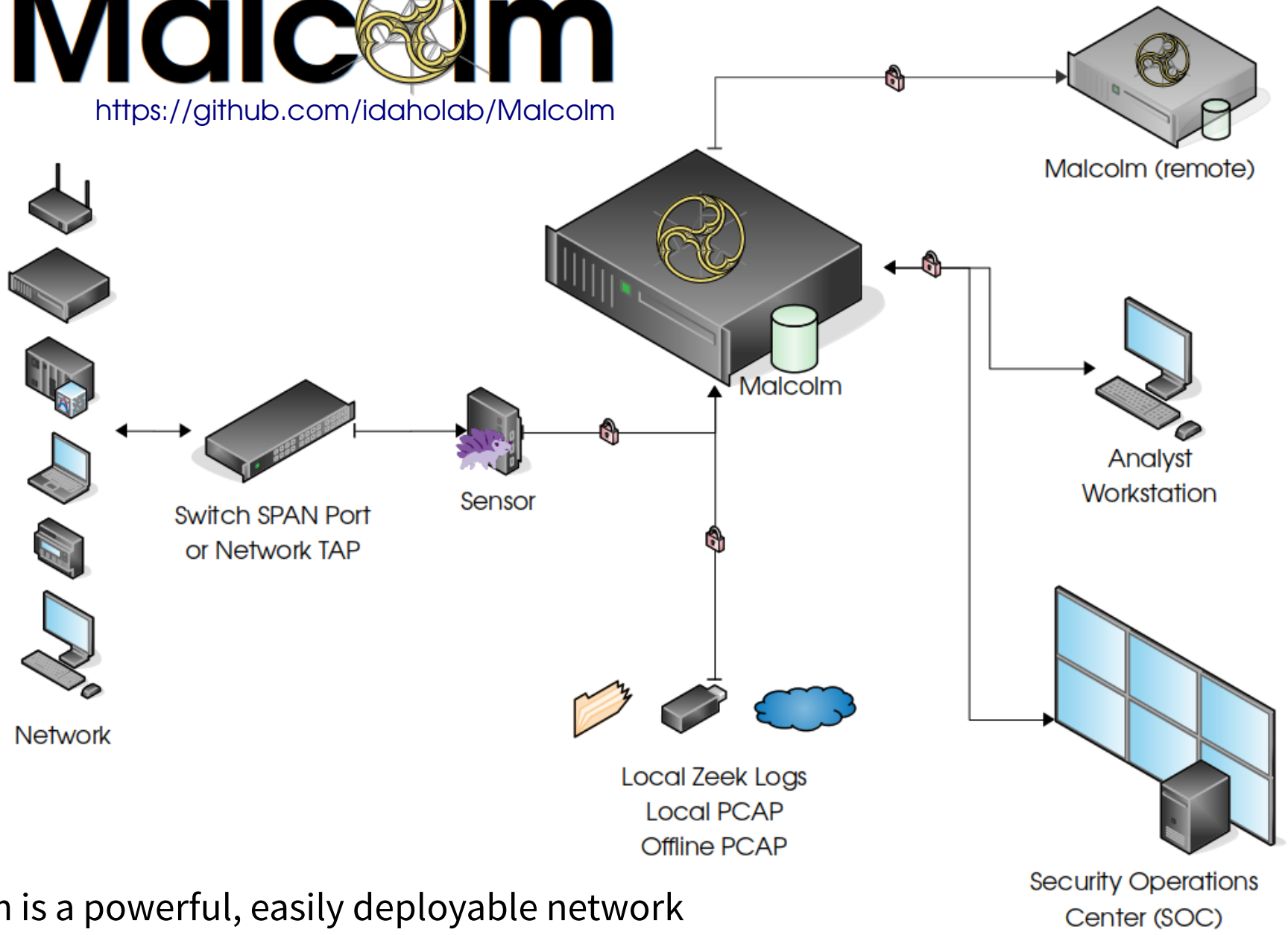
# Network Traffic Analysis with


https://github.com/idaholab/Malcolm

# Malcolm

https://github.com/idaholab/Malcolm




Malcolm is a powerful, easily deployable network traffic analysis tool suite for full packet capture artifacts (PCAP files) and Zeek logs.

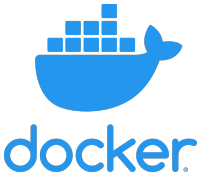
# Malcolm leverages industry-standard open source tools, including:


Elastic Stack




CyberChef







… and more!

## Supported Protocols

Link and Internet Layers
Border Gateway Protocol (BGP)
**Building Automation and Control (BACnet)**
Distributed Computing Env. / Remote Procedure Calls (DCE/RPC)
Dynamic Host Configuration Protocol (DHCP)
**Distributed Network Protocol 3 (DNP3)**
Domain Name System (DNS)
**EtherNet/IP / Common Industrial Protocol (CIP)**
FTP (File Transfer Protocol)
Google Quick UDP Internet Connections (gQUIC)
Hypertext Transfer Protocol (HTTP)
Internet Relay Chat (IRC)
Kerberos
Lightweight Directory Access Protocol (LDAP)
**Modbus**
**MQ Telemetry Transport (MQTT)**
MySQL
NT Lan Manager (NTLM)
Network Time Protocol (NTP)
Oracle
PostgreSQL
**Process Field Net (PROFINET)**
Remote Authentication Dial-In User Service (RADIUS)
Remote Desktop Protocol (RDP)
Remote Framebuffer (RFB)
**S7comm / Connection Oriented Transport Protocol (COTP)**
Session Initiation Protocol (SIP)
Server Message Block (SMB) / Common Internet File System (CIFS)
Simple Mail Transfer Protocol (SNMP)
Simple Network Management Protocol (SMTP)
SOCKS
Secure Shell (SSH)
Secure Sockets Layer (SSL) / Transport Layer Security (TLS)
Syslog
**Tabular Data Stream (TDS)**
Tunnel protocols (e.g., WireGuard, GTP, GRE, Teredo, AYIYA, IP-in-IP, …)

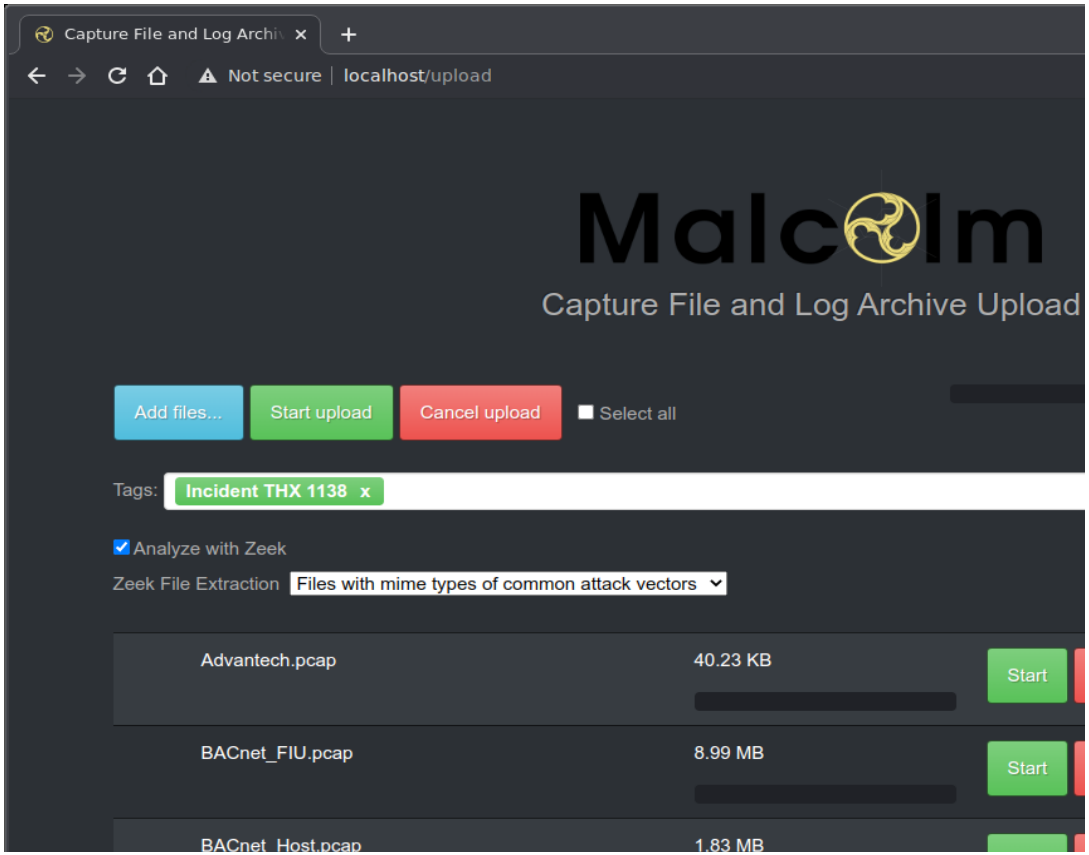
Common ICS protocols are listed in **bold**

# Installation and configuration

- Runs on any OS with Docker (Linux, macOS, Windows)
- Alternately, Linux-based Malcolm OS can be installed on hardware or in a virtual machine
- Minimum requirements: 12+ GB RAM, 4+ CPU cores, 8 GB storage + “enough” storage for traffic and logs
- Read
  - https://github.com/idaholab/Malcolm#QuickStart
- Watch
  - Malcolm Network Traffic Analysis Tool Suite 

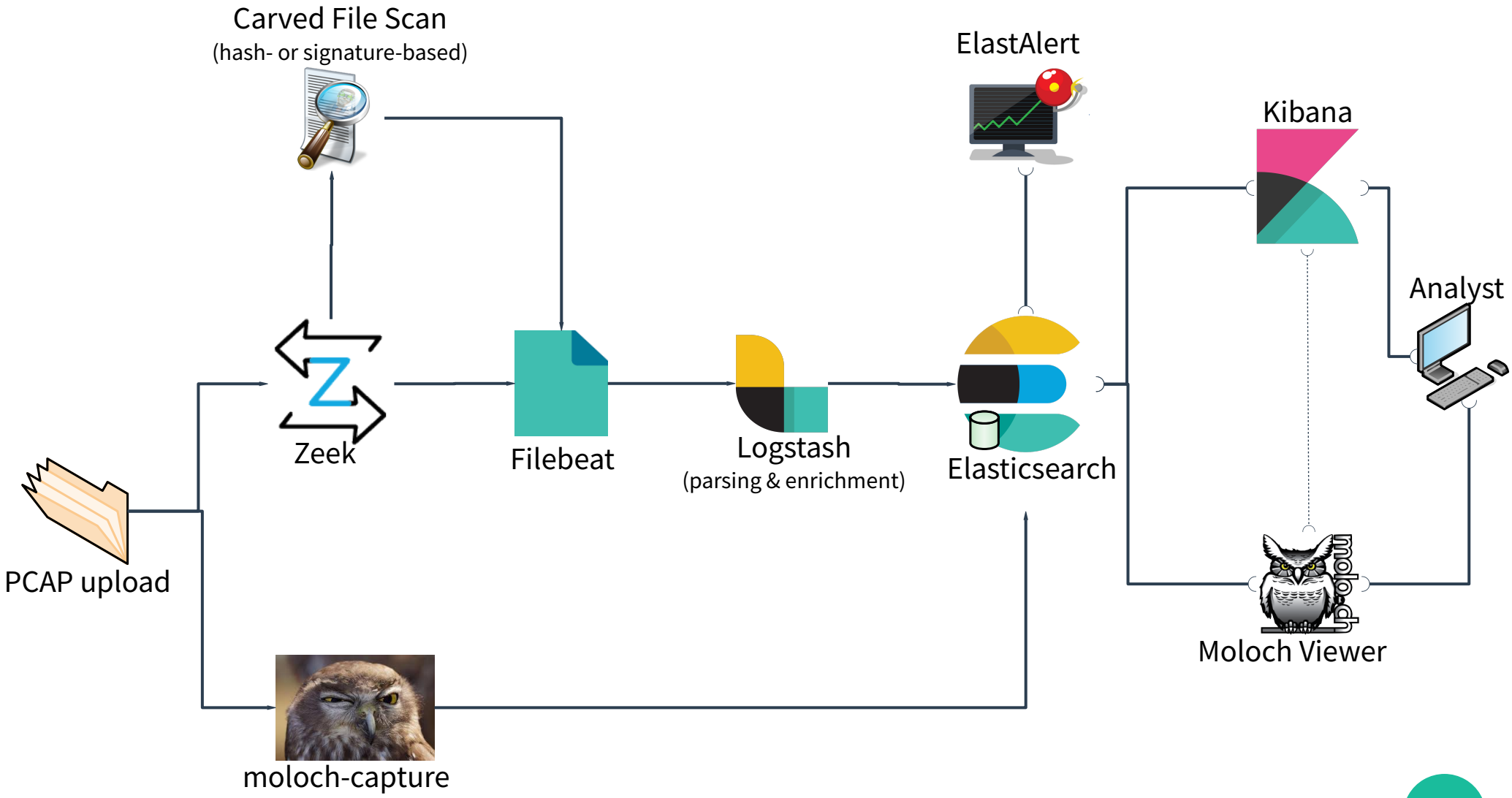
# Artifact upload

- https://localhost/upload once Malcolm is running*
- Apply searchable tags
- Enable Zeek analysis and file extraction
  - Can also be enabled in global defaults
- Upload PCAP files or archived Zeek logs
  - pcapng not supported yet




*Links and examples assume Malcolm is accessible on localhost, YMMV depending configuration

# Malcolm PCAP processing pipeline

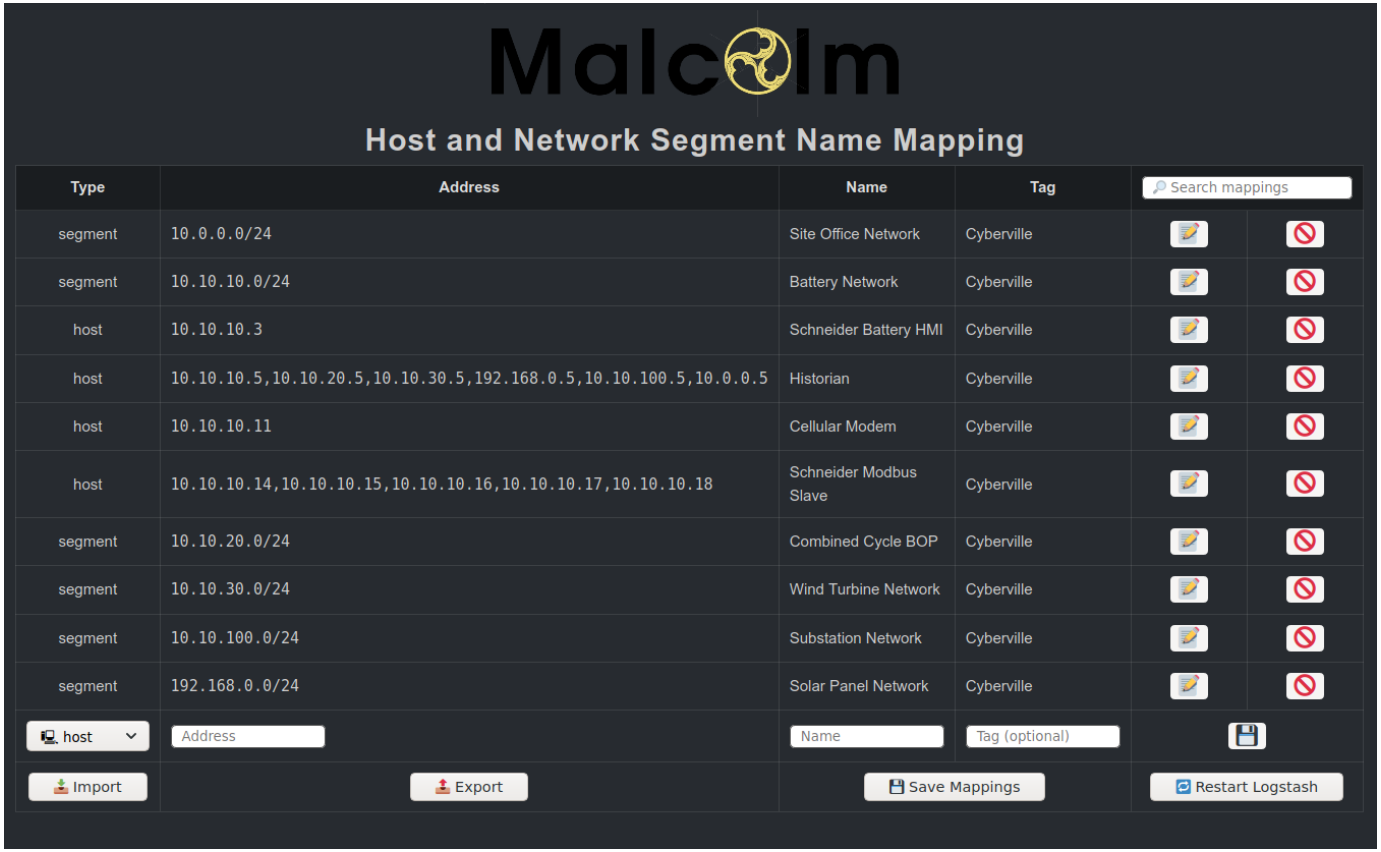


# Log enrichment

- **Logstash enriches Zeek log data**
  - MAC addresses to hardware vendor
  - GeoIP and ASN lookups
  - Internal/external traffic based on IP ranges
  - Reverse DNS lookups
  - DNS query and hostname entropy analysis
  - Connection fingerprinting (JA3 for TLS, HASSH for SSH, Community ID for flows)

# Custom host and subnet name assignment

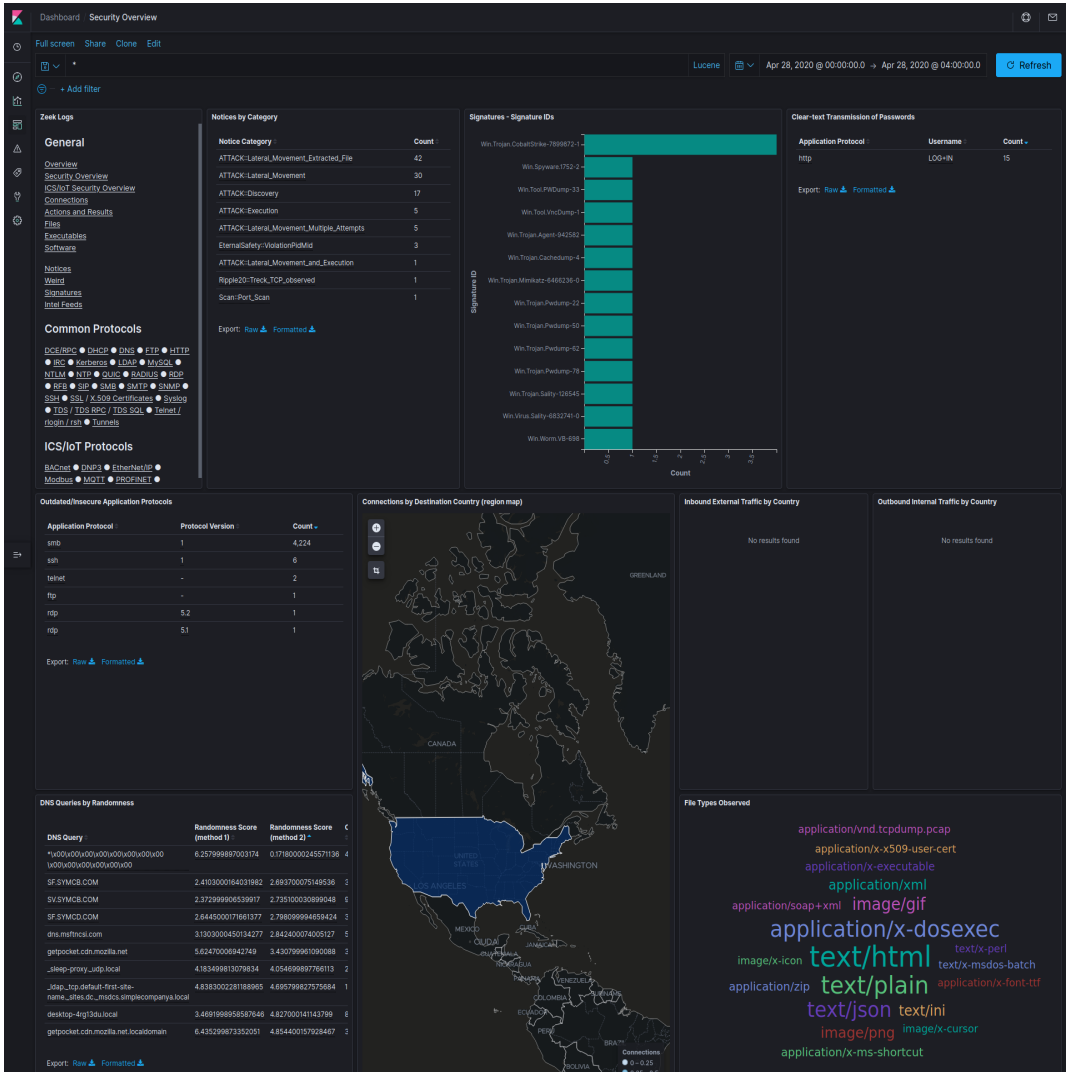
- https://localhost/name-map-ui
- Assign **custom names** to network hosts and segments

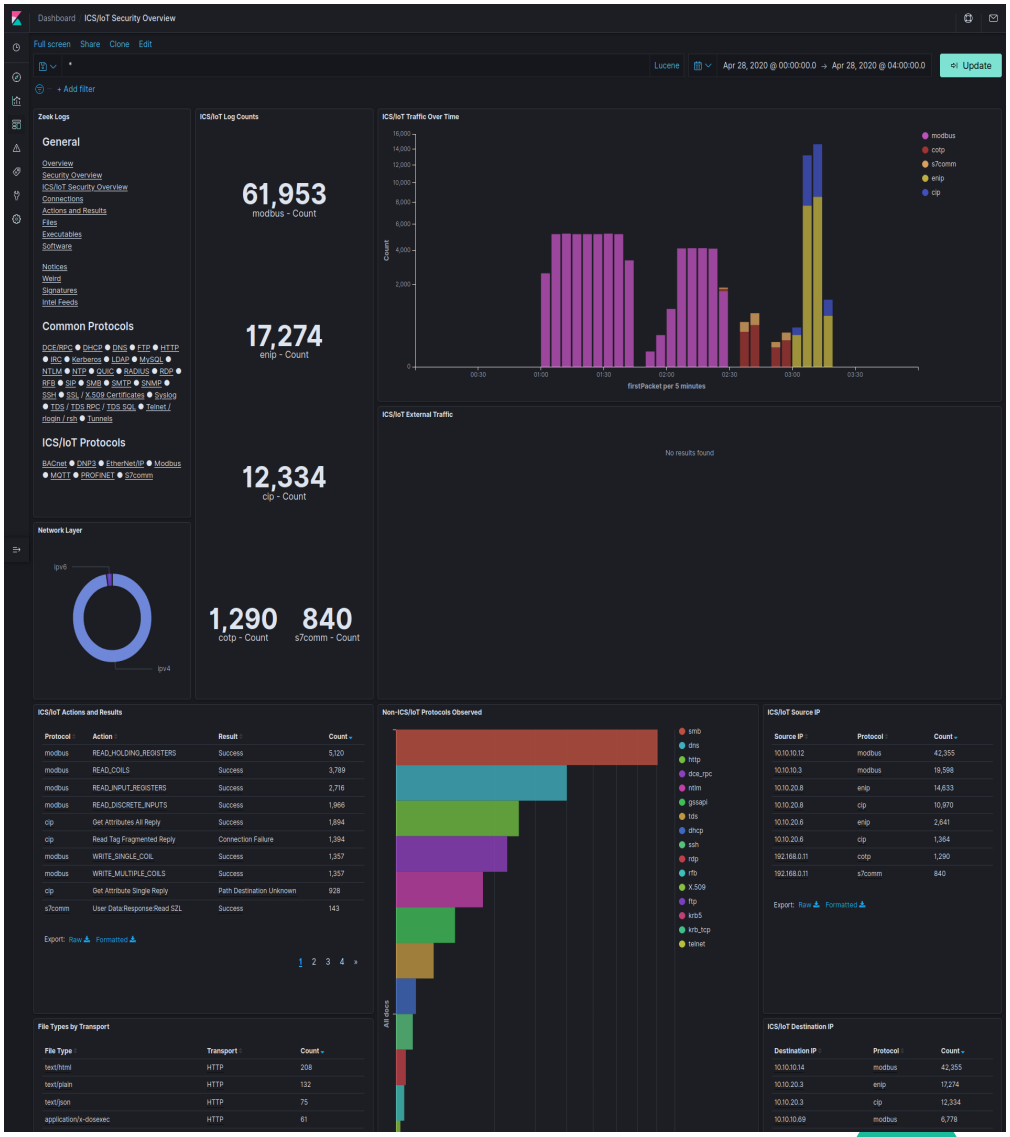


# Kibana

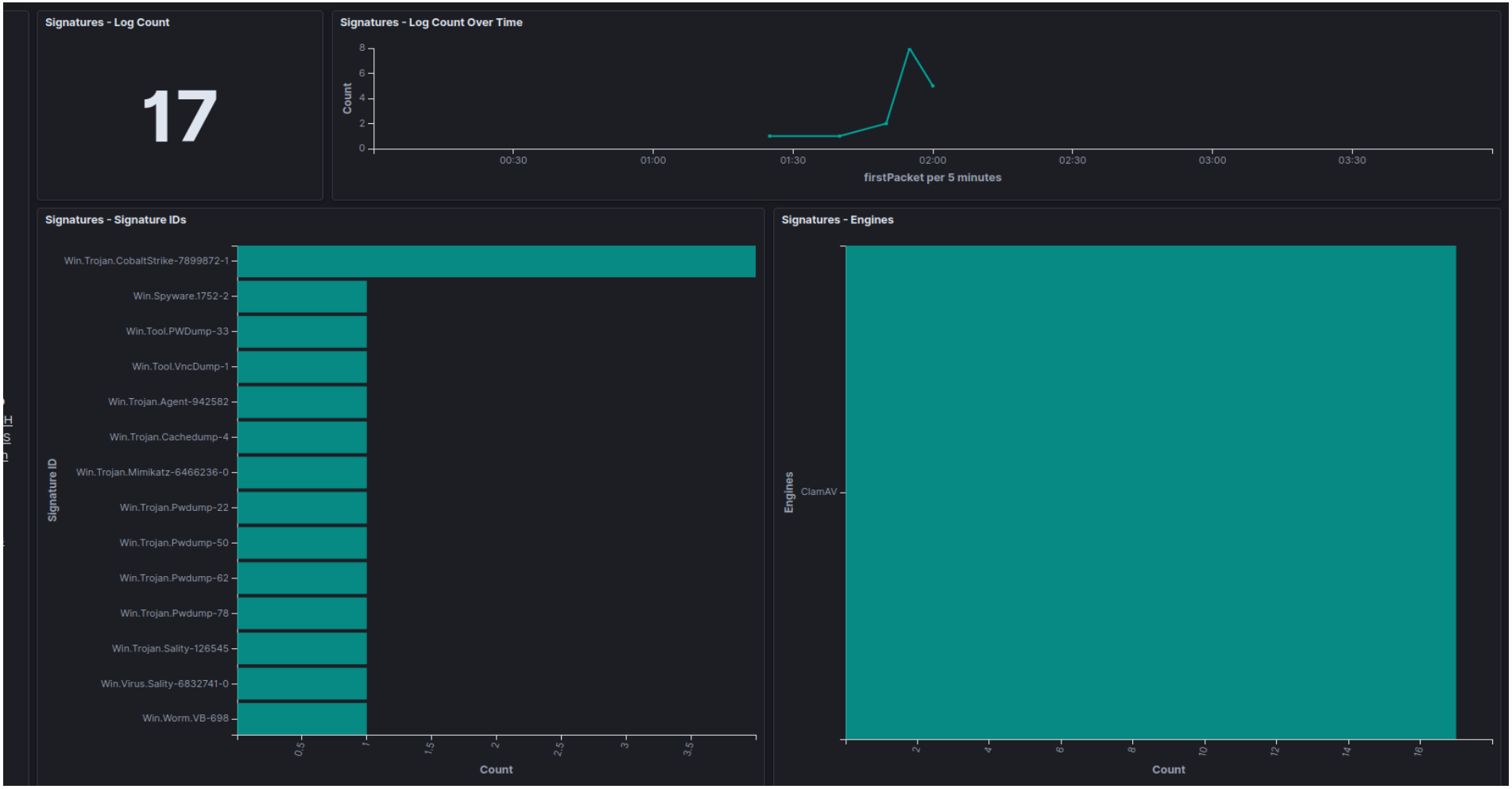
- **https://localhost/kibana**
- **Front end for enriched Zeek logs**
- **Search with Kibana Query Language or Lucene Query Syntax**
- **Dashboards**
  - Overviews: Overview, Security Overview, ICS/IoT Security Overview, Connections, Actions and Results, Files, Executables, Software, Notices, Weird and Signatures
  - Protocol-specific: prebuilt for all protocols Malcolm understands
  - WYSIWYG editors to create custom visualizations and dashboards
  - Great for drilling down from high-level trends of network traffic to specific items of interest

# Kibana: Security & ICS/IoT Security Overview



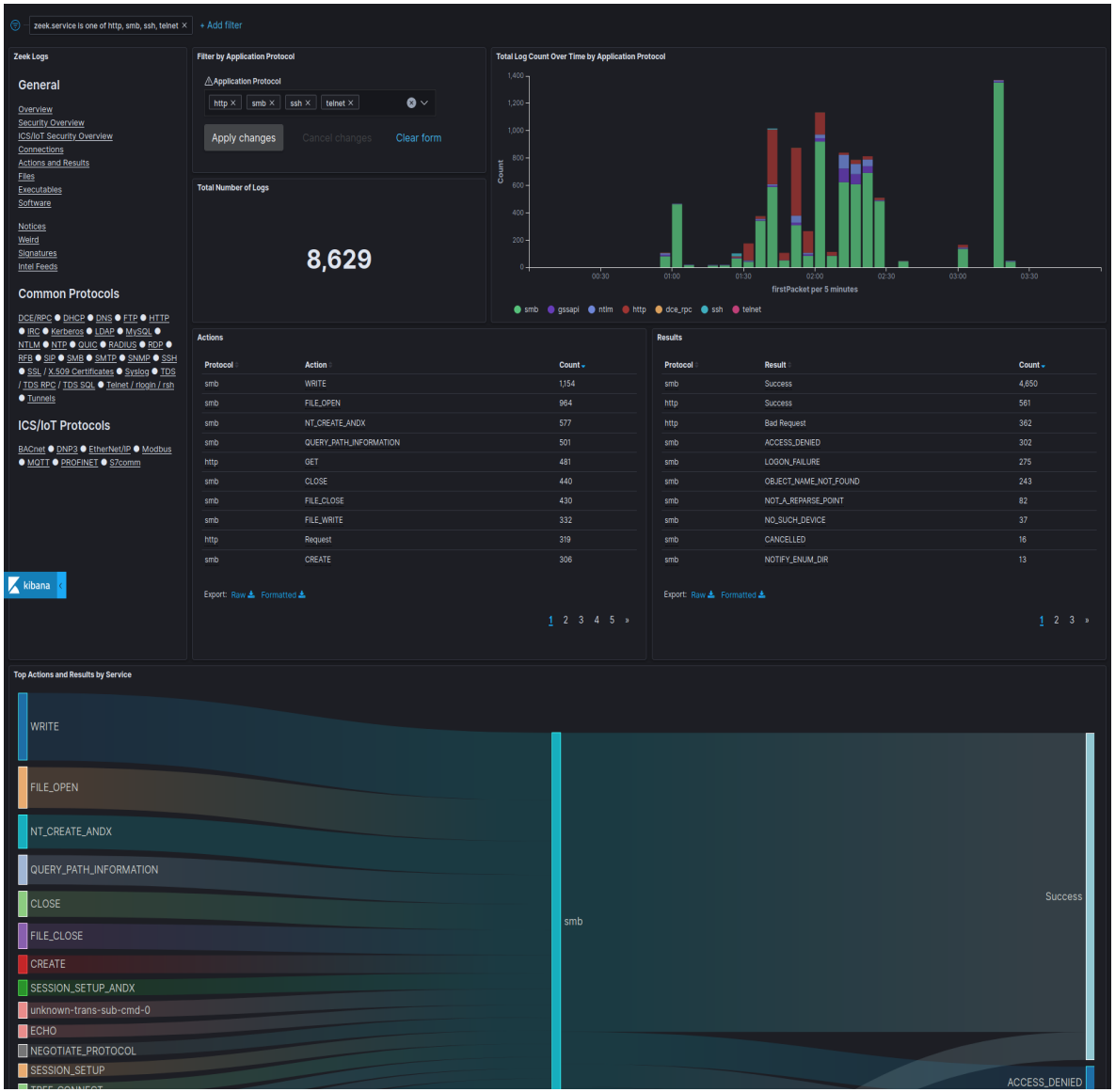



# Kibana: Signatures




- Reports hits from file scanning engines (e.g., ClamAV) against files carved from network traffic

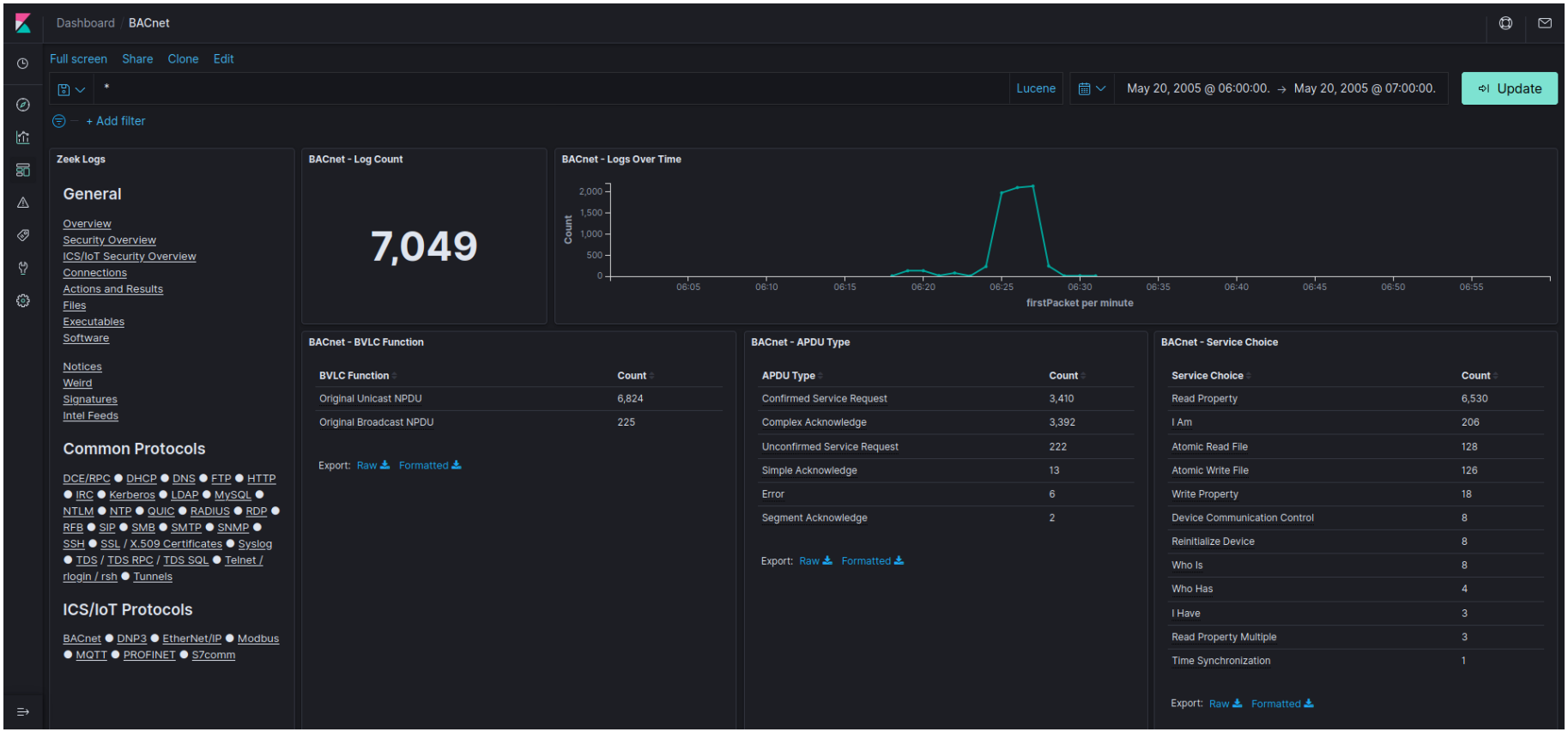
# Kibana: Actions and Results




- Malcolm normalizes “action” (e.g., write, read, create file, logon, logoff, etc.) and “result” (e.g., success, failure, access denied, not found) across protocols

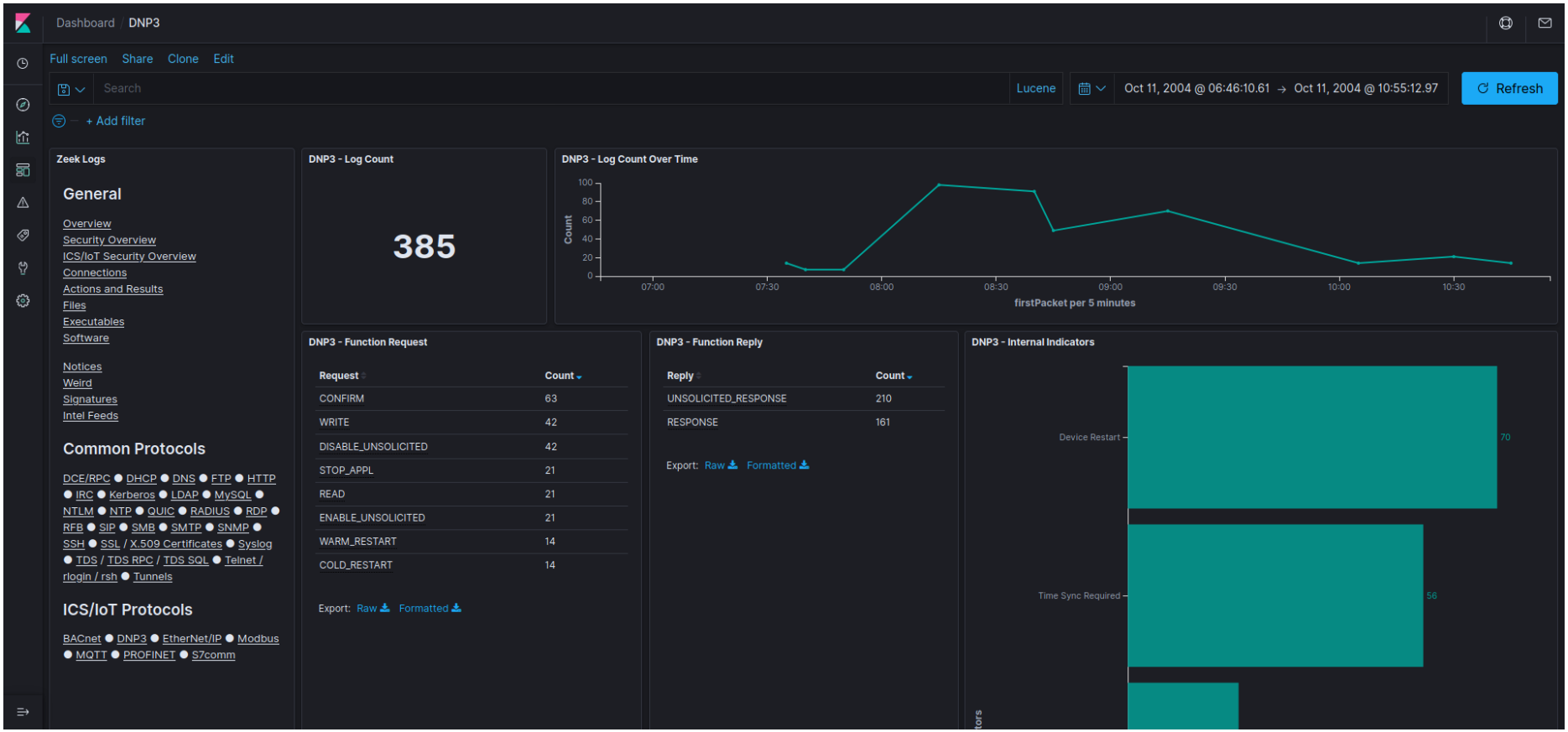
# Kibana: ICS protocol dashboards

## BACnet



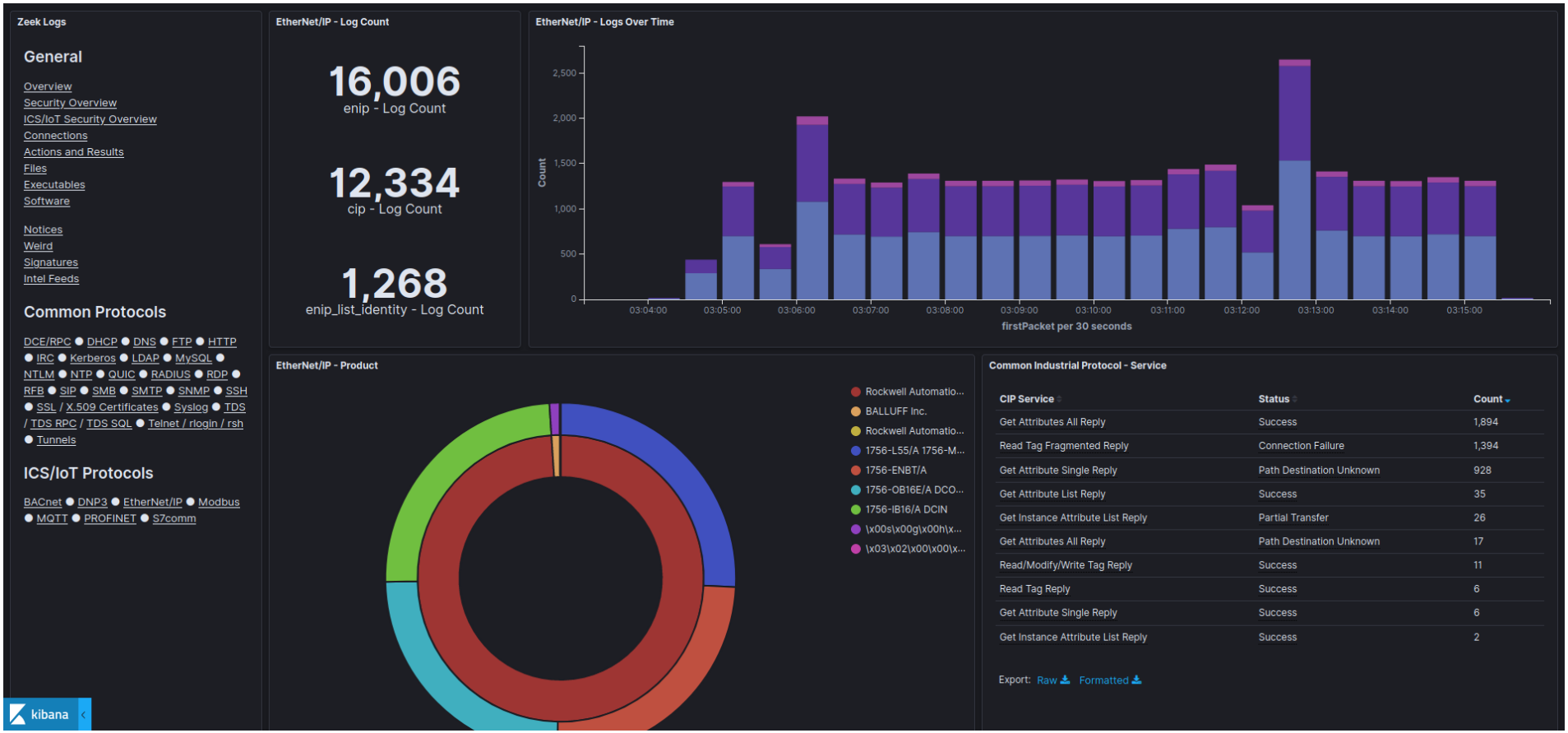
# Kibana: ICS protocol dashboards

## DNP3

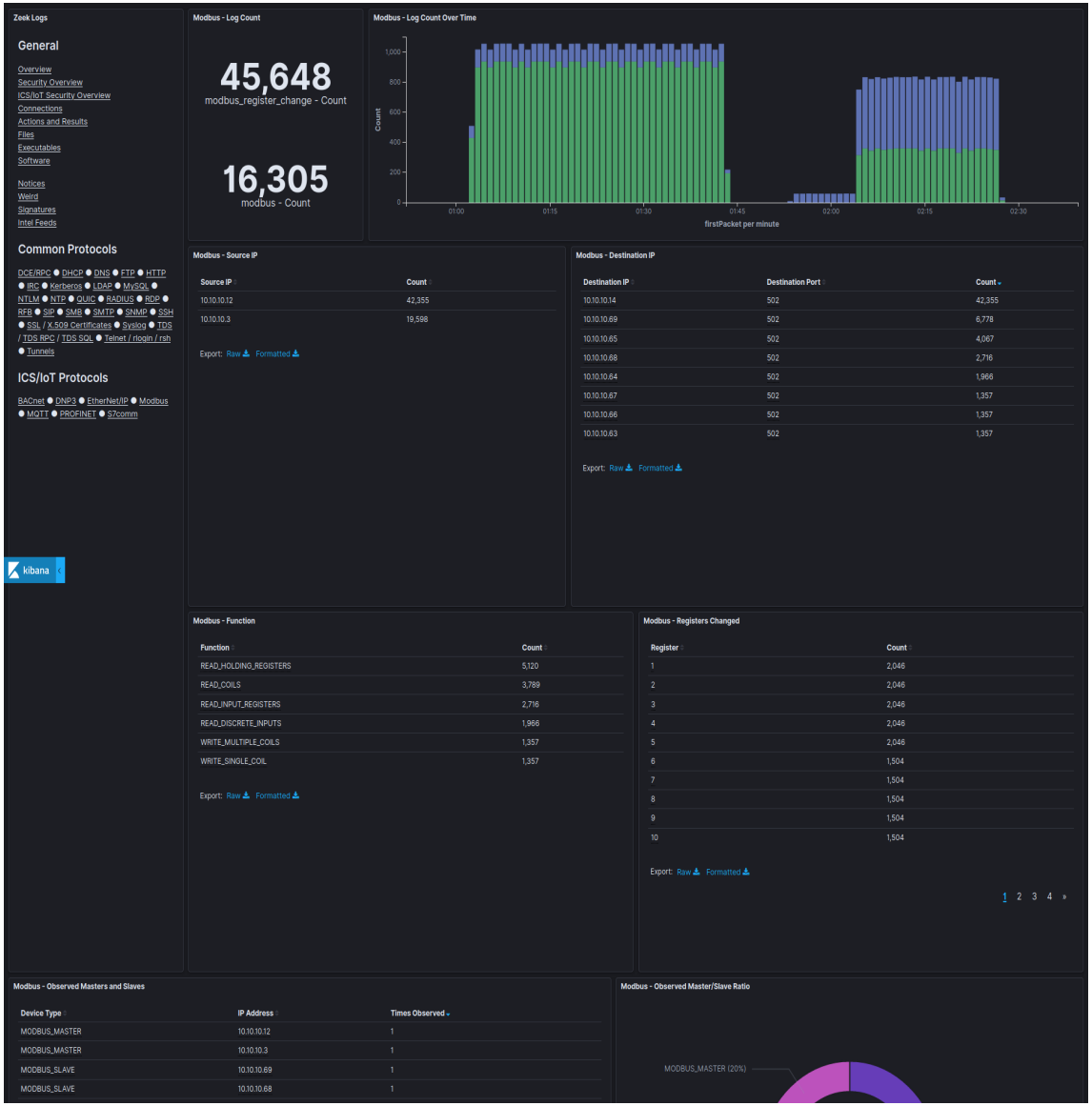


# Kibana: ICS protocol dashboards

## EtherNet/IP

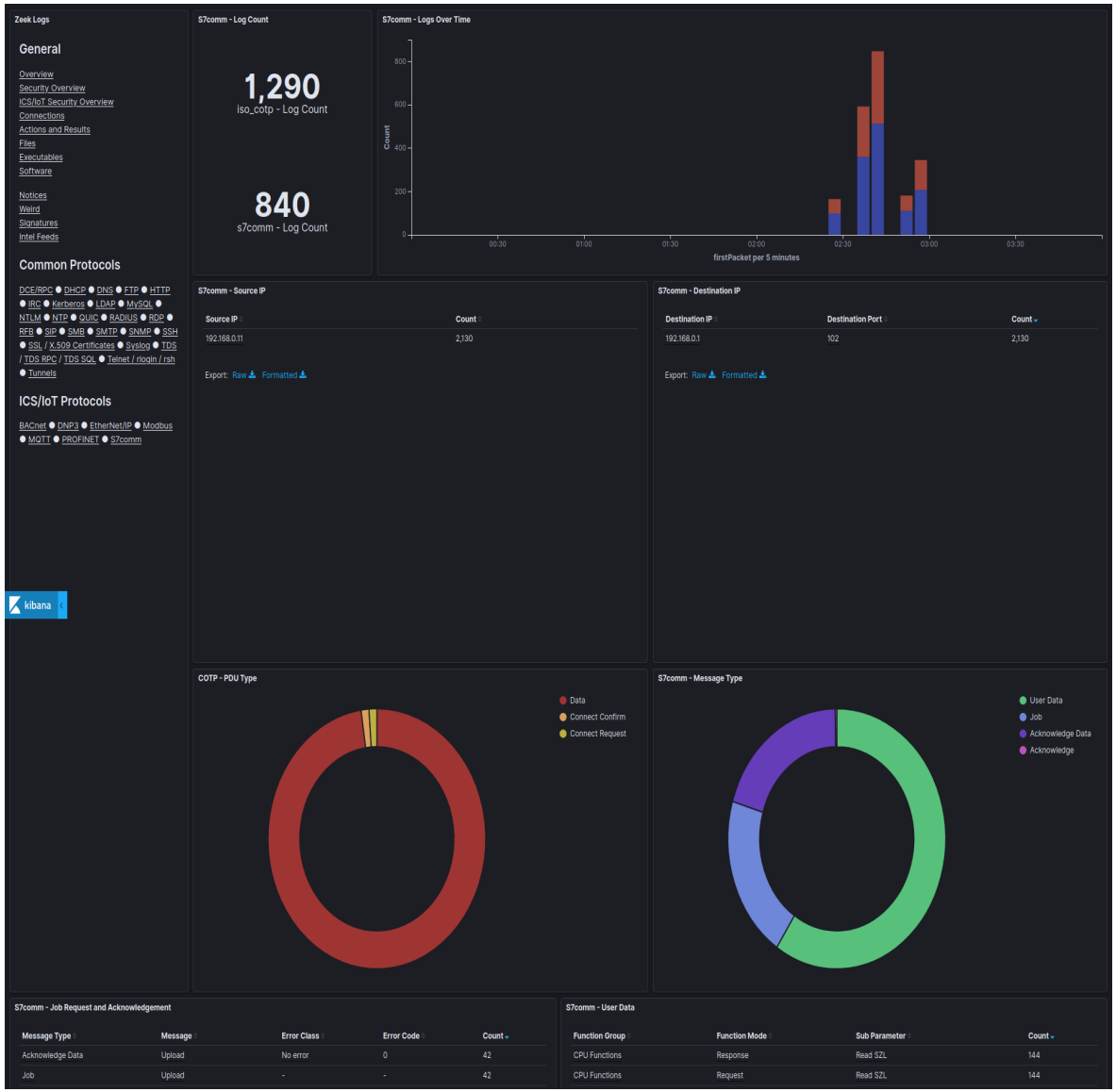


# Kibana: ICS protocol dashboards




Modbus

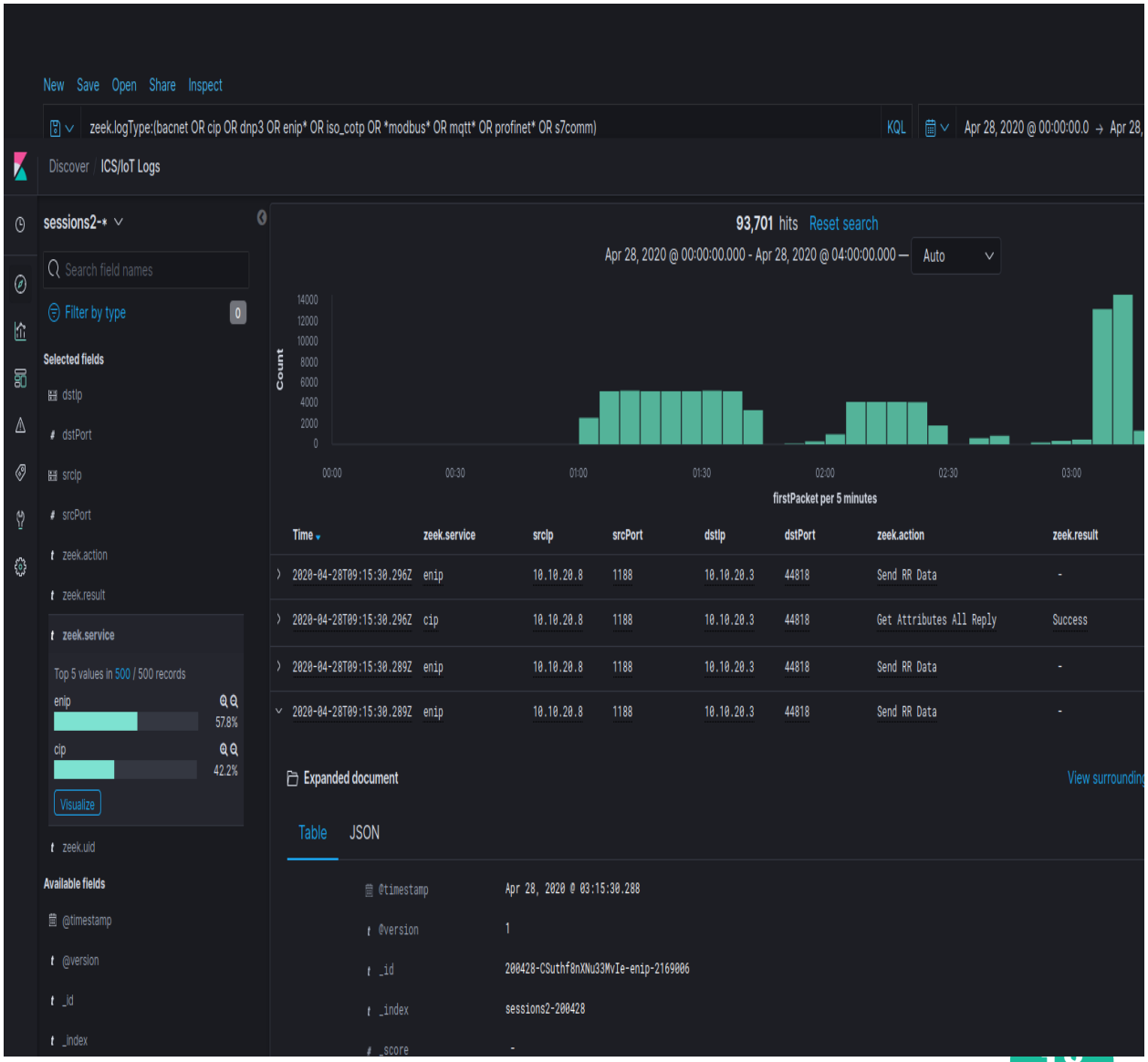
# Kibana: ICS protocol dashboards




S7comm

# Kibana

- **Discover**
  - Field-level details of logs matching filter criteria
  - Create and view saved searches
  - View other events just before and after an event

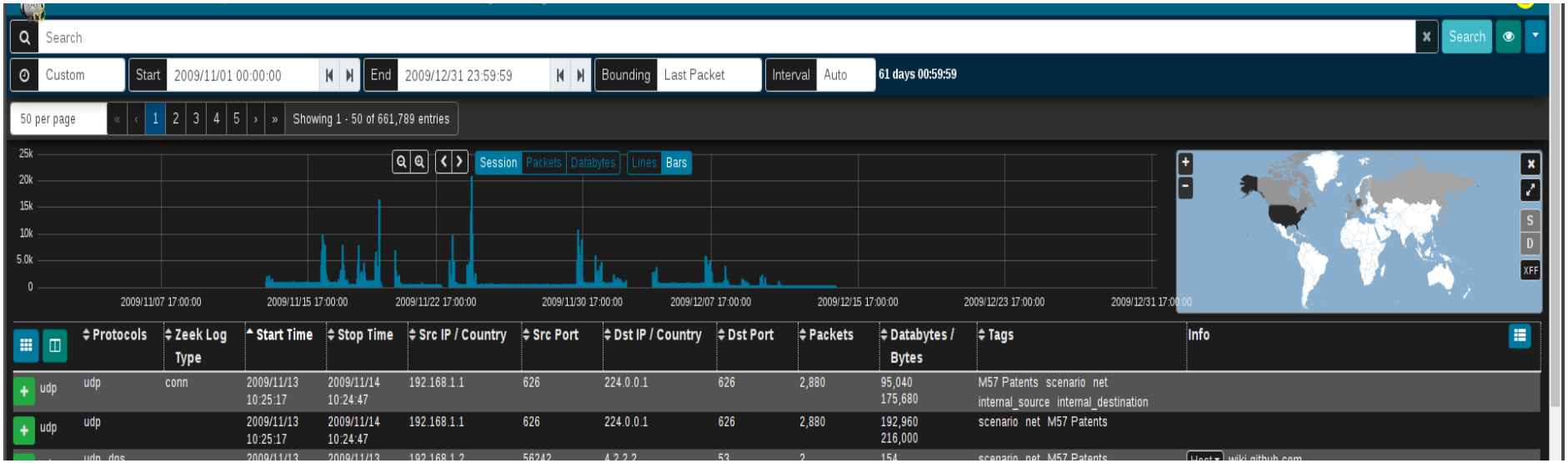


# Moloch

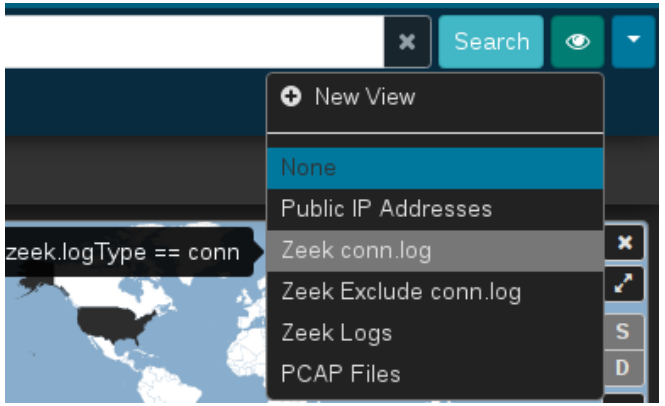

- https://localhost/
- Front end for **both** enriched Zeek logs **and** Moloch sessions
- Filter by Zeek logs or Moloch sessions; or, view both data sources together
- “Wireshark at scale”: full PCAP availability for viewing packet payloads, exporting filtered and joined PCAP sessions and running deep-packet searches

# Moloch: Sessions

- Field-level details of sessions/logs matching filters

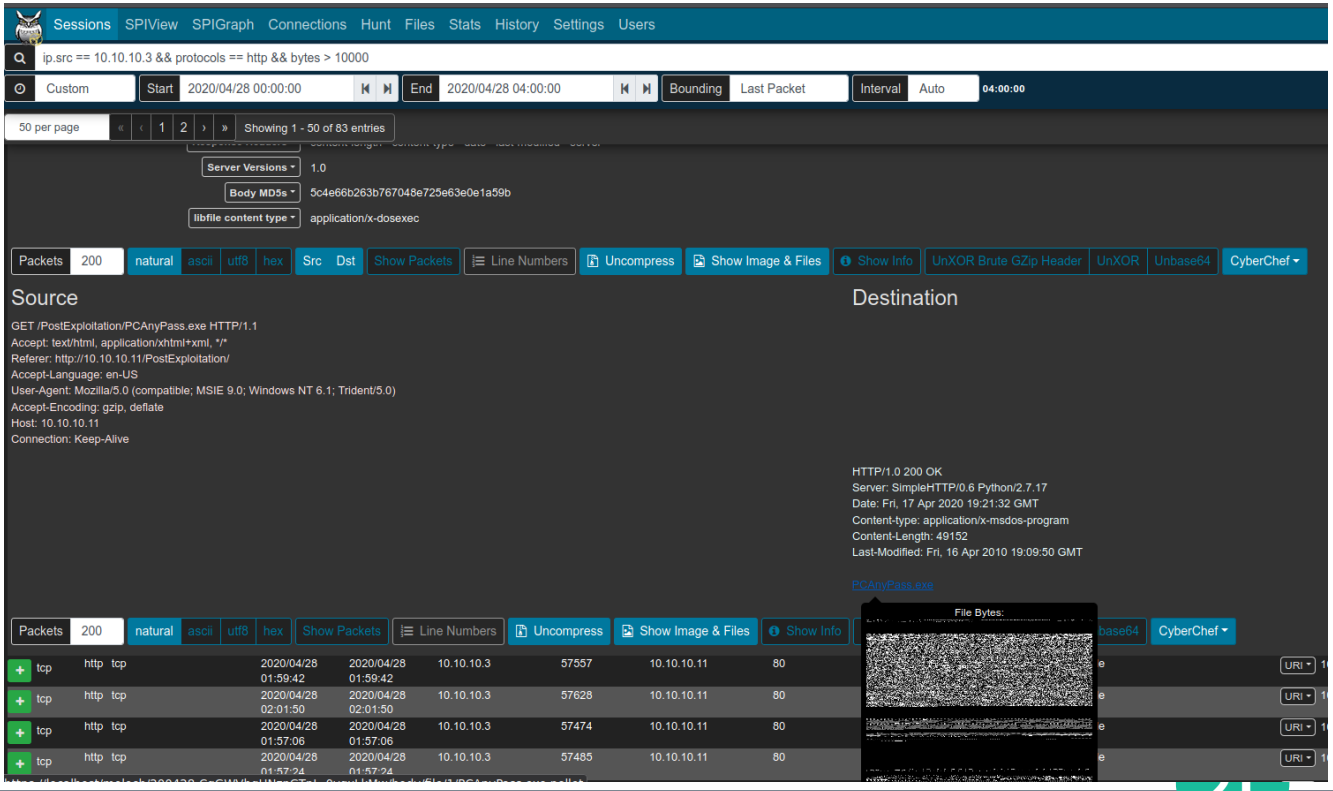



- Views: overlay saved filters on current search
  - E.g., show Moloch sessions vs. Zeek logs



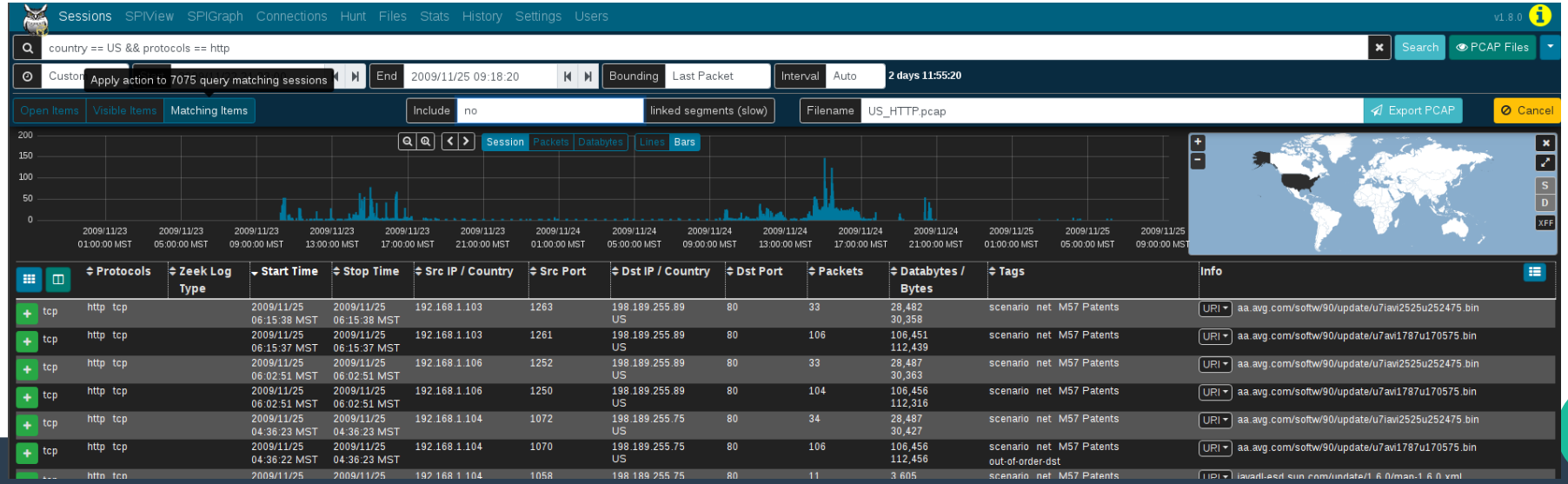
# Moloch: Packet payloads

- Displayed for Moloch sessions with full PCAP
- File carving on the fly
- Download session PCAP
- Examine payload with CyberChef



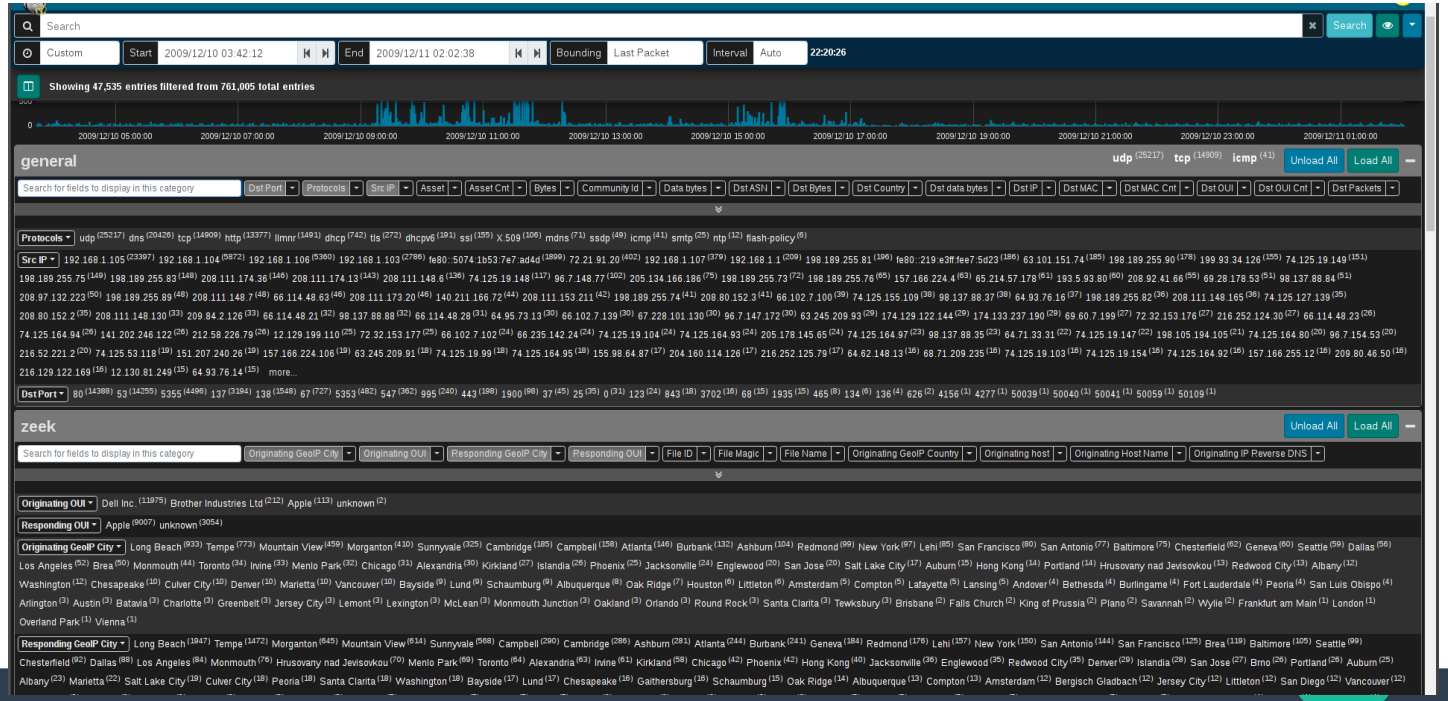
# Moloch: PCAP Export

- Creates a new PCAP file from filtered sessions
- Include open, visible or all matching sessions
- Apply “PCAP Files” view to sessions first
- Narrow as much as possible prior to exporting (huge PCAP files are a pain)



# Moloch: SPIView

- Explore “top *n*” and field cardinality for all fields of both Moloch sessions and Zeek logs
- Apply filters or pivot to Sessions or SPIGraph view for field values of interest
- Limit search to ≤ 1 week before using as it runs many queries

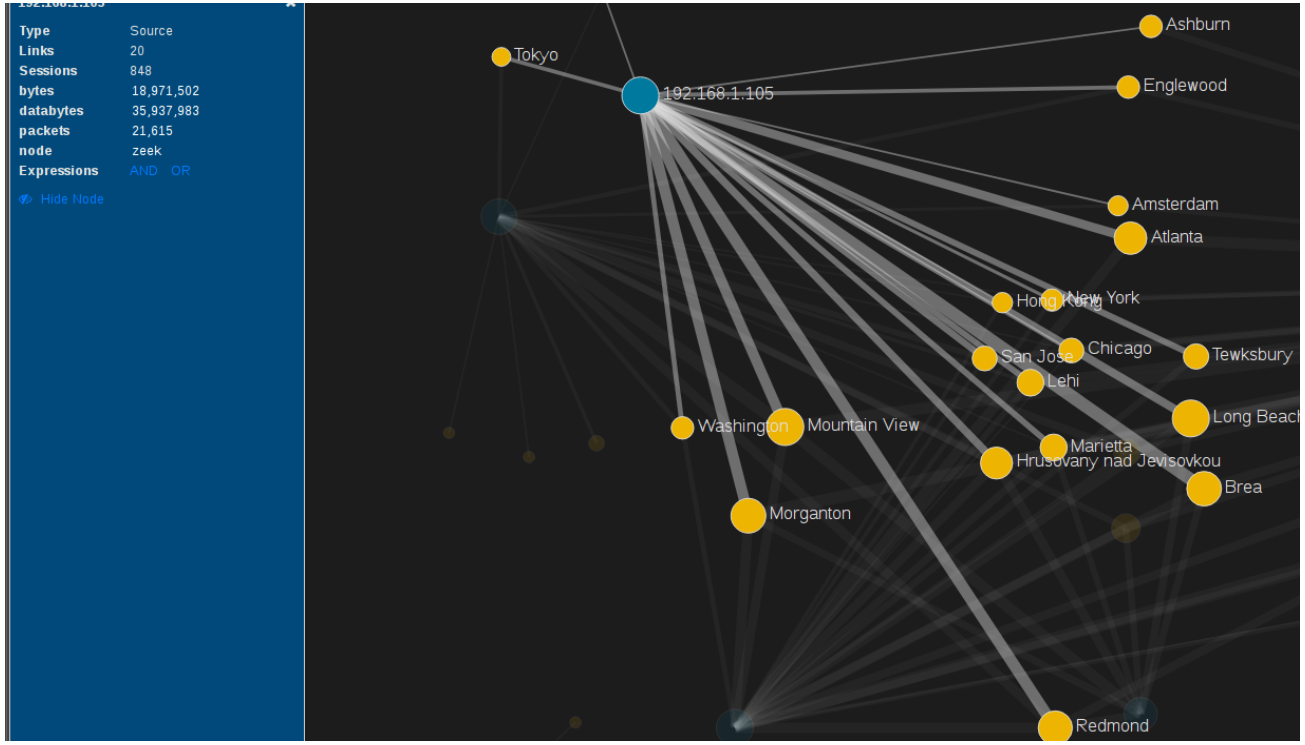


# Moloch: SPIGraph

- View “top *n*” field values chronologically and geographically
- Identify trends and patterns in network traffic

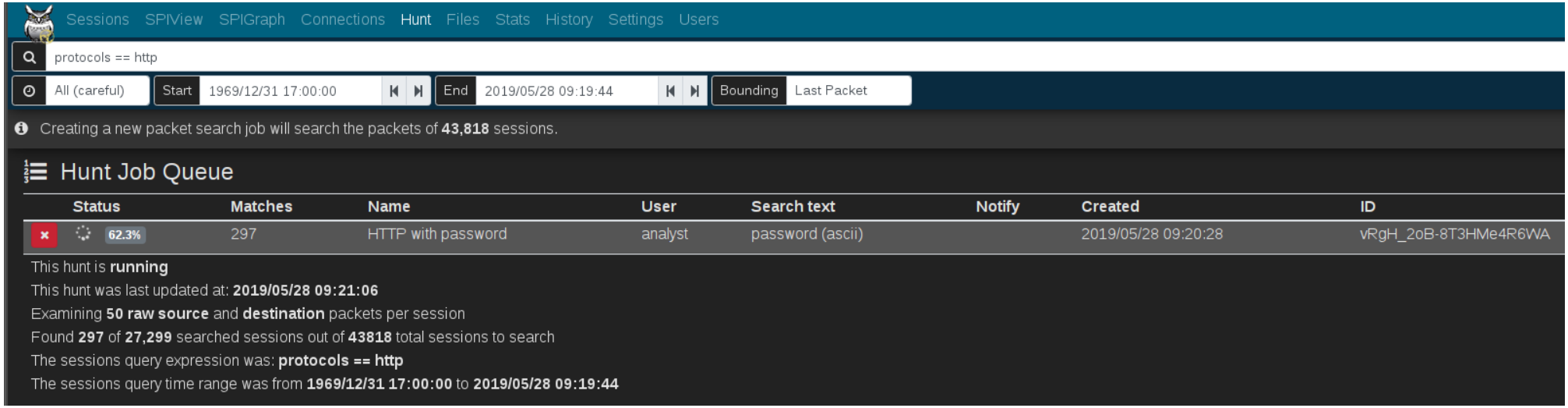
# Moloch: Connections

- Visualize logical relationship between hosts
- Use any combination of fields for source and dest. nodes
- Compare current vs. baseline traffic



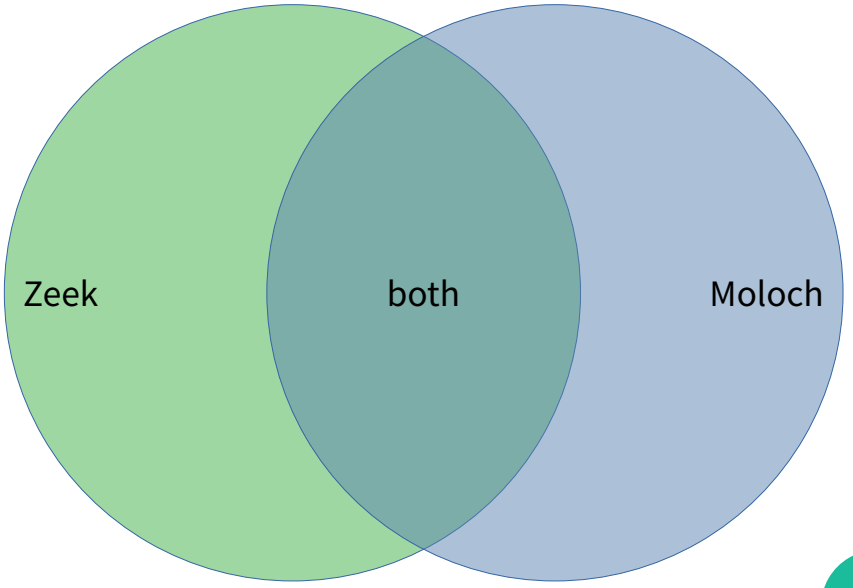
# Moloch: Hunt

- Deep-packet search (“PCAP grep”) of session payloads
- Search for ASCII, hex codes or regular expression matches
- Apply “PCAP Files” view to sessions first



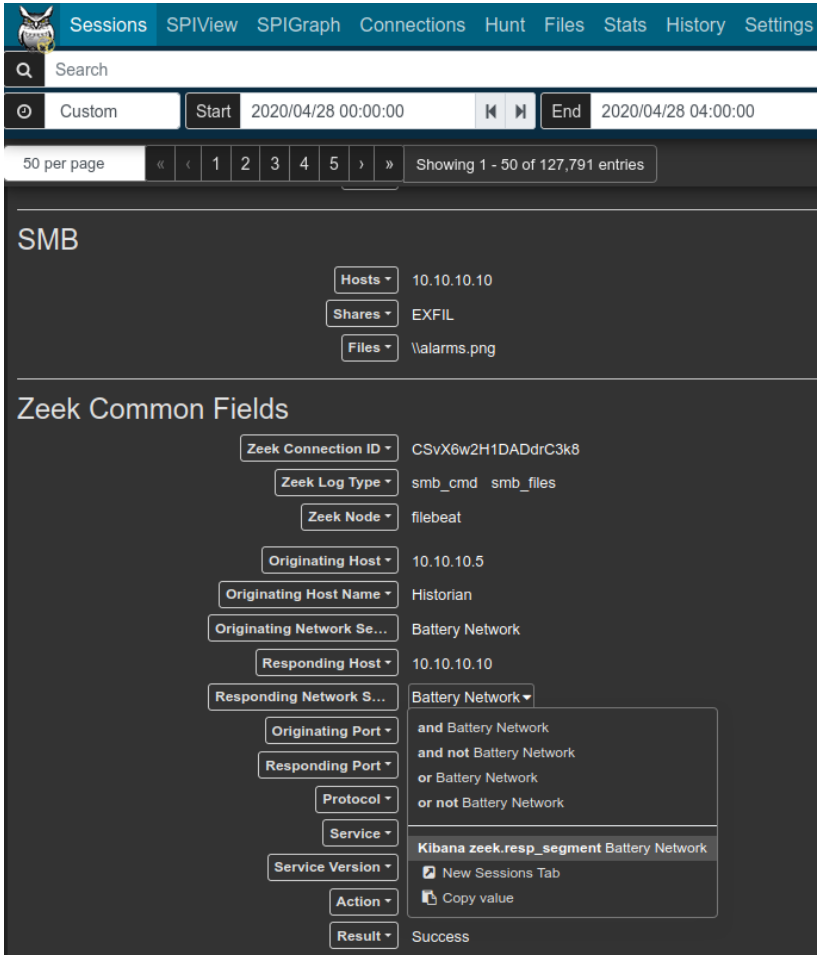
# Using Moloch and Kibana together in Malcolm

- Search syntax is different between Moloch and Kibana (and in some cases, so are **field names**): see **Malcolm documentation** for examples and **Moloch help**
- Despite considerable overlap, there are differences in **protocol parser support** between Zeek and Moloch
  - Learning the strengths of of each will help you more effectively find the good stuff

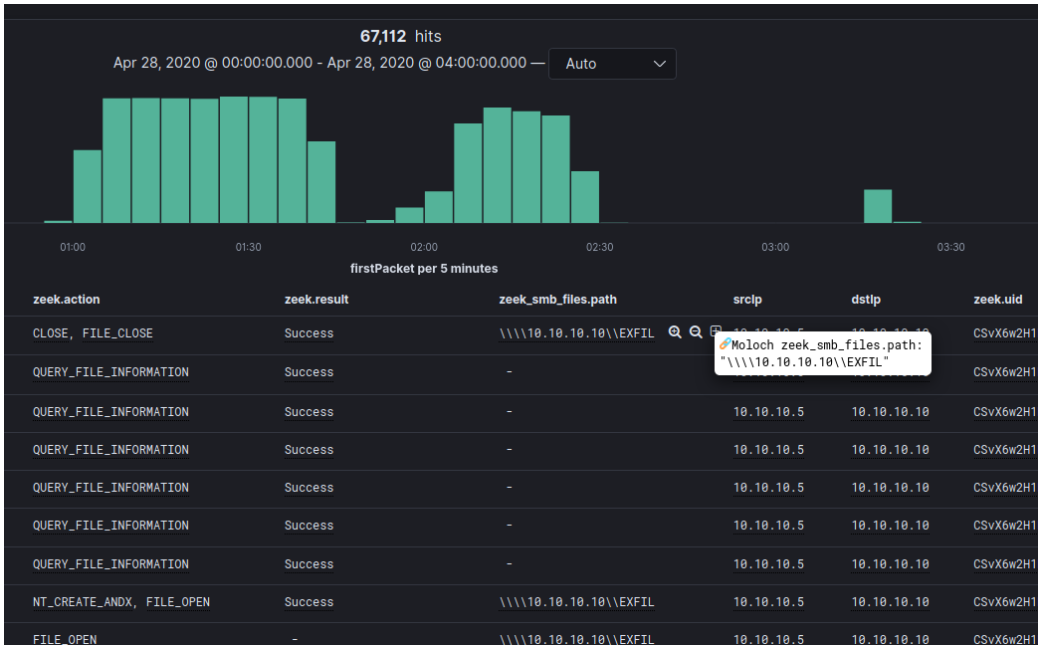


# Pivots: Moloch  Kibana

- Moloch → Kibana



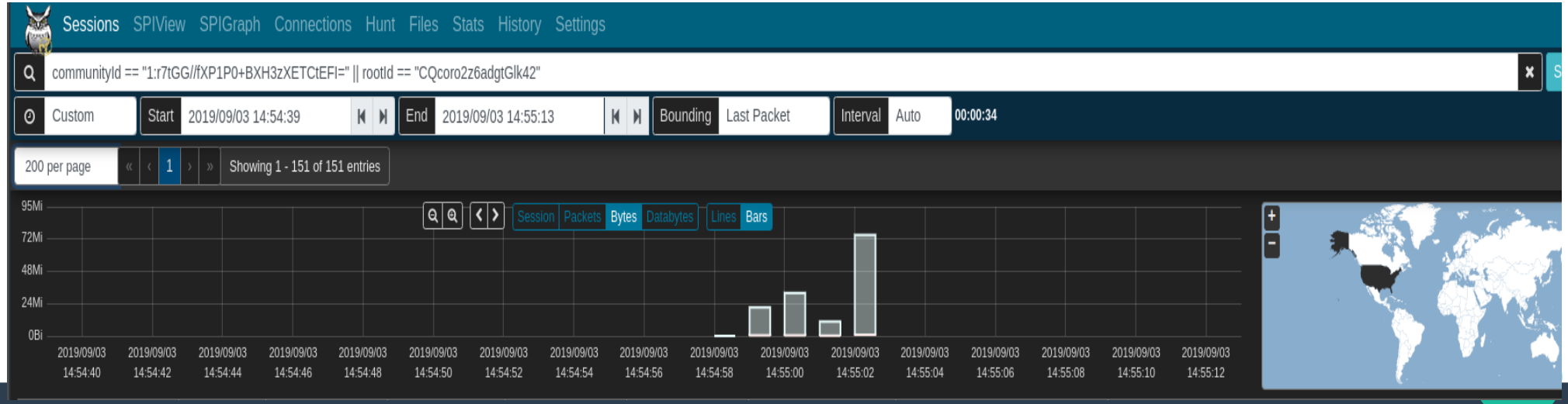

- Kibana → Moloch




Check query search time frame after pivot: sometimes it gets lost in translation.

# Correlating Moloch sessions and Zeek logs

- **Correlate** Zeek logs and Moloch sessions using common fields
- `communityId` fingerprints flows in both and can bridge the two
- `zeek.uid`/`rootId` filters Zeek logs for the same session
- Filter community ID OR'ed with zeek UID to see all Moloch sessions and Zeek logs for the same traffic
  - `communityId == "1:r7tGG//fXP1P0+BXH3zXETCtEFI=" || rootId == "CQcoro2z6adgtGlk42"`



# Search tips

- **Always check your search time frame**
- **“Zoom in” (apply filters) for a particular field value, pivot to another field then “zoom out” (remove filters)**
- **Most UI controls can work with any data field (1000+)**
- **Filter on** `zeek.logType` **(e.g.,** `conn` **to see conn.log)**
- **Filter on protocol or both Moloch and Zeek regardless of data source (e.g.,** `protocol:http` **in Kibana and** `protocols == http` **in Moloch)**
- `tags` **field**
  - Populated for both Moloch sessions and Zeek logs with tags provided on upload and words extracted from filenames
  - `internal_source`, `internal_destination`, `external_source`, `external_destination`, `cross_segment`

**Thanks! Visit Malcolm on GitHub to read the docs, make suggestions, report issues and st★r to show your support!**