as longitude
```



| uest_useragent | countryName | cityName | latitude | longitude |
|---|---|---|---|---|
| zilla/5.0 (Windows NT 5.1; rv:35.0) Gecko/20100101 Firefox/35.0 | United States | Buffalo | 42.8864 | -78.8781 |
| zilla/5.0 (Windows NT 5.1; rv:35.0) Gecko/20100101 Firefox/35.0 | United States | Buffalo | 42.8864 | -78.8781 |
| zilla/5.0 (Windows NT 6.3; WOW64; rv:34.0) Gecko/20100101 Firefox/34.0 AlexaToolbar/alxf-2.21 | United States | Wilmington | 39.8188 | -75.5064 |
| zilla/5.0 (Windows NT 6.3; WOW64; rv:34.0) Gecko/20100101 Firefox/34.0 AlexaToolbar/alxf-2.21 | United States | Wilmington | 39.8188 | -75.5064 |
| zilla/5.0 (Windows NT 5.1; rv:34.0) Gecko/20100101 Firefox/34.0 | France | Unknown | 48.8582 | 2.3387000000000002 |
| zilla/5.0 (Windows NT 5.1; rv:35.0) Gecko/20100101 Firefox/35.0 | Thailand | Pattaya | 13.05 | 100.9333 |
| zilla/5.0 (Windows NT 5.1; rv:35.0) Gecko/20100101 Firefox/35.0 | Thailand | Pattaya | 13.05 | 100.9333 |
| zilla/5.0 (Windows NT 5.1; rv:34.0) Gecko/20100101 Firefox/34.0 | Spain | Roldan | 37.798 | -1.0097 |
| zilla/5.0 (Windows NT 5.1; rv:34.0) Gecko/20100101 Firefox/34.0 | Spain | Roldan | 37.798 | -1.0097 |
| zilla/5.0 (Windows NT 5.1; rv:34.0) Gecko/20100101 Firefox/34.0 | United States | Unknown | 37.751 | -97.822 |

# What equipment are they using?

# What equipment are they using?

- Drill has support for multidimensional data structures including KV Pairs and Lists.
- There is a pre-existing UDF (https://github.com/nielsbasjes/yauaa) for Apache Drill which can parse User Agent Strings and get you a lot of useful information from the UA string.

# Who is Trying to Hack This Site?

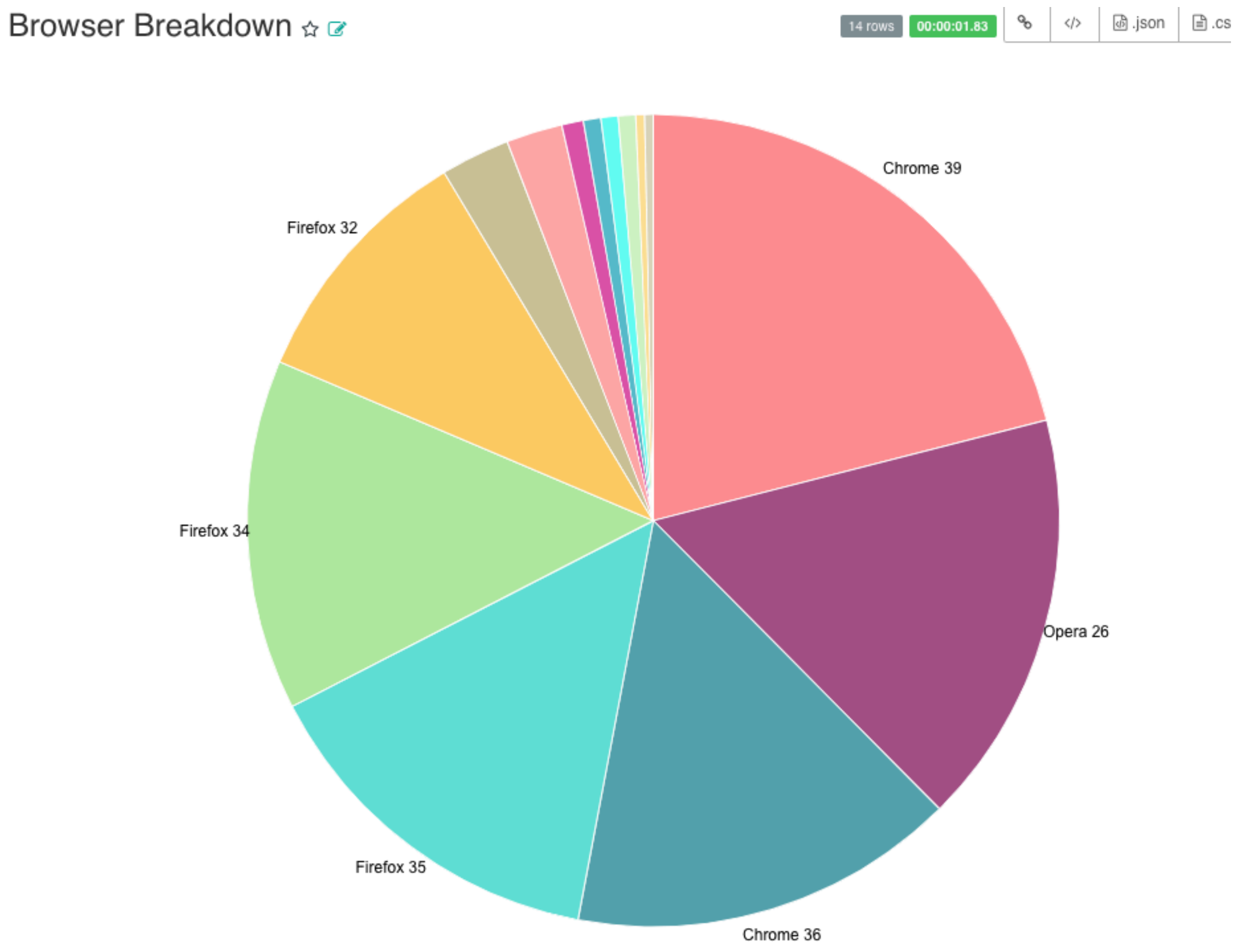
```
SELECT parse_user_agent(request_useragent) AS ua
FROM dfs.test.`hackers-access.httpd`
WHERE request_firstline_uri = '/join_form'
```

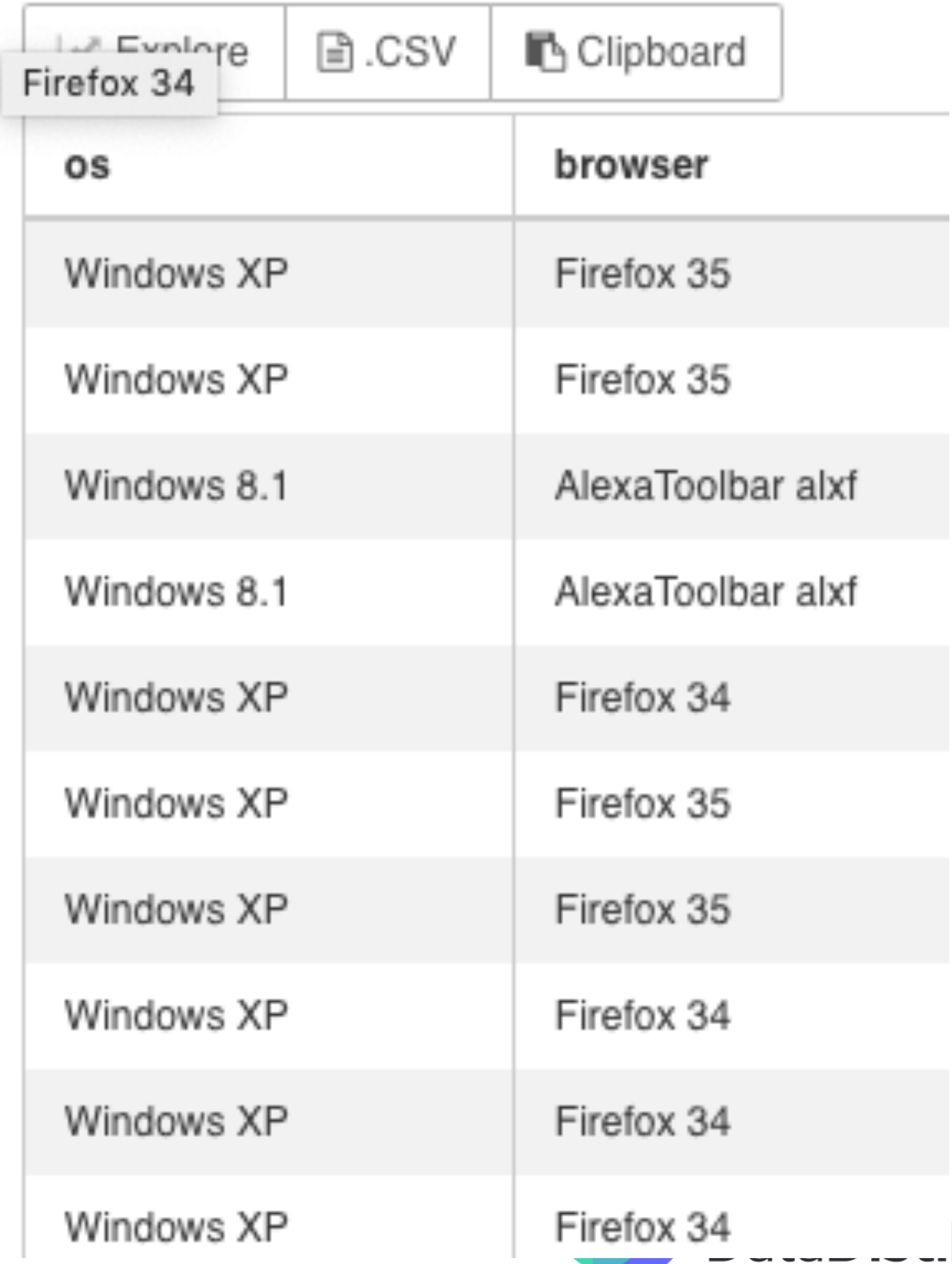
Explore
.CSV
Clipboard
Search Results

| ua |
|---|
| {"DeviceClass":"Desktop","DeviceName":"Desktop","DeviceBrand":"Unknown","DeviceCpuBits":"32","OperatingSystemClass":"Desktop","OperatingSystemName":"Windows NT","OperatingSystemVersion":"XP","Operati |
| {"DeviceClass":"Desktop","DeviceName":"Desktop","DeviceBrand":"Unknown","DeviceCpuBits":"32","OperatingSystemClass":"Desktop","OperatingSystemName":"Windows NT","OperatingSystemVersion":"XP","Operati |
| {"DeviceClass":"Desktop","DeviceName":"Desktop","DeviceBrand":"Unknown","DeviceCpuBits":"64","OperatingSystemClass":"Desktop","OperatingSystemName":"Windows NT","OperatingSystemVersion":"8.1","Operati |
| {"DeviceClass":"Desktop","DeviceName":"Desktop","DeviceBrand":"Unknown","DeviceCpuBits":"64","OperatingSystemClass":"Desktop","OperatingSystemName":"Windows NT","OperatingSystemVersion":"8.1","Operati |
| {"DeviceClass":"Desktop","DeviceName":"Desktop","DeviceBrand":"Unknown","DeviceCpuBits":"32","OperatingSystemClass":"Desktop","OperatingSystemName":"Windows NT","OperatingSystemVersion":"XP","Operati |
| {"DeviceClass":"Desktop","DeviceName":"Desktop","DeviceBrand":"Unknown","DeviceCpuBits":"32","OperatingSystemClass":"Desktop","OperatingSystemName":"Windows NT","OperatingSystemVersion":"XP","Operati |
| {"DeviceClass":"Desktop","DeviceName":"Desktop","DeviceBrand":"Unknown","DeviceCpuBits":"32","OperatingSystemClass":"Desktop","OperatingSystemName":"Windows NT","OperatingSystemVersion":"XP","Operati |
| {"DeviceClass":"Desktop","DeviceName":"Desktop","DeviceBrand":"Unknown","DeviceCpuBits":"32","OperatingSystemClass":"Desktop","OperatingSystemName":"Windows NT","OperatingSystemVersion":"XP","Operati |
| {"DeviceClass":"Desktop","DeviceName":"Desktop","DeviceBrand":"Unknown","DeviceCpuBits":"32","OperatingSystemClass":"Desktop","OperatingSystemName":"Windows NT","OperatingSystemVersion":"XP","Operati |
| {"DeviceClass":"Desktop","DeviceName":"Desktop","DeviceBrand":"Unknown","DeviceCpuBits":"32","OperatingSystemClass":"Desktop","OperatingSystemName":"Windows NT","OperatingSystemVersion":"XP","Operati |

# Who is Trying to Hack This Site?

```
SELECT table1.ua.OperatingSystemNameVersion as os,
table1.ua.AgentNameVersionMajor as browser
FROM
(
SELECT
parse_user_agent(request_useragent) AS ua
FROM dfs.test.`hackers-access.httpd`
WHERE request_firstline_uri = '/join_form'
) AS table1
```


Browser Breakdown



| os | browser |
|---|---|
| Windows XP | Firefox 35 |
| Windows XP | Firefox 35 |
| Windows 8.1 | AlexaToolbar alxf |
| Windows 8.1 | AlexaToolbar alxf |
| Windows XP | Firefox 34 |
| Windows XP | Firefox 35 |
| Windows XP | Firefox 35 |
| Windows XP | Firefox 34 |
| Windows XP | Firefox 34 |
| Windows XP | Firefox 34 |

# Questions so far?

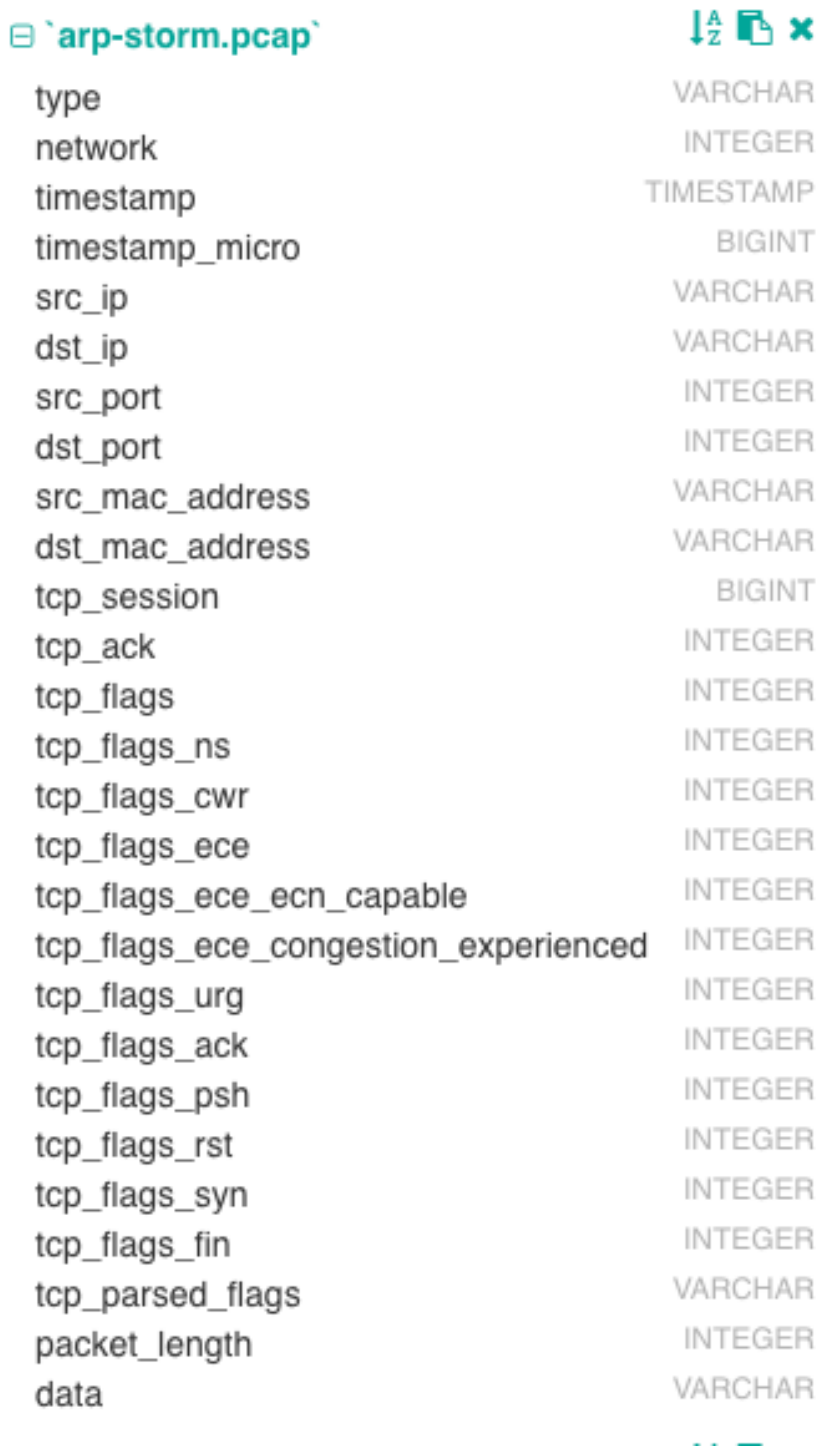
# Let's have some fun with PCAP

# Let's have some fun with PCAP

- PCAP is short for **P**acket **Cap**ture and represent raw network traffic.

- Files are encoded in binary format, but various tools exist to analyze PCAP files or convert them into more easily accessible formats, such as JSON.

# PCAP Analysis



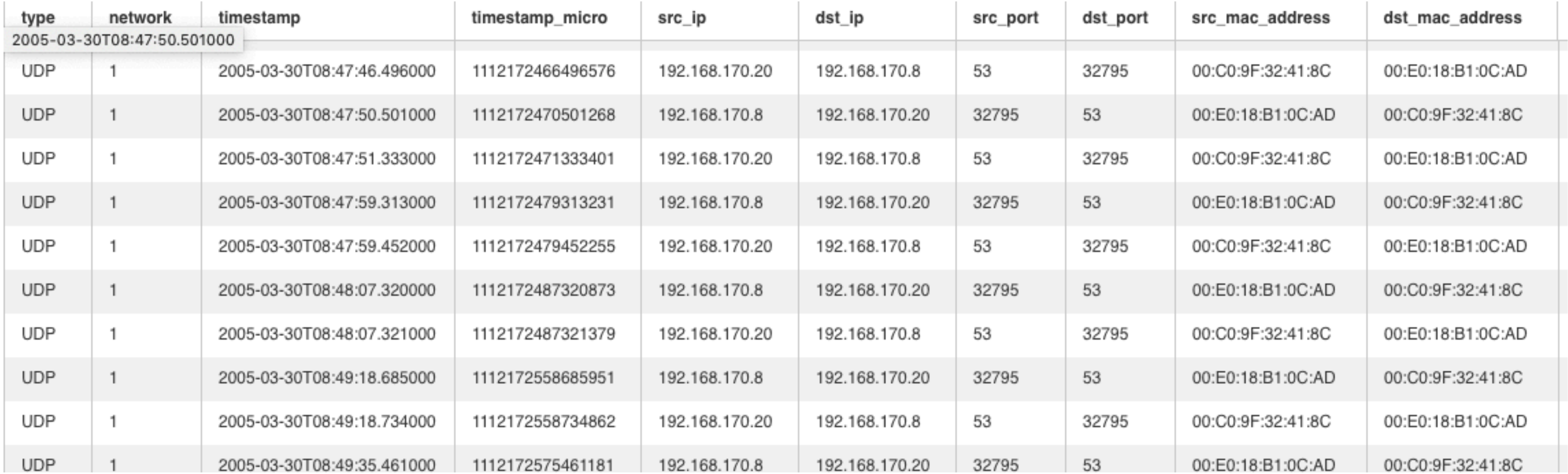
⊟ `arp-storm.pcap`

| | |
|---|---|
| type | VARCHAR |
| network | INTEGER |
| timestamp | TIMESTAMP |
| timestamp_micro | BIGINT |
| src_ip | VARCHAR |
| dst_ip | VARCHAR |
| src_port | INTEGER |
| dst_port | INTEGER |
| src_mac_address | VARCHAR |
| dst_mac_address | VARCHAR |
| tcp_session | BIGINT |
| tcp_ack | INTEGER |
| tcp_flags | INTEGER |
| tcp_flags_ns | INTEGER |
| tcp_flags_cwr | INTEGER |
| tcp_flags_ece | INTEGER |
| tcp_flags_ece_ecn_capable | INTEGER |
| tcp_flags_ece_congestion_experienced | INTEGER |
| tcp_flags_urg | INTEGER |
| tcp_flags_ack | INTEGER |
| tcp_flags_psh | INTEGER |
| tcp_flags_rst | INTEGER |
| tcp_flags_syn | INTEGER |
| tcp_flags_fin | INTEGER |
| tcp_parsed_flags | VARCHAR |
| packet_length | INTEGER |
| data | VARCHAR |

# PCAP Analysis

- Drill has a comprehensive collection of networking functions to facilitate network analysis
- is_private_ip(), is_valid_ip(), in_network() and many others.
- inet_aton() and inet_ntoa() exist to facilitate sorting IP ranges.

# PCAP Analysis



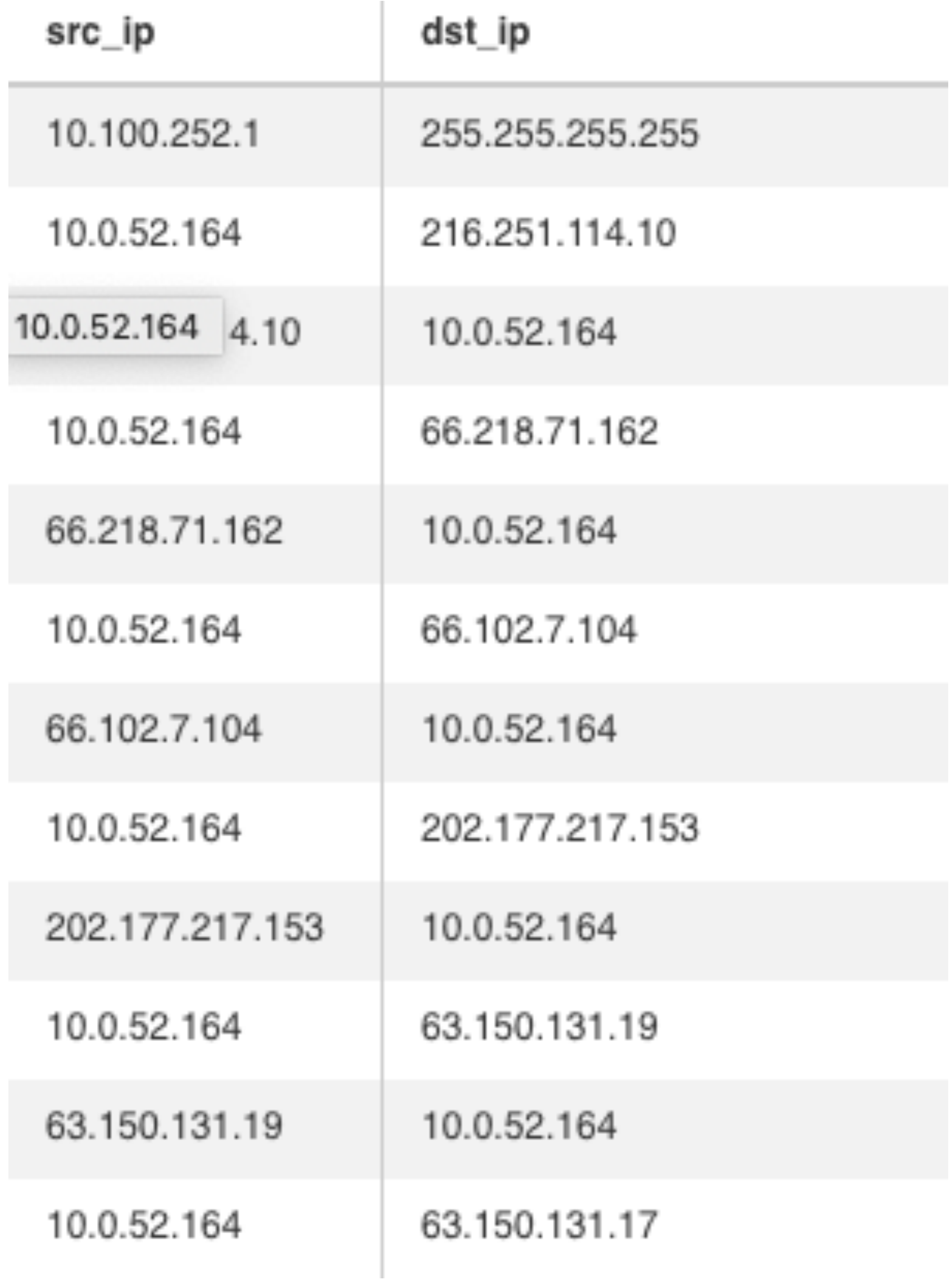
2005-03-30T08:47:50.501000

| type | network | timestamp | timestamp_micro | src_ip | dst_ip | src_port | dst_port | src_mac_address | dst_mac_address |
|---|---|---|---|---|---|---|---|---|---|
| UDP | 1 | 2005-03-30T08:47:46.496000 | 1112172466496576 | 192.168.170.20 | 192.168.170.8 | 53 | 32795 | 00:C0:9F:32:41:8C | 00:E0:18:B1:0C:AD |
| UDP | 1 | 2005-03-30T08:47:50.501000 | 1112172470501268 | 192.168.170.8 | 192.168.170.20 | 32795 | 53 | 00:E0:18:B1:0C:AD | 00:C0:9F:32:41:8C |
| UDP | 1 | 2005-03-30T08:47:51.333000 | 1112172471333401 | 192.168.170.20 | 192.168.170.8 | 53 | 32795 | 00:C0:9F:32:41:8C | 00:E0:18:B1:0C:AD |
| UDP | 1 | 2005-03-30T08:47:59.313000 | 1112172479313231 | 192.168.170.8 | 192.168.170.20 | 32795 | 53 | 00:E0:18:B1:0C:AD | 00:C0:9F:32:41:8C |
| UDP | 1 | 2005-03-30T08:47:59.452000 | 1112172479452255 | 192.168.170.20 | 192.168.170.8 | 53 | 32795 | 00:C0:9F:32:41:8C | 00:E0:18:B1:0C:AD |
| UDP | 1 | 2005-03-30T08:48:07.320000 | 1112172487320873 | 192.168.170.8 | 192.168.170.20 | 32795 | 53 | 00:E0:18:B1:0C:AD | 00:C0:9F:32:41:8C |
| UDP | 1 | 2005-03-30T08:48:07.321000 | 1112172487321379 | 192.168.170.20 | 192.168.170.8 | 53 | 32795 | 00:C0:9F:32:41:8C | 00:E0:18:B1:0C:AD |
| UDP | 1 | 2005-03-30T08:49:18.685000 | 1112172558685951 | 192.168.170.8 | 192.168.170.20 | 32795 | 53 | 00:E0:18:B1:0C:AD | 00:C0:9F:32:41:8C |
| UDP | 1 | 2005-03-30T08:49:18.734000 | 1112172558734862 | 192.168.170.20 | 192.168.170.8 | 53 | 32795 | 00:C0:9F:32:41:8C | 00:E0:18:B1:0C:AD |
| UDP | 1 | 2005-03-30T08:49:35.461000 | 1112172575461181 | 192.168.170.8 | 192.168.170.20 | 32795 | 53 | 00:E0:18:B1:0C:AD | 00:C0:9F:32:41:8C |

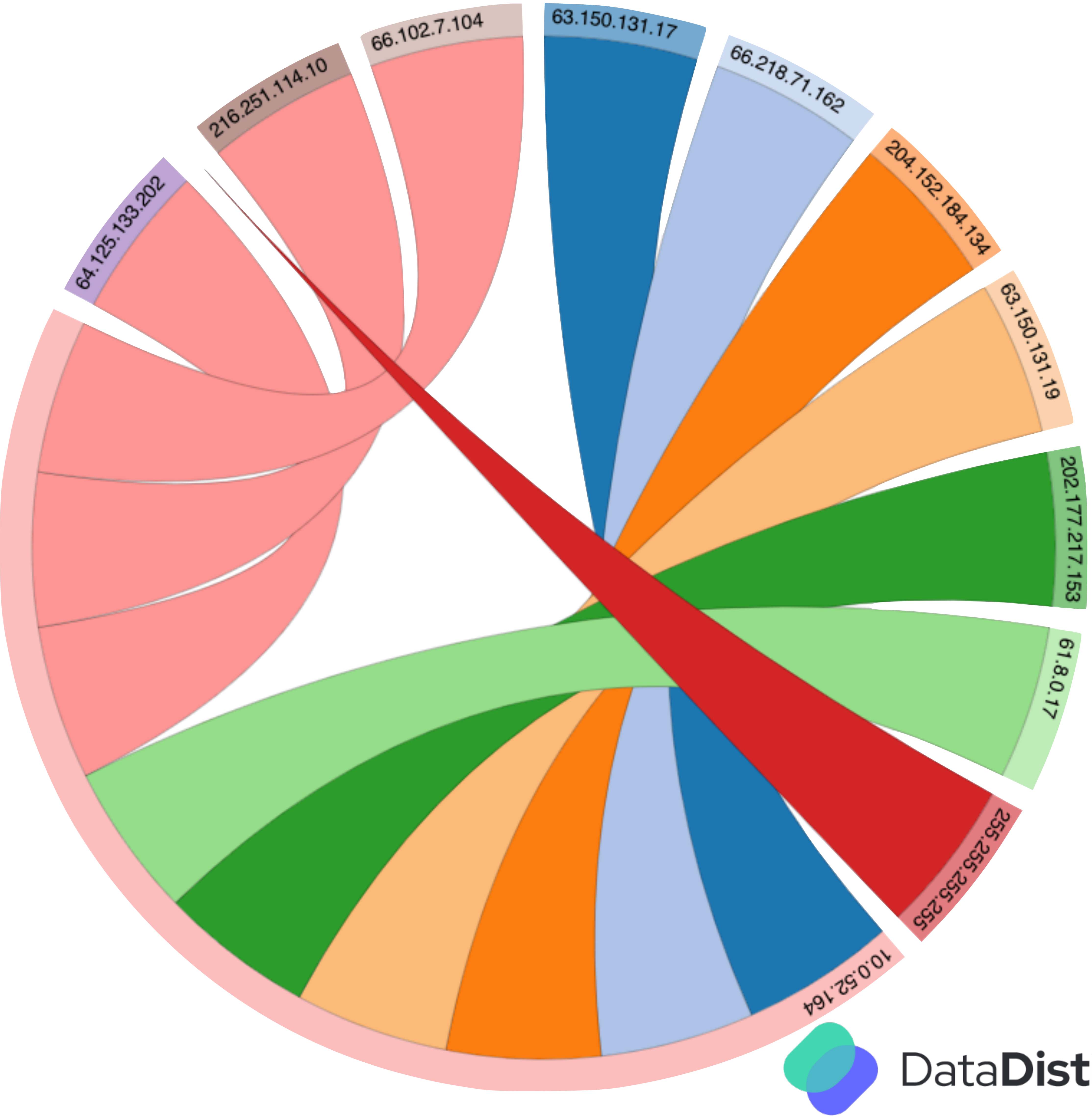
Let's look for who is talking to whom...

# PCAP Analysis

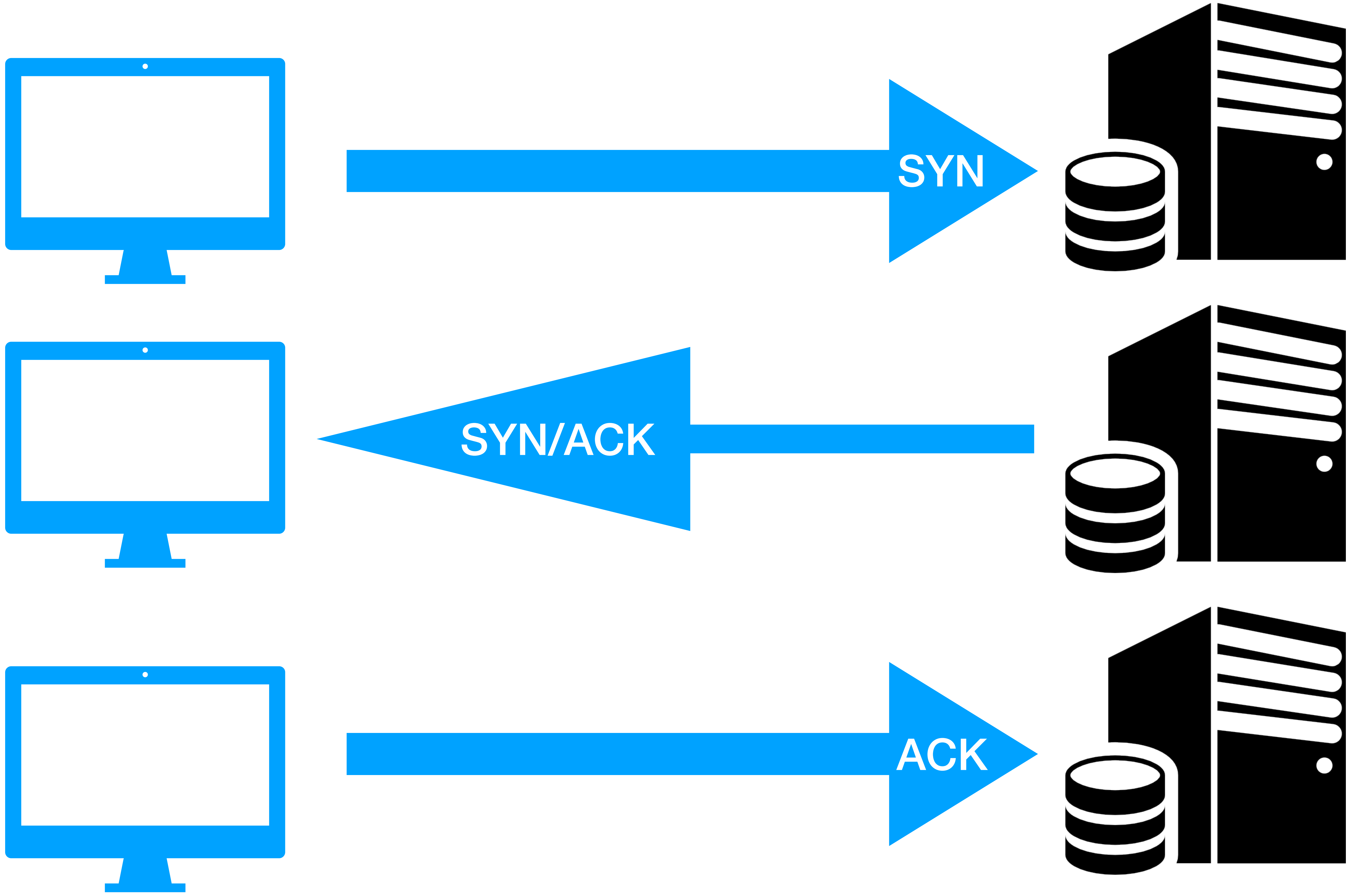
```
SELECT DISTINCT src_ip, dst_ip
FROM dfs.test.`slowdownload.pcap`
WHERE src_ip is not null
```



| src_ip | dst_ip |
|---|---|
| 10.100.252.1 | 255.255.255.255 |
| 10.0.52.164 | 216.251.114.10 |
| 10.0.52.164 4.10 | 10.0.52.164 |
| 10.0.52.164 | 66.218.71.162 |
| 66.218.71.162 | 10.0.52.164 |
| 10.0.52.164 | 66.102.7.104 |
| 66.102.7.104 | 10.0.52.164 |
| 10.0.52.164 | 202.177.217.153 |
| 202.177.217.153 | 10.0.52.164 |
| 10.0.52.164 | 63.150.131.19 |
| 63.150.131.19 | 10.0.52.164 |
| 10.0.52.164 | 63.150.131.17 |

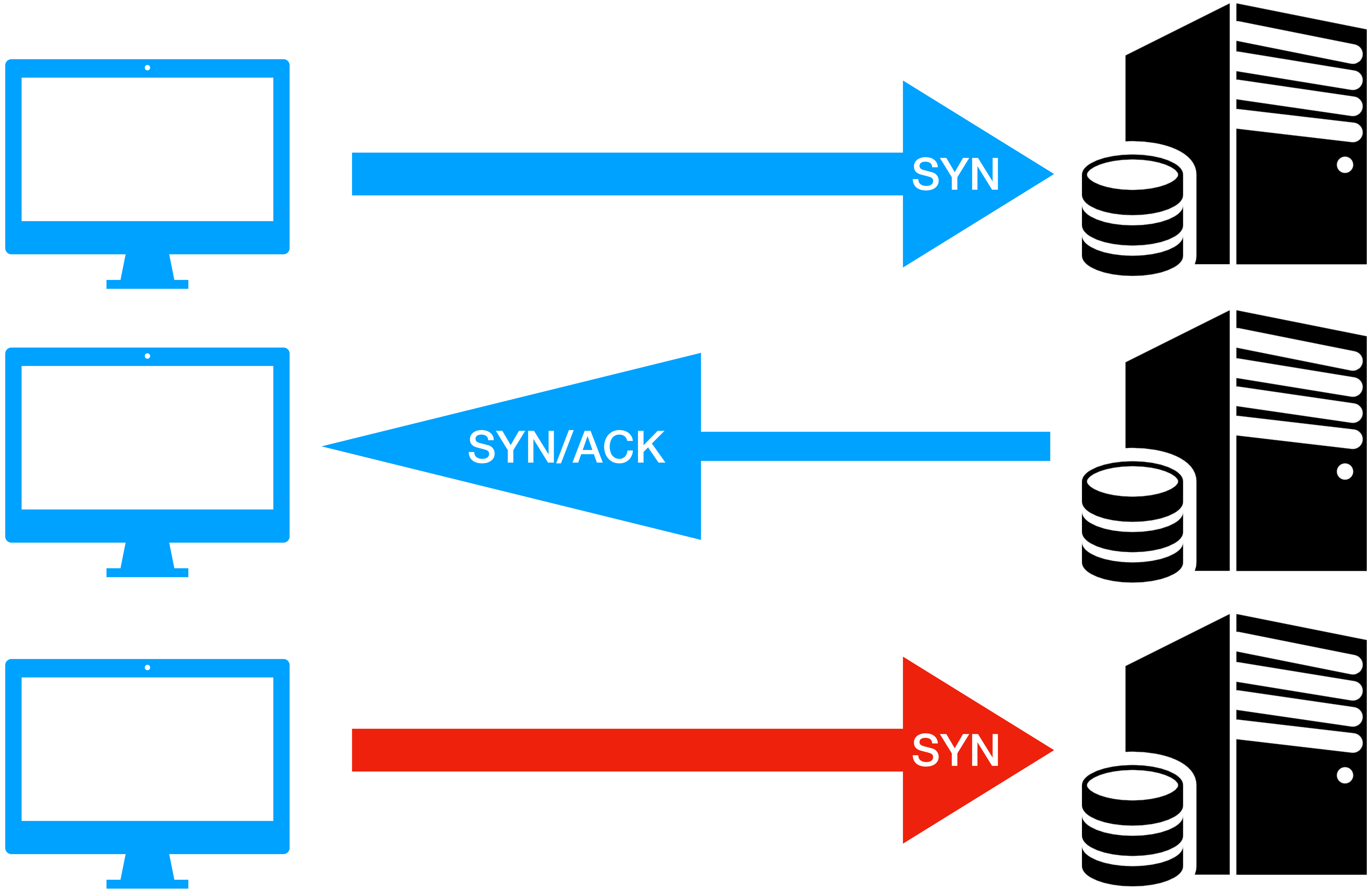


# SynFlood Detection



DataDistillr

# SynFlood Detection



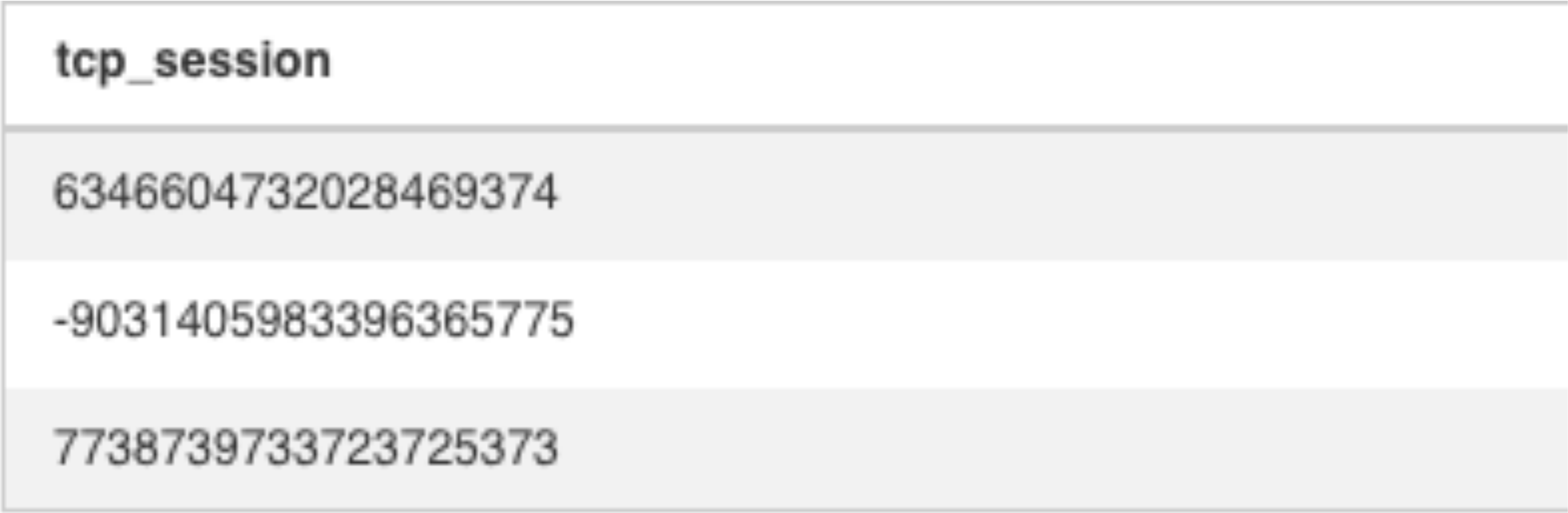
DataDistillr

# Drill can automate SYN Flood detection with a custom UDF.

*https://github.com/cgivre/drill-synflood-udf*

# SYN Flood Detection

```
SELECT tcp_session
FROM dfs.test.`synscan.pcap`
GROUP BY tcp_session
HAVING is_syn_flood(tcp_session, tcp_flags_syn, tcp_flags_ack)
```

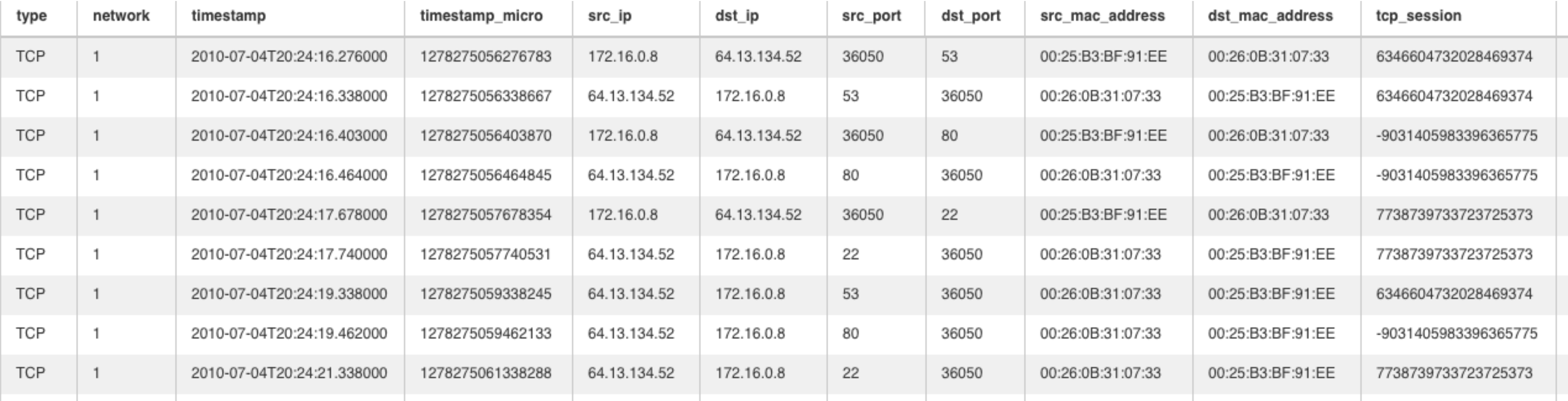


| tcp_session |
|---|
| 6346604732028469374 |
| -9031405983396365775 |
| 7738739733723725373 |

DataDistillr

# SYN Flood Detection

```
SELECT *
FROM dfs.test.`synscan.pcap`
WHERE tcp_session IN
(
   SELECT tcp_session
   FROM dfs.test.`synscan.pcap`
   GROUP BY tcp_session
   HAVING is_syn_flood(tcp_session, tcp_flags_syn, tcp_flags_ack)
)
```



| type | network | timestamp | timestamp_micro | src_ip | dst_ip | src_port | dst_port | src_mac_address | dst_mac_address | tcp_session |
|---|---|---|---|---|---|---|---|---|---|---|
| TCP | 1 | 2010-07-04T20:24:16.276000 | 1278275056276783 | 172.16.0.8 | 64.13.134.52 | 36050 | 53 | 00:25:B3:BF:91:EE | 00:26:0B:31:07:33 | 6346604732028469374 |
| TCP | 1 | 2010-07-04T20:24:16.338000 | 1278275056338667 | 64.13.134.52 | 172.16.0.8 | 53 | 36050 | 00:26:0B:31:07:33 | 00:25:B3:BF:91:EE | 6346604732028469374 |
| TCP | 1 | 2010-07-04T20:24:16.403000 | 1278275056403870 | 172.16.0.8 | 64.13.134.52 | 36050 | 80 | 00:25:B3:BF:91:EE | 00:26:0B:31:07:33 | -9031405983396365775 |
| TCP | 1 | 2010-07-04T20:24:16.464000 | 1278275056464845 | 64.13.134.52 | 172.16.0.8 | 80 | 36050 | 00:26:0B:31:07:33 | 00:25:B3:BF:91:EE | -9031405983396365775 |
| TCP | 1 | 2010-07-04T20:24:17.678000 | 1278275057678354 | 172.16.0.8 | 64.13.134.52 | 36050 | 22 | 00:25:B3:BF:91:EE | 00:26:0B:31:07:33 | 7738739733723725373 |
| TCP | 1 | 2010-07-04T20:24:17.740000 | 1278275057740531 | 64.13.134.52 | 172.16.0.8 | 22 | 36050 | 00:26:0B:31:07:33 | 00:25:B3:BF:91:EE | 7738739733723725373 |
| TCP | 1 | 2010-07-04T20:24:19.338000 | 1278275059338245 | 64.13.134.52 | 172.16.0.8 | 53 | 36050 | 00:26:0B:31:07:33 | 00:25:B3:BF:91:EE | 6346604732028469374 |
| TCP | 1 | 2010-07-04T20:24:19.462000 | 1278275059462133 | 64.13.134.52 | 172.16.0.8 | 80 | 36050 | 00:26:0B:31:07:33 | 00:25:B3:BF:91:EE | -9031405983396365775 |
| TCP | 1 | 2010-07-04T20:24:21.338000 | 1278275061338288 | 64.13.134.52 | 172.16.0.8 | 22 | 36050 | 00:26:0B:31:07:33 | 00:25:B3:BF:91:EE | 7738739733723725373 |

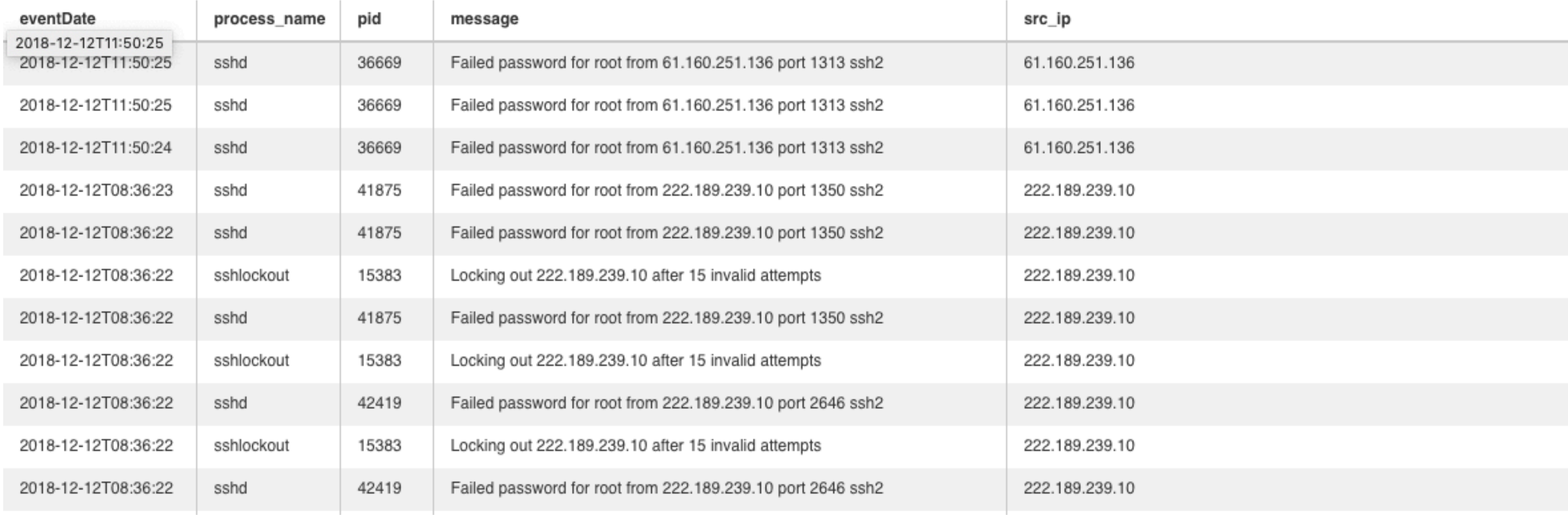
r

# Questions so far?

# How about log files?

DataDistillr

```
Dec 12 2018 06:50:25    sshd[36669]: Failed password for root from 61.160.251.136 port 1313 ssh2
Dec 12 2018 06:50:25    sshd[36669]: Failed password for root from 61.160.251.136 port 1313 ssh2
Dec 12 2018 06:50:24    sshd[36669]: Failed password for root from 61.160.251.136 port 1313 ssh2
Dec 12 2018 03:36:23    sshd[41875]: Failed password for root from 222.189.239.10 port 1350 ssh2
Dec 12 2018 03:36:22    sshd[41875]: Failed password for root from 222.189.239.10 port 1350 ssh2
Dec 12 2018 03:36:22    sshlockout[15383]: Locking out 222.189.239.10 after 15 invalid attempts
Dec 12 2018 03:36:22    sshd[41875]: Failed password for root from 222.189.239.10 port 1350 ssh2
Dec 12 2018 03:36:22    sshlockout[15383]: Locking out 222.189.239.10 after 15 invalid attempts
Dec 12 2018 03:36:22    sshd[42419]: Failed password for root from 222.189.239.10 port 2646 ssh2
```



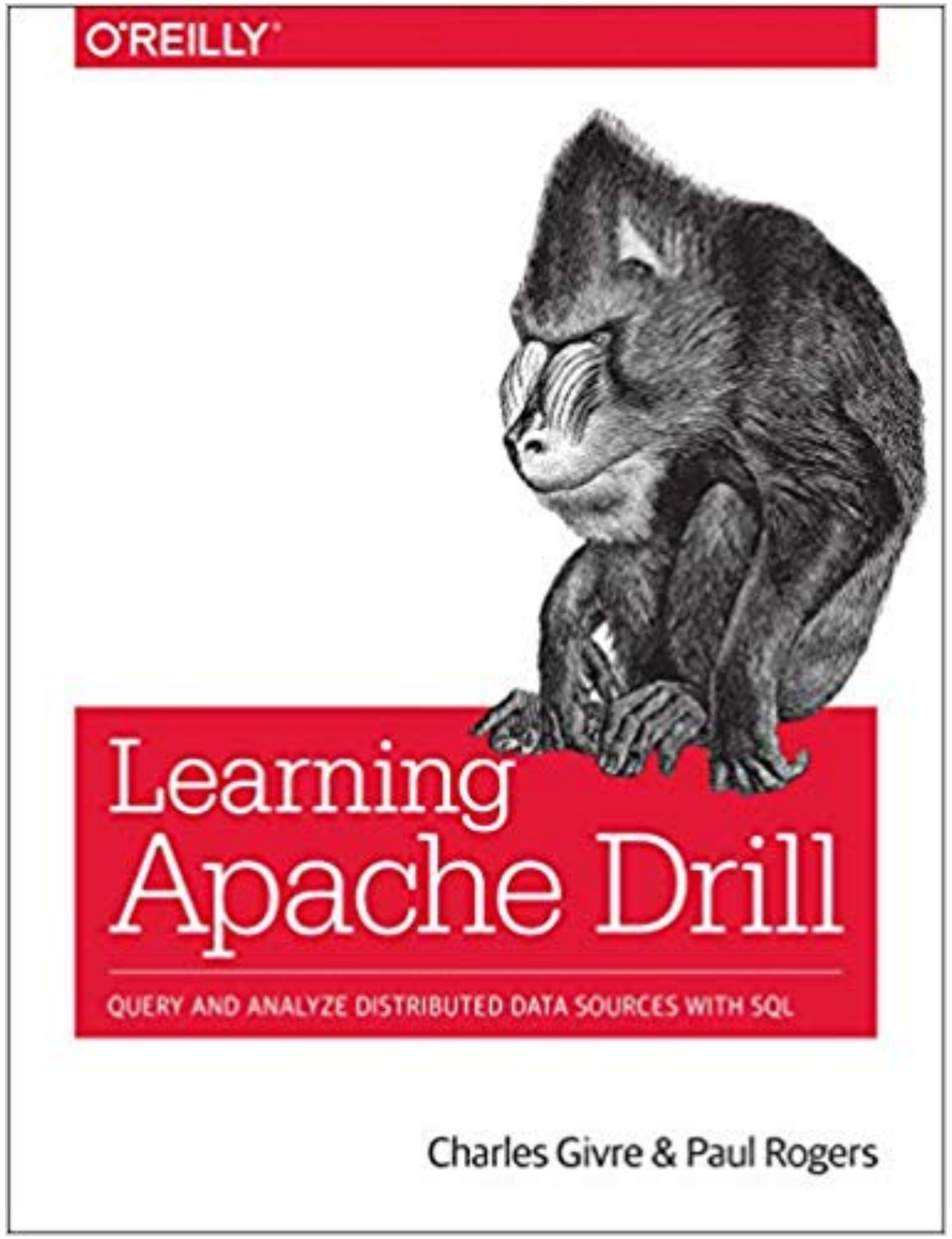
2018-12-12T11:50:25

| eventDate | process_name | pid | message | src_ip |
|---|---|---|---|---|
| 2018-12-12T11:50:25 | sshd | 36669 | Failed password for root from 61.160.251.136 port 1313 ssh2 | 61.160.251.136 |
| 2018-12-12T11:50:25 | sshd | 36669 | Failed password for root from 61.160.251.136 port 1313 ssh2 | 61.160.251.136 |
| 2018-12-12T11:50:24 | sshd | 36669 | Failed password for root from 61.160.251.136 port 1313 ssh2 | 61.160.251.136 |
| 2018-12-12T08:36:23 | sshd | 41875 | Failed password for root from 222.189.239.10 port 1350 ssh2 | 222.189.239.10 |
| 2018-12-12T08:36:22 | sshd | 41875 | Failed password for root from 222.189.239.10 port 1350 ssh2 | 222.189.239.10 |
| 2018-12-12T08:36:22 | sshlockout | 15383 | Locking out 222.189.239.10 after 15 invalid attempts | 222.189.239.10 |
| 2018-12-12T08:36:22 | sshd | 41875 | Failed password for root from 222.189.239.10 port 1350 ssh2 | 222.189.239.10 |
| 2018-12-12T08:36:22 | sshlockout | 15383 | Locking out 222.189.239.10 after 15 invalid attempts | 222.189.239.10 |
| 2018-12-12T08:36:22 | sshd | 42419 | Failed password for root from 222.189.239.10 port 2646 ssh2 | 222.189.239.10 |
| 2018-12-12T08:36:22 | sshlockout | 15383 | Locking out 222.189.239.10 after 15 invalid attempts | 222.189.239.10 |
| 2018-12-12T08:36:22 | sshd | 42419 | Failed password for root from 222.189.239.10 port 2646 ssh2 | 222.189.239.10 |

# Drill can natively query Syslog formatted data*

*** This feature will be available in Drill version 1.16**

# Questions?


O'REILLY
Learning Apache Drill
QUERY AND ANALYZE DISTRIBUTED DATA SOURCES WITH SQL
Charles Givre & Paul Rogers

**https://amzn.to/2HDwE92**

# Drill and DataDistillr

- Drill is a FOSS product. You can download and install at https://drill.apache.org.
- DataDistillr is a commercial SaaS product built on top of Drill.
- DataDistillr runs the bleeding edge version of Drill, so there are some features that are not yet available in the current stable version of Drill.

# Advanced SQL Functionality

# Advanced SQL Functionality

DataDistllr/Drill uses a SQL Dialect that is very similar to MySQL, so if you are comfortable with that, you'll find the learning curve to be minimal. However, there are a few key areas where it diverges significantly:

- Files
- Nested/Complex Data
- Remote Data
- Specialized Functionalities

# Querying Files

- Querying files is fundamentally no different in DataDistillr than querying a database, however the FROM clause is a little different.

- When querying a database, typically the from clause will look like this:

```
FROM <catalog>.<database>.<table>
```

- When you query a file the components are a little different:

```
FROM `<filesystem>`.`<optional workspace>`.`<somefile.csv>`
```

- Note that Drill uses backticks for identifiers.

# Querying Files

- The workspace is an optional shortcut to a file path.
- You can query directories, as well as glob files.

**Examples:**

```
FROM dfs.`/path/to/my/files/file.json`
FROM dropbox.`my_logs/*.log`
FROM box.`my_data/`
```

# Querying Files

- Almost every file format has configurable options which can be set globally in the Drill config or at query time by using the table function.

- The options are different for every format, however documentation is located in the Drill docs: (https://drill.apache.org/docs/querying-a-file-system-introduction/)  or in Drill's github repository: (https://github.com/apache/drill/tree/master/contrib)

- For example, Excel and MS Access files contain more than one table. You can choose which table to query with the following syntax:

```
SELECT <fields>
FROM table(dfs.`test_data.xlsx`(type => 'excel', sheetName => 'secondSheet'))
```

DataDistillr

# Querying Files

- Try it yourself!  Take a few minutes to try querying some of the files in the demo data folder.

# Nested/Complex Data

- One of the fundamental principles of database design is normalization, which states that each table "cell" should be atomic, and thus have only one piece of data in it.

- SQL was originally designed with this in mind.

- Modern SQL systems have extensions to allow users to query nested & complex data.  Unfortunately, the syntax is not standardized across platforms.

- Drill supports two data structures: lists & dictionaries.  AKA arrays and maps.

# Nested/Complex Data: Lists

If you have data that contains a list,  you can access individual elements using bracket notation:

```
SELECT my_list[0] as field_1, my_list[1] AS field_2
```

If you have nested lists, the syntax is the same:

```
SELECT my_nested_list[1][0], my_nested_list[1][2]
```

# Nested/Complex Data: Lists

You can also transform lists or repeated maps using the FLATTEN function.

For instance, if you have a list [1,2,3] called my_list, you could write a query:

```
SELECT FLATTEN(my_list) ...
```

This will create a row for every entry in the list.

# Nested/Complex Data

DataDistillr also supports Key/Value AKA maps AKA associative arrays.

DataDistillr supports two syntax for accessing map elements:

```
SELECT my_map.field
```

OR

```
SELECT my_map['field']
```

My recommendation is to use the bracket notation rather than the dotted notation.

There is also a function KVGEN(), which when used with FLATTEN() can be used to transform maps into columns.

# Nested/Complex Data

- Using the http-pcap.json, try out some queries to extract nested fields. Some questions include:

- What are the different source IP addresses?

- How many different MAC addresses are there?

# Specialized Functionalities

- DataDistillr has many specialized functions for data cleaning and manipulation, not all are OSS.

- You've already seen the IP geolocation functions, there are also functions for working with phone numbers, street addresses and others

# Specialized Functions: URLs

- parse_url(<url>): This function accepts a URL as an argument and returns a map of the URL's protocol, authority, host, and path.
- parse_query( <query_string> ): This function accepts a query string and returns a key/value pairing of the variables submitted in the request.
- Try it out using the url_log.csv file.  See if you can extract all the query parameters as kv pairs

# Specialized Functions: DNS

- getHostName(<IP address>): Returns the host name associated with an IP address.

- getHostAddress(<host>): Returns an IP address associated with a host name.

- dnsLookup(<host>, [<Resolver>]): Performs a DNS lookup on a given host. You can optionally provide a resolver. Possible resolver values are: cloudflare, cloudflare_secondary, google, google_secondary, verisign, verisign_secondary, yandex, yandex_secondary.

- whois(<host>, [<Resolver>]): Performs a whois lookup on the given host name. You can optionally provide a resolver URL. Note that not all providers allow bulk automated whois lookups, so please follow the terms fo service for your provider.

DataDistillr

# Querying Remote Data

- DataDistillr can query remote data VIA API calls.
- DataDistillr will support virtually any API as long as the API returns JSON, CSV or XML.
- DataDistillr supports many different options for APIs including pagination, OAuth 2.0 and more.
- When properly configured, DataDistillr will push down filters to the downstream API.

# Thank You!!

Charles S. Givre, CISSP
charles.givre@gtkcyber.com
www.thedataist.com

DataDistillr