SilkWeb – Analyze silk data through API and Javascript frameworks

Silkweb – Flocon Jan 2017

Copyright and license

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

Presentation agenda

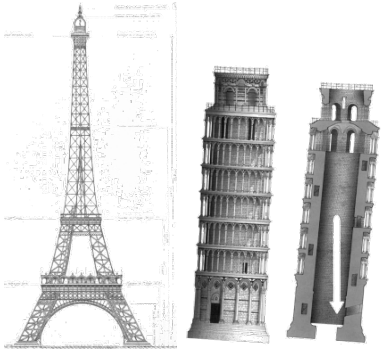
- Introduction and Background
- SilkWeb in a nutshell
- SilkWeb components
- Silkweb in a world of frameworks
- Silk CLI capabilities in SilkWeb
- Use cases from NOC and SOC
- Demo on live data
- Limitations and way forward

Introduction and Background

- Authors : Vijay Sarvepalli & Dwight Beaver

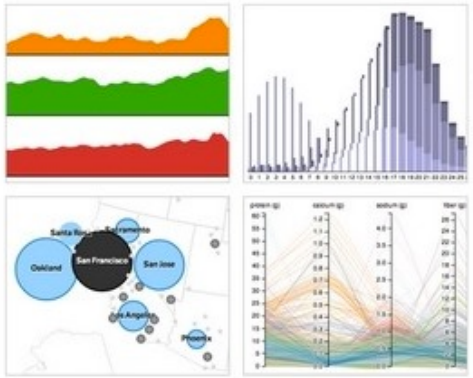
- Sponsors : DOD, DISA
- Collaborators : MPW (ISP)
- Recognition of roles and support

Architecture
== can go either way



Automation
== needs API

Visualization
== lower TTL



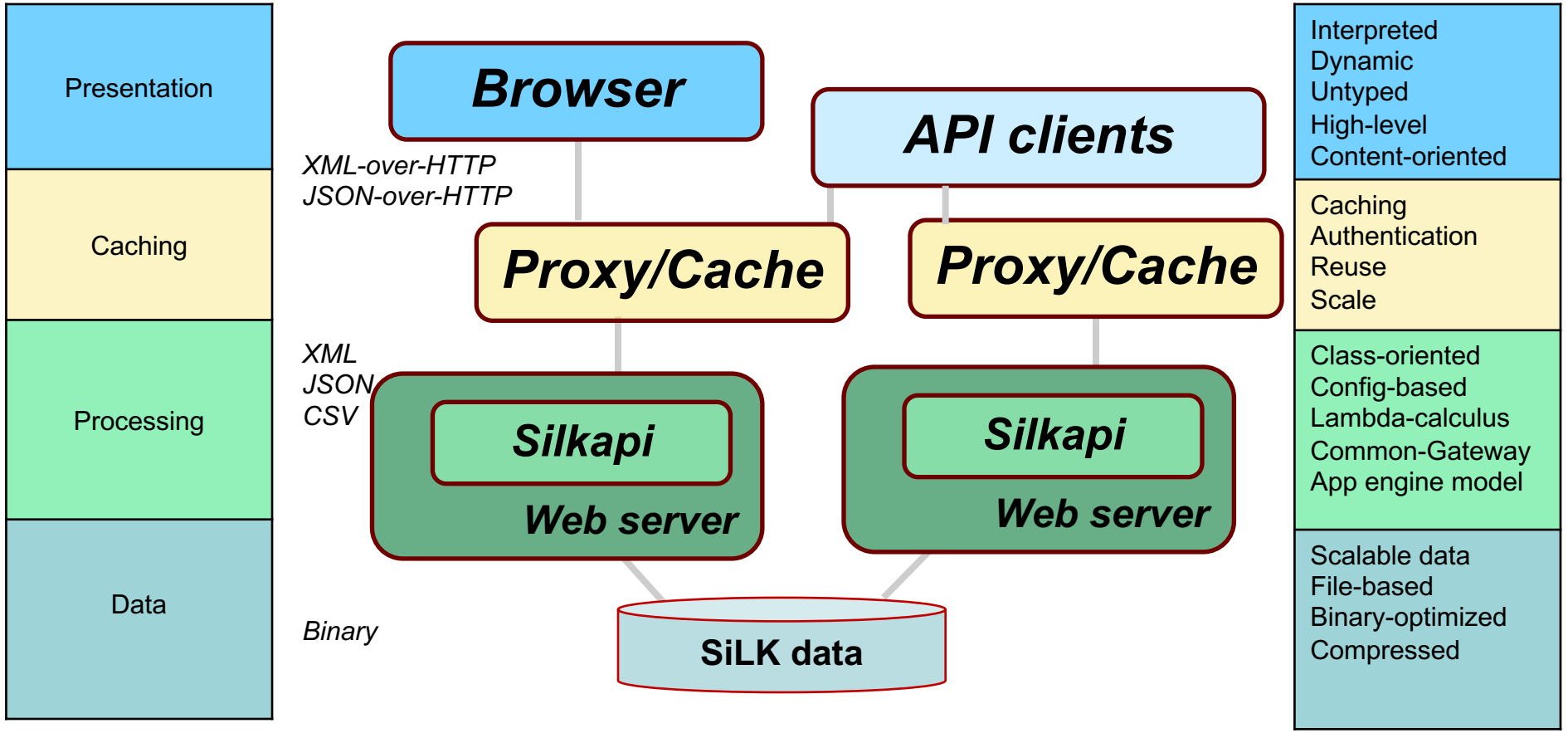
[1] TTL – Time To Learn.

SilkWeb in a nutshell

- SilkWeb is a web application software.
- SilkWeb is designed to simplify access to a SiLK data repository through network data access (JSON/XML over webservices)
- SilkWeb is built with the modern design patterns (AJAX, View-Controller)
- SilkWeb is NOT a standalone web-UI for Silk, it is designed to work with multiple modern software frameworks (SIEM/Dashboards etc.)

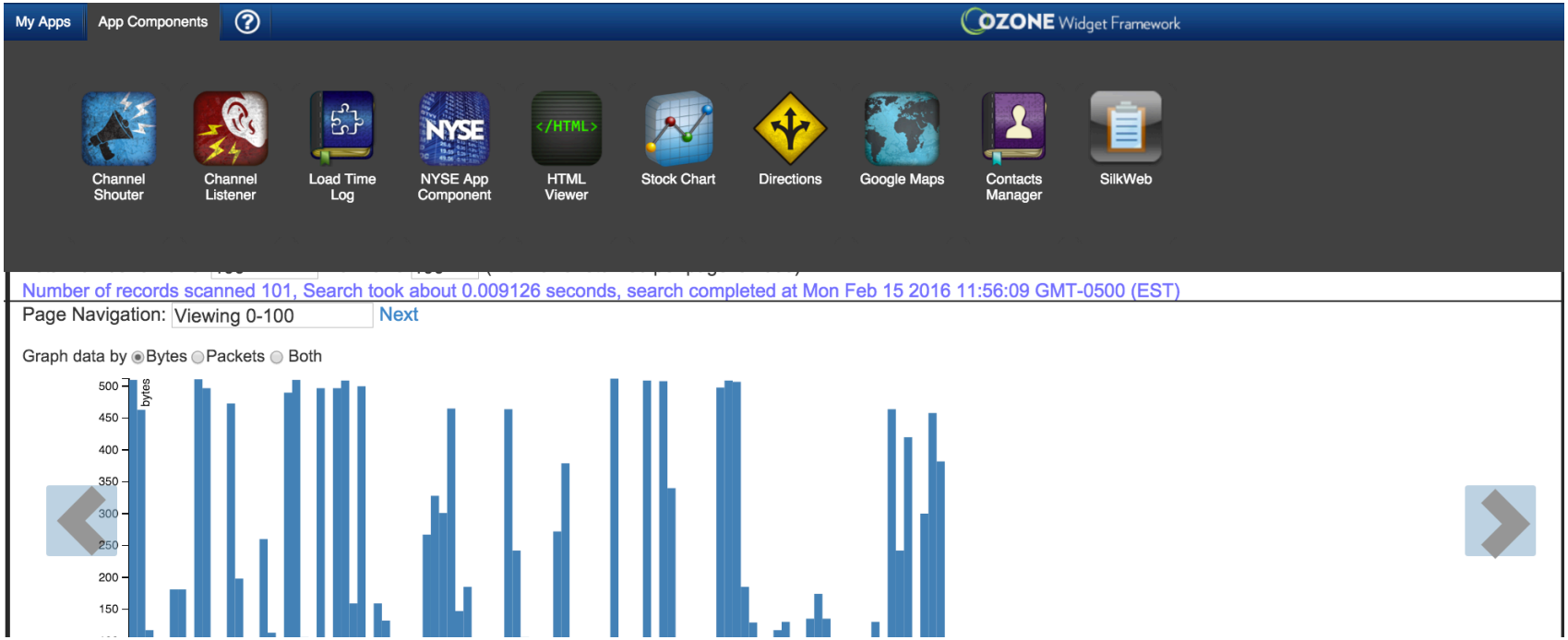
[1] SiLK - *the System for Internet-Level Knowledge built by SEI CERT division*.

Components of SilkWeb (3-tier)



SilkWeb in a world of frameworks

SilkWeb in DISA lab at SEI



- *Modes of integrating SilkWeb to Dashboard*
- *iFrame, IWC widget, component widget, JSON API, XML API*
- *Consideration browser XSS and authentication*

Silk CLI capabilities built into SilkWeb

- Rwfilter simple searches

```
$ rwfilter --type=out,outweb --start-date=2003/02/19                \
        --scidr=10.1.2.0/24 --pass=stdout                           \
```

- Rwstats group by searches

```
$ rwstats --fields=sip --count=4 data.rw
INPUT: 549092 Records for 12990 Bins and 549092 Total Records
OUTPUT: Top 4 Bins by Records
        sIP|     Records|  %Records|   cumul_%|
   10.1.1.1|       36604|  6.666278|  6.666278|
   10.1.1.2|       13897|  2.530906|  9.197184|
   10.1.1.3|       12739|  2.320012| 11.517196|
   10.1.1.4|       11807|  2.150277| 13.667473|
```

- Rwstats with time-bin

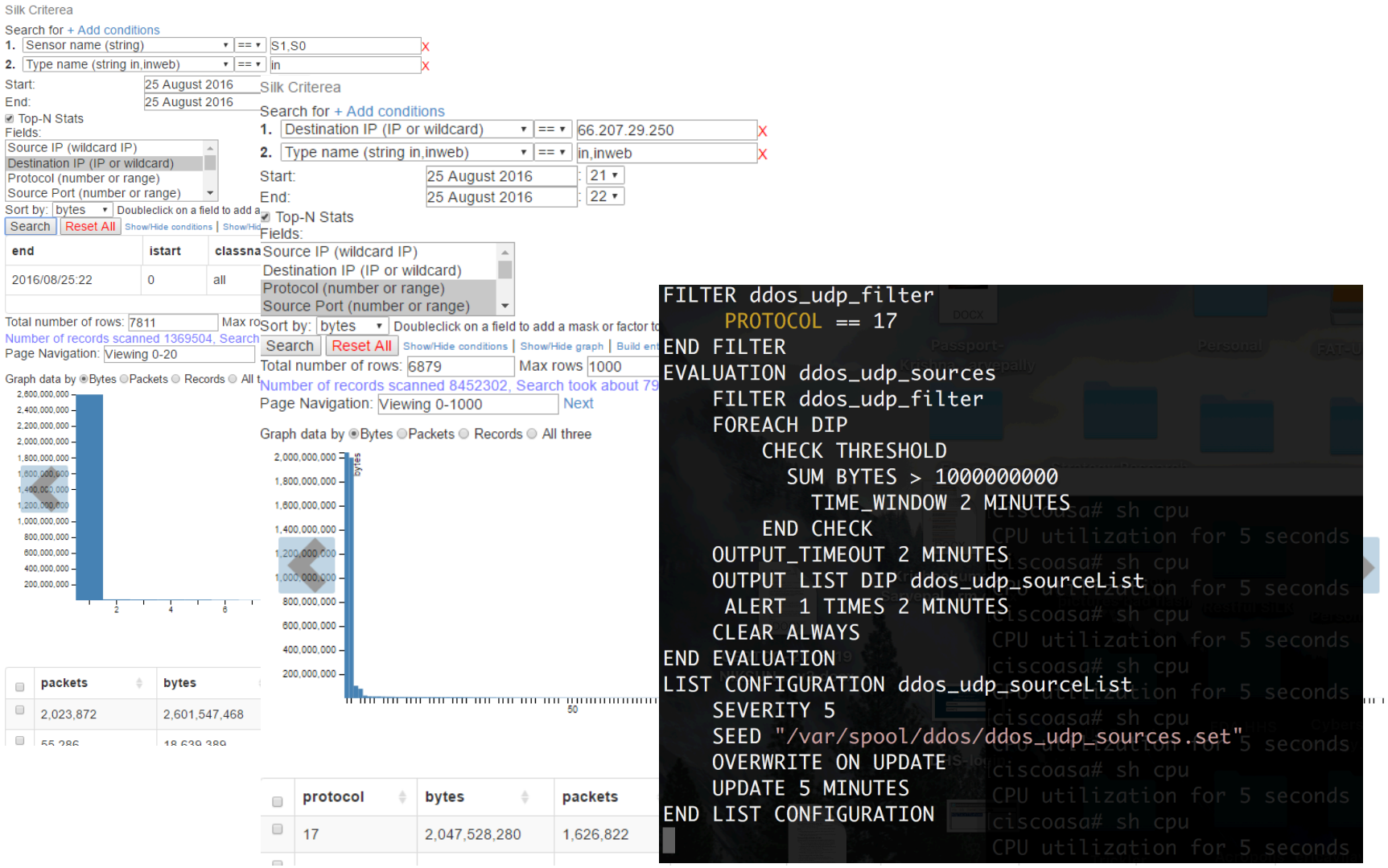
--bin-time
--bin-time=*SECONDS*

Adjust the key fields 'sTime' and 'eTime' to appear on *SECONDS*-second boundaries (the floor of the time is used). When no value is provided to the switch, 60-second time bins are used.

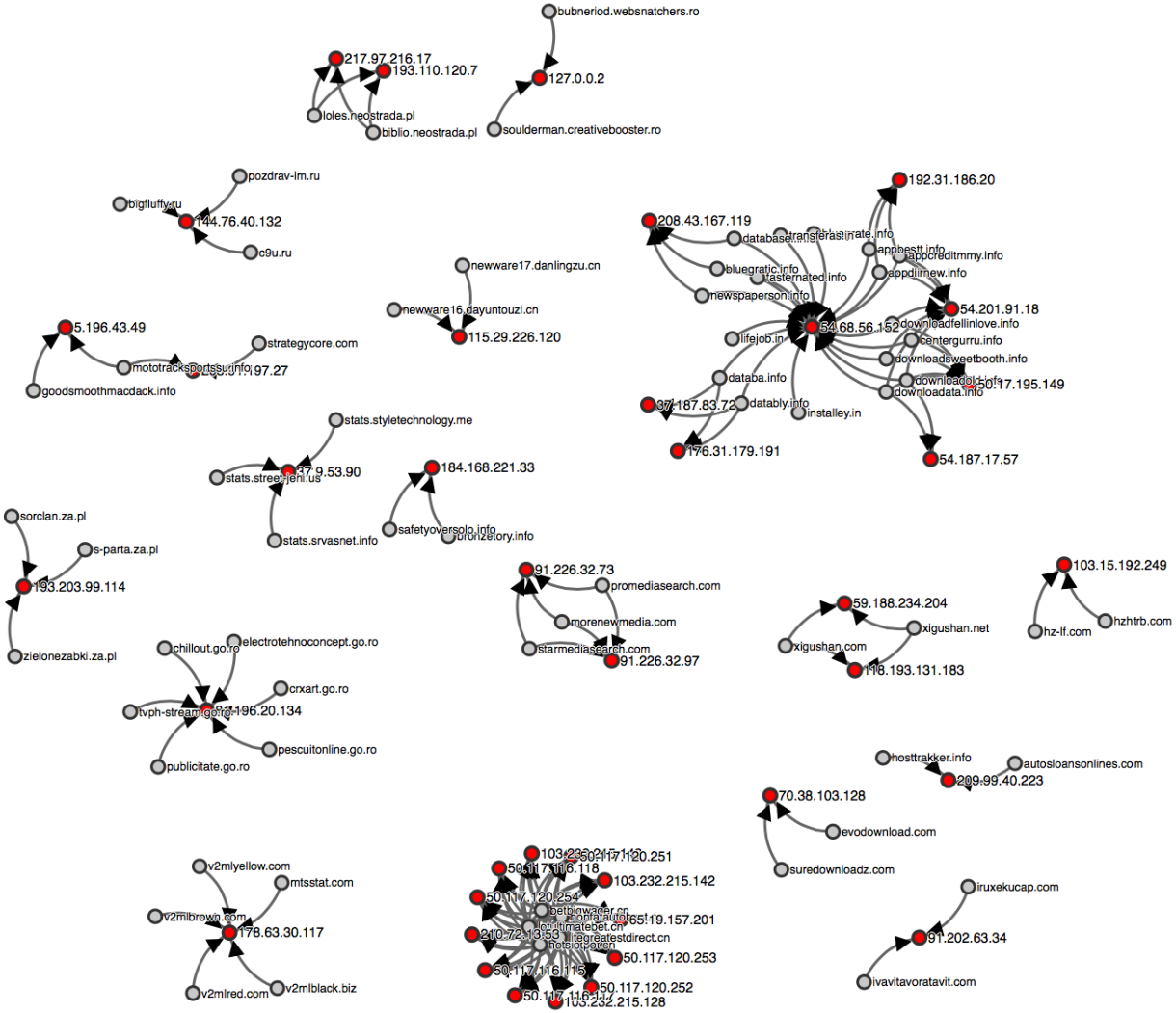
Use cases and live demo

- DDOS workflow (MPW use case)
- Build entity graphs of compromised home routers
- Dyn DDOS analysis
- Building qualifiers to move to “Analysis Pipeline”
- Find unauthorized port/protocols
- Call JSON/XML data from API
- Call JSON data from CLI

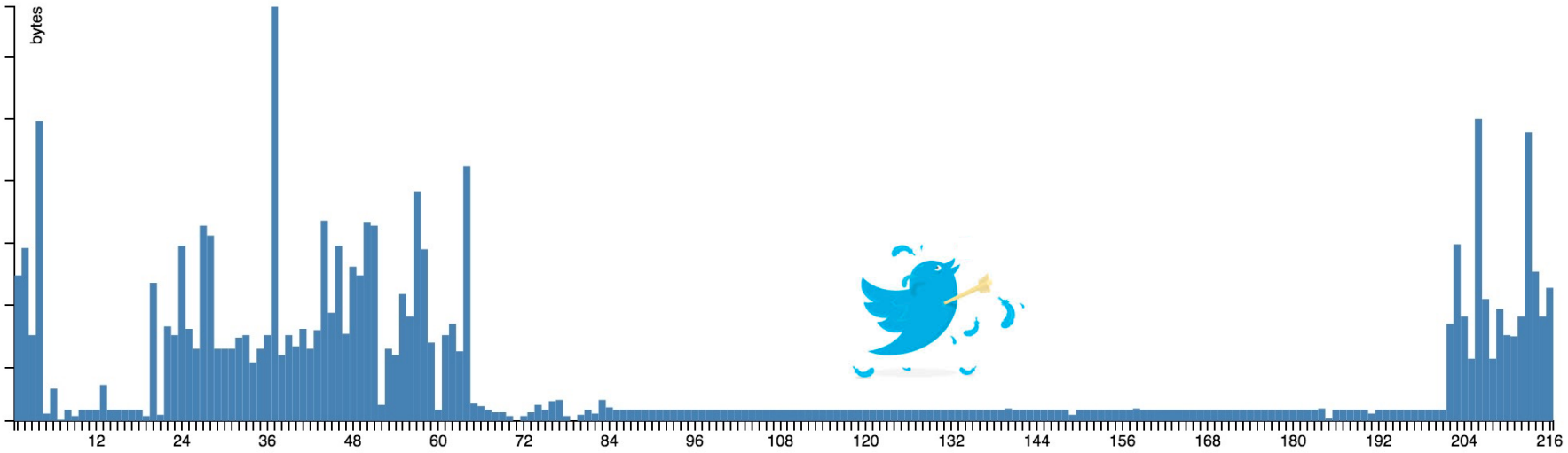
DDOS workflow



Compromised home routers – entity graph

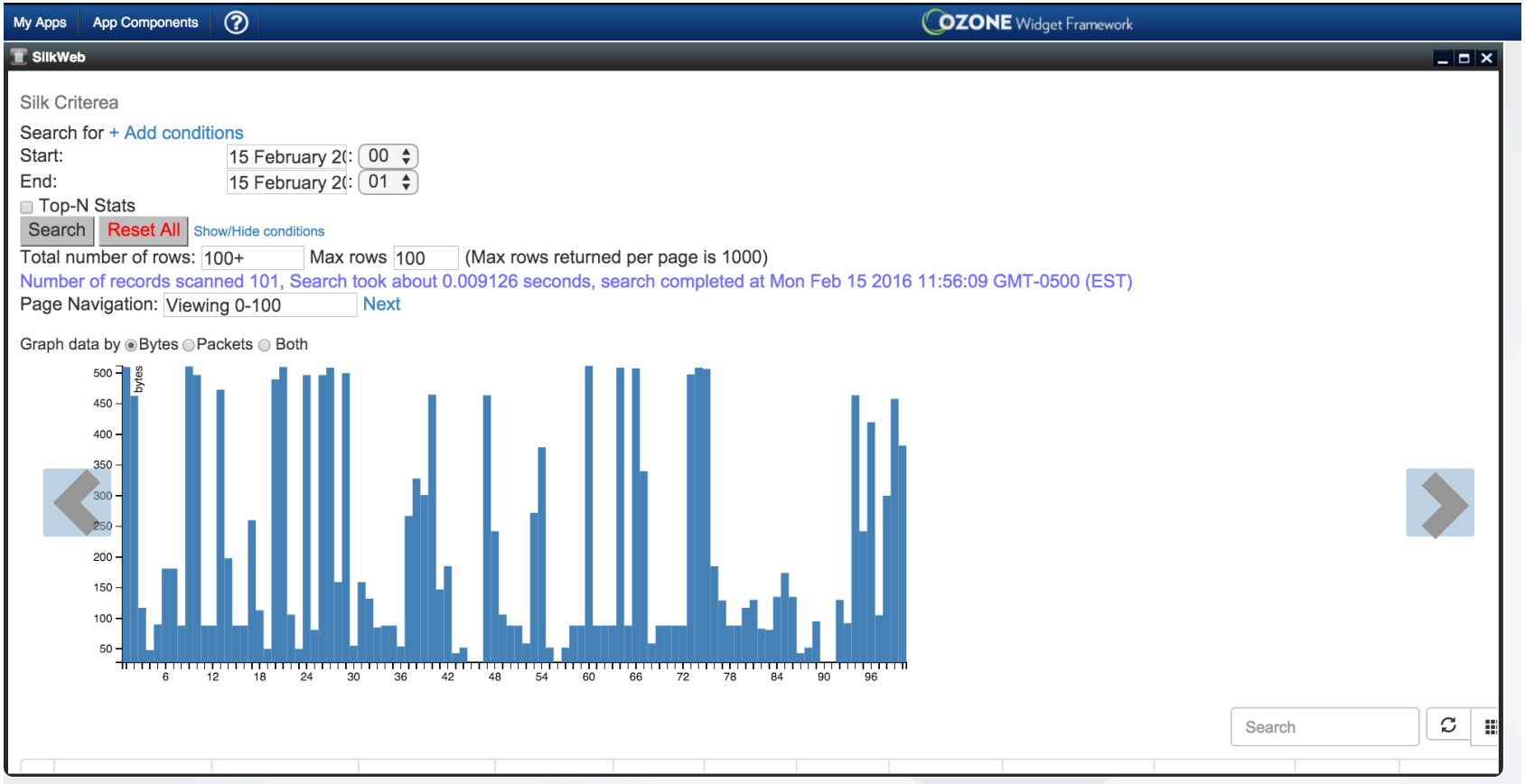


Dyn DDOS analysis

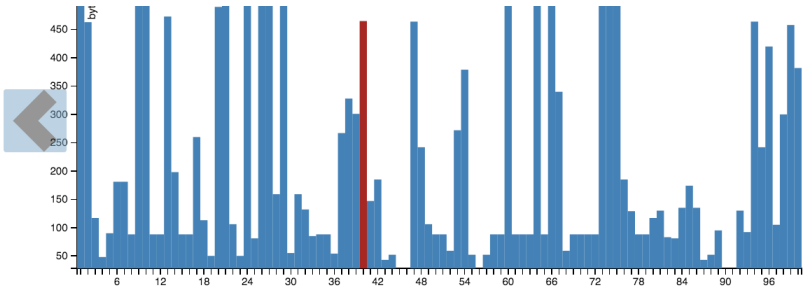


- *The day that your tweets died*

Screenshots basic search



Screenshot pivoting from D3 graph



stime	sip	dip	protocol	sport	dport	bytes	packets	duration secs	application	sensor	typenam
2016-02-15 00:00:56.644000	198.199.94.104	192.168.5.167	17	53	57,037	465	1	0.17	-	asa02	in

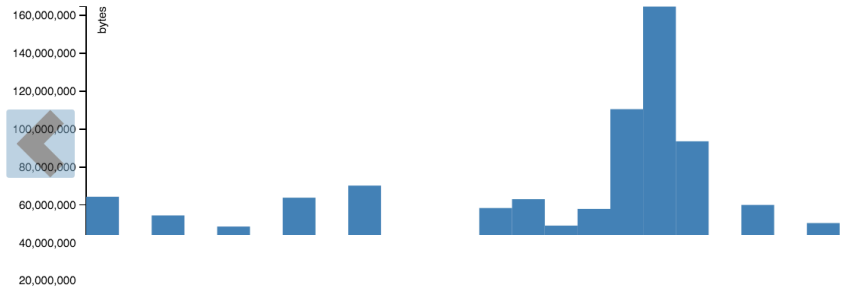
Search

stime	sip	dip	protocol	sport	dport	bytes	packets	duration ...	application	sensor	typename
2016-02-15 00:00:07.244000	104.131.10.18	192.168.5.153	17	53	60,537	510	1	0.03	-	asa02	in
2016-02-15 00:00:07.734000	198.199.94.104	192.168.5.168	17	53	61,438	463	1	0.15	-	asa02	in

Demo of Stats and summary by time

Top-N Stats
Fields:
End Time (seconds)
Start Time (seconds)
Start time/300
End Time
Sort by: time Doubleclick on a field to add a mask or factor to group by
Search Reset All Show/Hide conditions
Total number of rows: 24 Max rows 100 (Max rows returned per page is 1000)
Number of records scanned 21059, Search took about 1.768839 seconds, search completed at Mon Feb 15 2016 11:59:42 GMT-0500 (EST)
Page Navigation: Viewing 0-100

Graph data by Bytes Packets Records All three



173.194.204

stime	sip	dip	protocol	sport	dport	bytes	packets	duration ...	application	sensor	ty
2016-02-15 00:21:18.854000	173.194.204.95	192.168.5.166	6	443	49,640	6,315	1	11.16	-	asa02	in
2016-02-15 00:21:29.834000	173.194.204.95	192.168.5.166	6	443	49,644	7	1	1.46	-	asa02	in
2016-02-15 00:21:31.674000	173.194.204.95	192.168.5.166	6	443	49,648	4,005	1	33.29	-	asa02	in
2016-02-15 00:21:30.744000	173.194.204.95	192.168.5.166	6	443	49,646	11,332	1	243.77	-	asa02	in

JSON/XML API for other widgets to consume


```
{"gdata": [{"bytes": 64332509, "packets": 1128, "rowid": 1, "records": 1128, "stime/300": "2016-02-15 01:55:00-
2016-02-15 02:00:00"}, {"bytes": 43597295, "packets": 1262, "rowid": 2, "records": 1262, "stime/300": "2016-02-
15 01:50:00-2016-02-15 01:55:00"}, {"bytes": 54490113, "packets": 1055, "rowid": 3, "records": 1055,
"stime/300": "2016-02-15 01:45:00-2016-02-15 01:50:00"}], "rows": "3", "query_conditions": {"end":
"2016/02/15:01", "istart": "0", "out_type": "json", "classname": "all", "start": "2016/02/15:00", "sortby":
"time", "stats": "stime/300", "sensors": ["asa02", "kansascity", "squid", "squidkc"], "iend": "3", "types":
["in", "inweb", "inicmp"]}, "stats_totals": {"packets": 21059, "length": 24, "bytes": 1323440288, "records":
21059}, "header": {"timestamp": "1455555636", "version": "1.57", "time_execution": "1.936805 seconds"},
"rows_searched": 21059}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<o>
<header><timestamp>1455555682</timestamp><version>1.57</version><time_execution>1.823312
seconds</time_execution></header>
<query_conditions><end>2016/02/15:01</end><istart>0</istart><out_type>xml</out_type>
<classname>all</classname><start>2016/02/15:00</start><sortby>time</sortby><stats>stime/300</stats><sensors>
(u'asa02', u'kansascity', u'squid', u'squidkc')</sensors><iend>3</iend><types>(u'in', u'inweb', u'inicmp')
</types></query_conditions>
<gdata class="array">
<record><bytes>64332509</bytes><packets>1128</packets><rowid>1</rowid><records>1128</records><stime:300>2016-
02-15 01:55:00-2016-02-15 02:00:00</stime:300></record>
<record><bytes>43597295</bytes><packets>1262</packets><rowid>2</rowid><records>1262</records><stime:300>2016-
02-15 01:50:00-2016-02-15 01:55:00</stime:300></record>
<record><bytes>54490113</bytes><packets>1055</packets><rowid>3</rowid><records>1055</records><stime:300>2016-
02-15 01:45:00-2016-02-15 01:50:00</stime:300></record>
</gdata>
<stats_totals><packets>21059</packets><length>24</length><bytes>1323440288</bytes><records>21059</records>
</stats_totals>
<rows>3</rows>
<rows_searched>21059</rows_searched>
</o>
```

Limitations and Way forward

- JSON/XML is noisy throttle and use wisely
- Test with command line and understand limitations
- Be careful with calculus
  - In-memory IPSets are used in lambda functions
- Move to your graphics platform once you understand D-3
- Use asynchronous to keep user engaged not to fool the analysis.
- Try it!