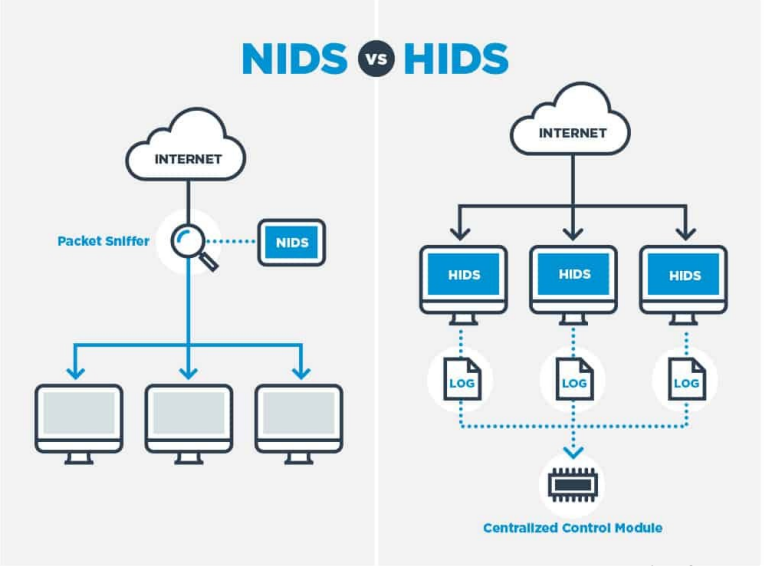
Network Traffic Analysis with Malcolm

Seth Grover, Malcolm developer • Cybersecurity R&D • Idaho National Lab

Intrusion Detection Systems

- HIDS: Host Intrusion Detection Systems
  - Agents run on individual hosts or devices on a network
  - Not what we're talking about today
- NIDS: Network Intrusion Detection Systems
  - Monitor and analyze network traffic for anomalies: suspicious activity, policy violations, etc.
  - Generally passive/out-of-band; otherwise it's an Intrusion Prevention System
  - Detection methods
    - Signature-based detection
    - Statistical anomaly-based detection
    - Stateful protocol analysis detection



comparitech.com

IDS: Types of Attacks

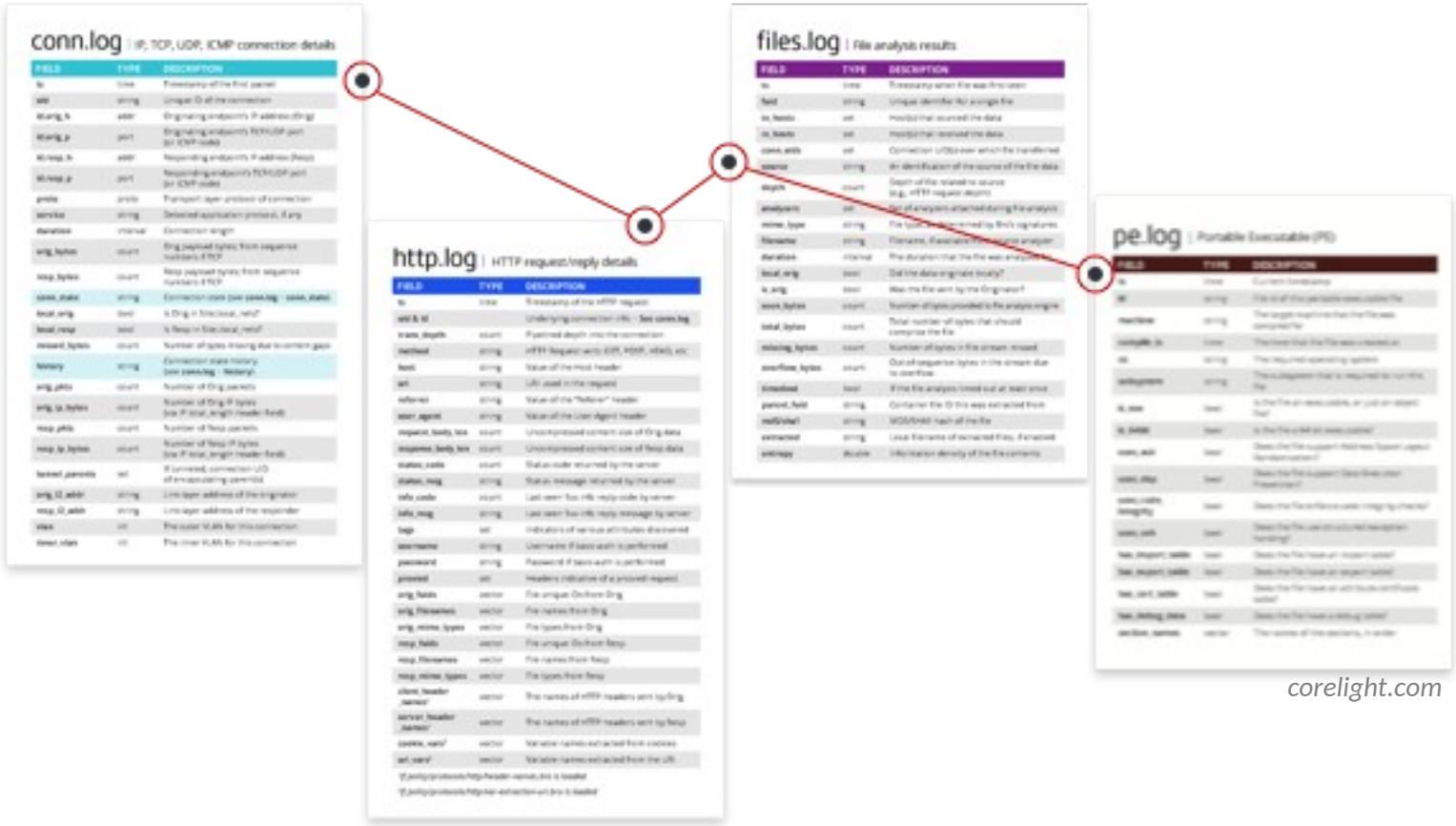
- Scanning Attack
  - Determine network topology
  - IDS highlights connections from one host to many other hosts in the network, or connection attempts to sequential IP addresses and/or ports
- Denial of Service Attack
  - Interrupt service by flooding requests or flaws in protocol implementations
  - IDS identifies large volume of traffic from or to a particular host or invalid connection states (e.g., TCP SYN/ACK with no ACK)
- Penetration Attack
  - Gain access to system resources by exploiting a software or configuration flaw
  - Trickier, but IDS may detect vulnerable software versions or simply alert on unusual operations (e.g., a “write” operation in an already-configured environment with mostly “read” operations)

- Extensible, open-source passive network analysis framework
- More than just an Intrusion Detection System:
  - Packet capture (like TCPDUMP)
  - Traffic inspection (like Wireshark)
  - Intrusion detection (like SNORT)
  - Log recording (like NetFlow and syslog)
  - Scripting framework (like python)

Strengths	Weaknesses
• Analyzes both link-layer and application-layer behavior • Content extraction • Behavioral analysis • Session correlation • Can add support for uncommon protocols through scripts/plugins	• Session metadata only (not full payload) • Setup and configuration can be complicated • Produces flat textual log files which can be unwieldy for in-depth analysis

Zeek Log Files

- Network Protocols
- Files
- Detection
- Network Observations



corelight.com

Network Protocols

- `conn` – Network session tracking
  - Identified by session 4-tuple (originating IP:port, responding IP:port)
  - One session (line in a log file) for every IP connection
  - Unique identifier (UID) ties lines from other logs to a session
- `http`, `modbus`, `ftp`, `dns`, etc.
  - Protocol-specific log files created as traffic is seen
  - Contain application-layer metadata about network activities

Files

- `files` – File analysis results
  - Each transferred file identified with FUID
  - Associated with connection UID(s) over which file was transferred
  - File name, mime type, file size, etc. provided when available
- `pe` – Analysis of Portable Executable (PE) files
  - Target platform, architecture, OS, etc. for executables transferred across the network
- `x509` – Analysis of X.509 public key certificates

Detection

- `notice` – Zeek concept of "alarms," notices draw extra attention to an event
  - `Conn::Content_Gap, DNS::External_Name, FTP::Bruteforcing, Heartbleed::SSL_Heartbeat_Attack, HTTP::SQL_Injection_Attacker, Scan::Address_Scan, Scan::Port_Scan, Software::Vulnerable_Version, SSH::Password_Guessing, SSL::Certificate_Expired, Weird::Activity, …`
  - https://docs.zeek.org/en/stable/zeek-noticeindex.html

Detection (cont.)

- `weird` – Unexpected network-level activity
  - > 150 weirdness indicators across many protocols
  - https://docs.zeek.org/en/stable/scripts/base/frameworks/notice/weird.zeek.html#id1
- `signatures` – Signature matches, including hits from enabled carved file scanners like ClamAV, YARA and capa

Network Observations

- Periodic dump of entities seen over the last day
  - `known_certs` – SSL certificates
  - `known_devices` – MAC addresses
  - `known_hosts` – Hosts with TCP handshakes
  - `known_modbus` – Modbus masters and slaves
  - `known_services` – Services (TCP “servers”)
  - `software` – Software being used on the network (e.g., Apache, OpenSSH, etc.)
    - Could be used for identifying vulnerable versions of software or firmware

Arkime

Strengths	Weaknesses
• Large scale index packet capture and search tool • Packet analysis engine with support for many common IT protocols • Web interface for browsing, searching, analysis and PCAP carving for exporting • PCAP payloads (not just session header/metadata) are viewable and searchable	• No OT protocol support • Adding new protocol parsers requires C programming

Malcolm

A powerful open-source network traffic analysis tool suite.

https://github.com/idaholab/Malcolm

Streamlined deployment

- Suitable for field use (hunt or incident response) or SOC deployment. Runs in Docker on Linux, macOS and Windows platforms. Provides easy-to-use web-based user interfaces.

Industry-standard tools

- Uses Arkime and Zeek for network traffic capture, Logstash for parsing and enrichment, OpenSearch for indexing and Dashboards and Arkime Viewer for visualization. Also leverages OpenSearch Anomaly Detection, YARA, capa, ClamAV, CyberChef and other proven tools for analysis of traffic and artifacts.

Expanding control systems visibility

- Analyzes more protocols used in operational technology (OT) networks than other open-source or paid solutions. Ongoing development is focused on increasing the quantity and quality of industrial control systems (ICS) traffic.

Dedicated sensor appliance

- Includes Hedgehog Linux, a hardened Linux distribution for capturing network traffic and forwarding its metadata to Malcolm.

Malcolm Components

https://github.com/idaholab/Malcolm/#Components

Capture

- zeek
- Arkime
- netsniff-ng
- TCPDUMP

File Scanning

- yara
- ClamAV
- CAPA
- VIRUSTOTAL

Forwarding & Enrichment

- beats
- logstash

Storage

- OpenSearch

Anomaly Detection

- OpenSearch Anomaly Detection Plugin

Alerting

- OpenSearch Alerting Plugin

Visualization

- OpenSearch Dashboards
- Arkime

Payload Analysis

- CyberChef
- Arkime session PCAP export to WIRESHARK

Framework

- docker
- NGINX

Supported Protocols

https://github.com/idaholab/Malcolm/#Protocols

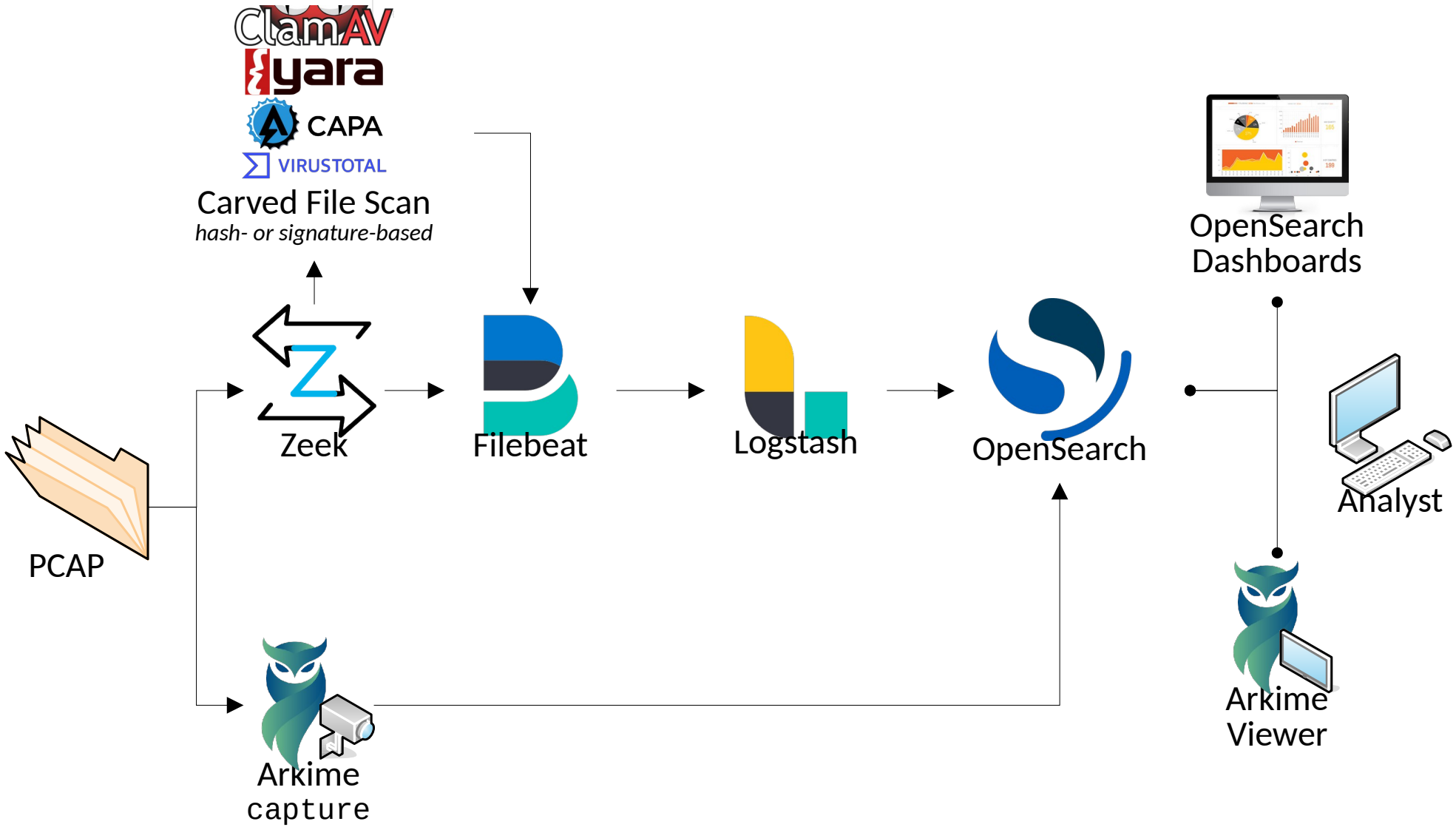
Internet layer
Border Gateway Protocol (BGP)
Building Automation and Control (BACnet)
Bristol Standard Asynchronous Protocol (BSAP)
Distributed Computing Environment / Remote Procedure Calls (DCE/RPC)
Dynamic Host Configuration Protocol (DHCP)
Distributed Network Protocol 3 (DNP3)
Domain Name System (DNS)
EtherCAT
EtherNet/IP / Common Industrial Protocol (CIP)
FTP (File Transfer Protocol)
Google Quick UDP Internet Connections (gQUIC)
Hypertext Transfer Protocol (HTTP)
IPsec
Internet Relay Chat (IRC)
Lightweight Directory Access Protocol (LDAP)
Kerberos
Modbus
MQ Telemetry Transport (MQTT)
MySQL
NT Lan Manager (NTLM)
Network Time Protocol (NTP)
Oracle
Open Shortest Path First (OSPF)
OpenVPN
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)
Secure Shell (SSH)
Secure Sockets Layer (SSL) / Transport Layer Security (TLS)
Session Initiation Protocol (SIP)
Server Message Block (SMB) / Common Internet File System (CIFS)
Simple Mail Transfer Protocol (SMTP)
Simple Network Management Protocol (SNMP)
SOCKS
STUN (Session Traversal Utilities for NAT)
Syslog
Tabular Data Stream (TDS)
Telnet / remote shell (rsh) / remote login (rlogin)
TFTP (Trivial File Transfer Protocol)
WireGuard
various tunnel protocols (e.g., GTP, GRE, Teredo, AYIYA, IP-in-IP, etc.)

* *Industrial control systems protocols indicated with **bold***

Malcolm

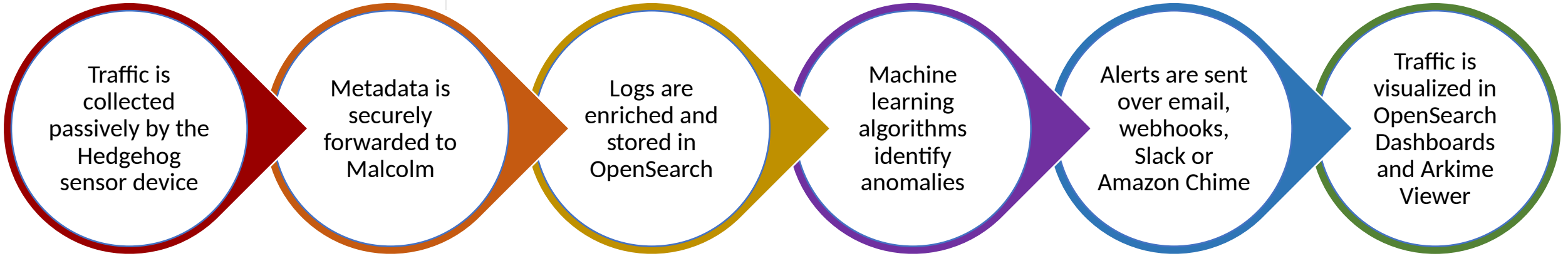
Data Pipeline

https://github.com/idaholab/Malcolm



Malcolm Data Pipeline

https://github.com/idaholab/Malcolm



- Zeek and Arkime Capture generate metadata about network communications
- Full PCAP may be stored locally on the sensor
- Files transfers are detected and the files scanned for threats
- PCAP may also be uploaded to or captured by Malcolm without requiring a dedicated sensor

- All communications between the sensor and aggregator are TLS-encrypted
- Sensor data including resource utilization, syslog, audit logs, temperatures and more may also be forwarded

- Lookups are performed for GeoIP, ASN, MAC-to-vendor, community ID, domain name entropy, etc.
- Network events normalized across protocols and data sources
- Best-guess techniques applied for identifying obscure ICS traffic
- Enriched metadata may be forwarded to higher-tiered Malcolm instance

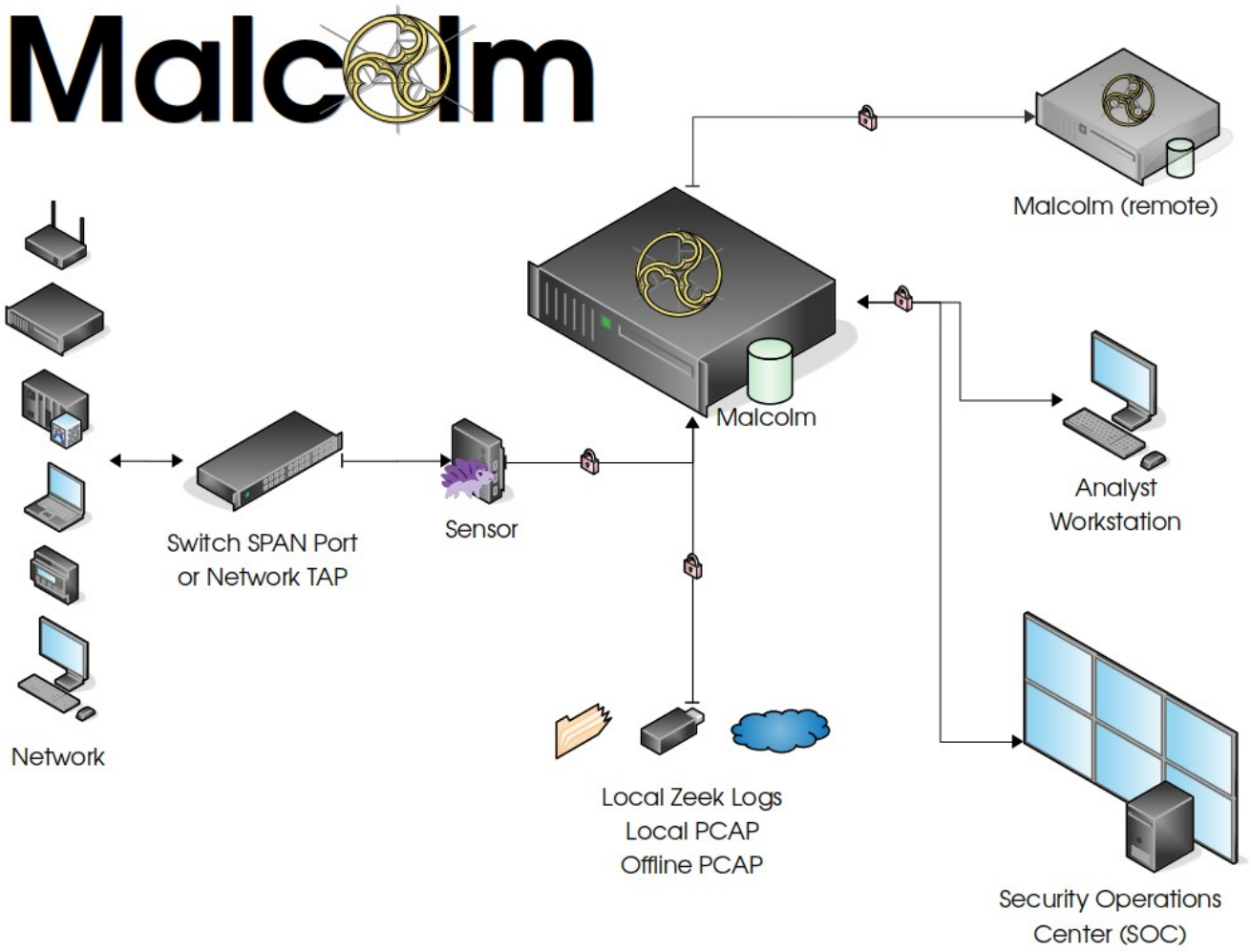
- Default detectors are provided for action and result, flow size and types of transferred files
- Custom detectors may be created for any aspect of any supported protocol

- Alerts may be triggered by exceeded thresholds, anomalies detected, custom queries, etc.

- Dozens of custom dashboards are provided for all supported protocols
- PCAP payloads are retrieved from sensor automatically on demand
- Custom visualizations may be created via drag-and-drop interface
- Malcolm can authenticate users from its own list or via Active Directory / LDAP

Configuring and Running Malcolm

- Runs natively in Docker or in a Virtual Machine
- 16+GB RAM, 4+ cores, “enough” disk for PCAP and logs suggested
- Documentation and source code on GitHub: github.com/idaholab/Malcolm
- Walkthroughs on YouTube: search “Malcolm Network Traffic Analysis”

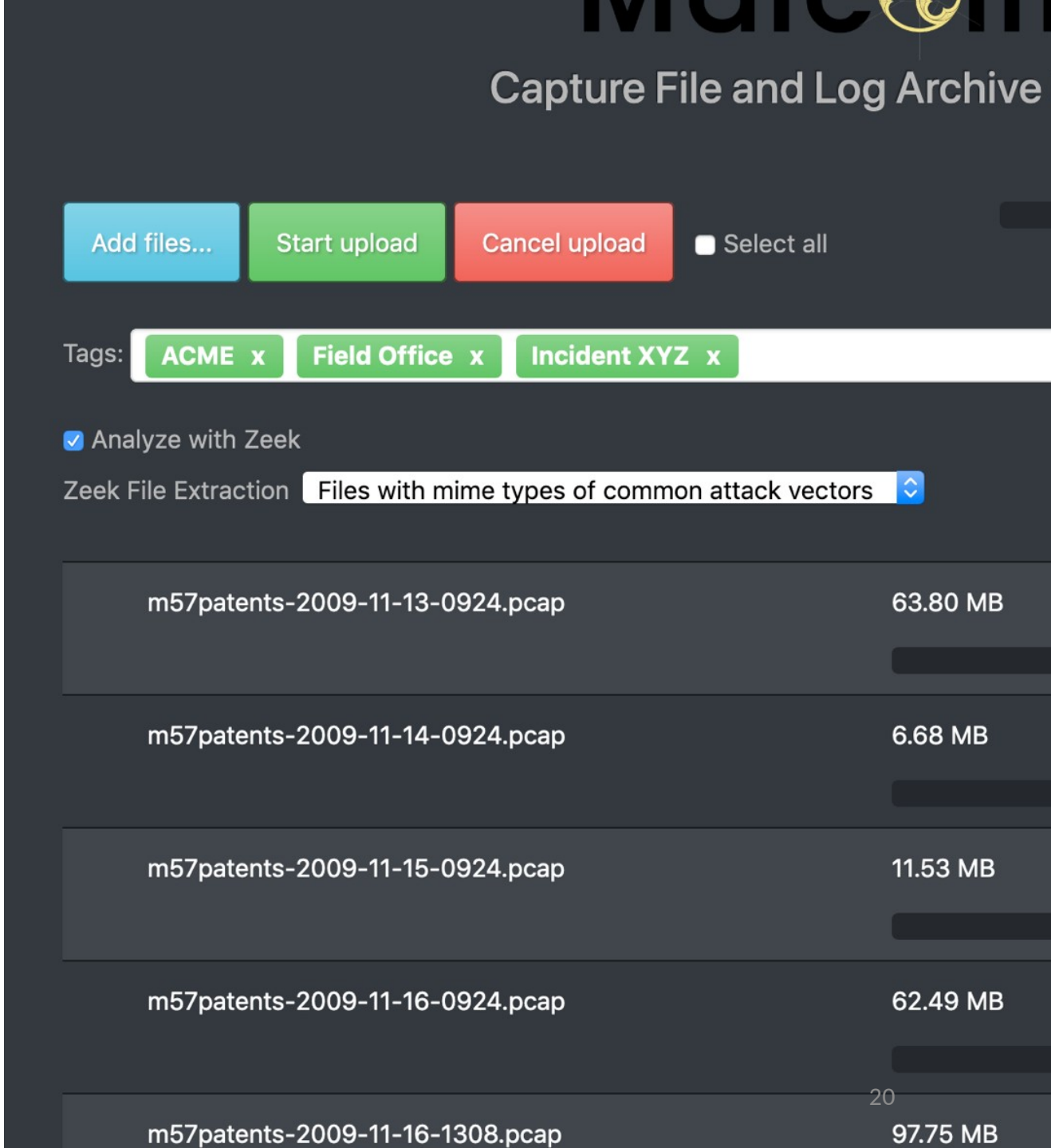


Identifying Network Hosts and Subnets

- Assign custom names to network hosts and subnets prior to PCAP import
- Allows identification of cross-segment traffic and name-based search and filter
- Define in text file(s) or via web interface
- https://localhost/name-map-ui

Importing Traffic Captures for Analysis

- Specify tags for search and filter
- Enable Zeek analysis and file extraction
  - Or configure as global default
- Upload PCAP files or archived Zeek logs
  - pcapng not supported yet
- https://localhost/upload



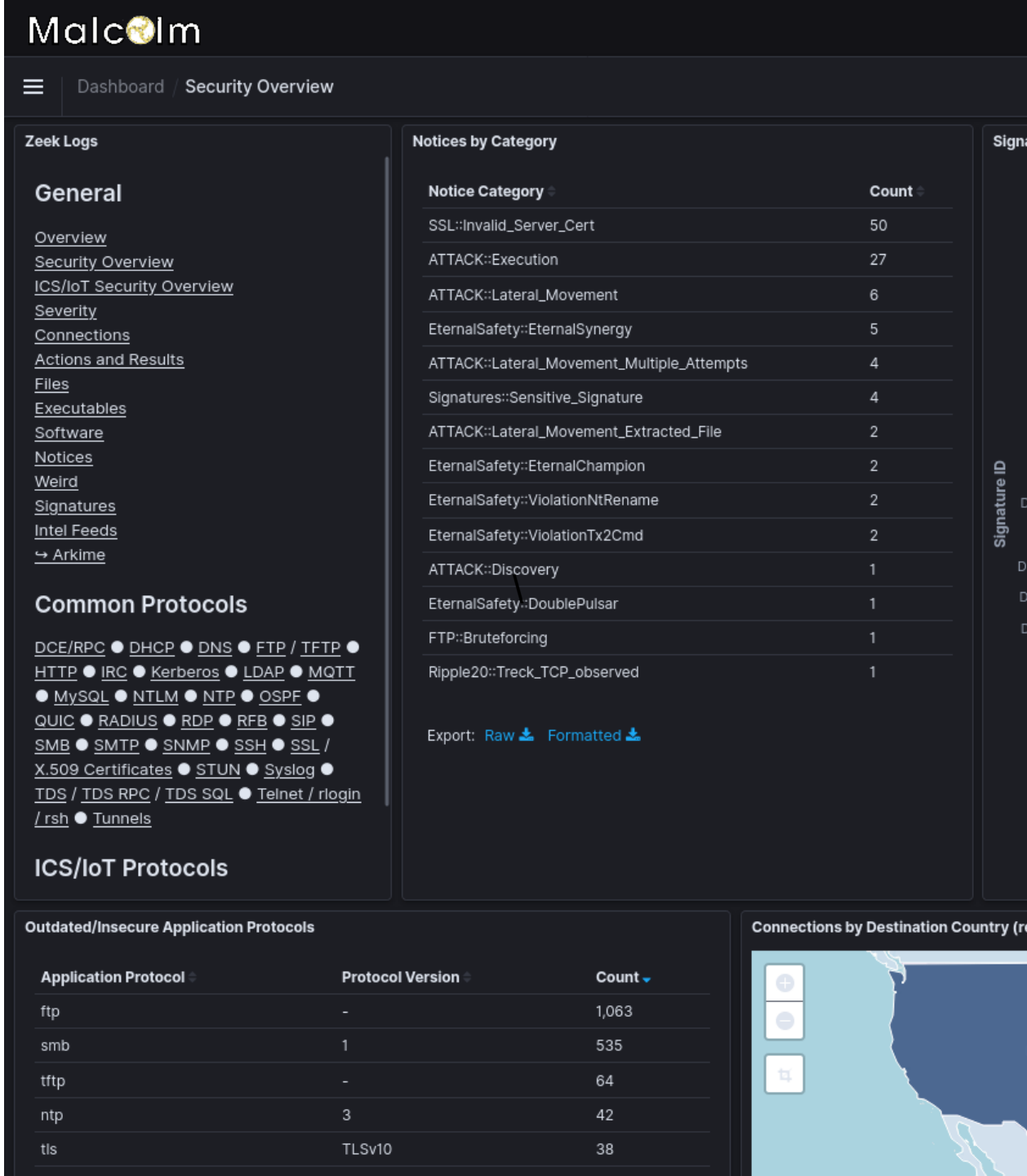
Data Tagging and Enrichment

- 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)

- `tags` field
  - Populated for both Arkime sessions and Zeek logs with tags provided on upload and words extracted from PCAP filenames
  - `internal_source`, `internal_destination`, `external_source`, `external_destination`, `cross_segment`

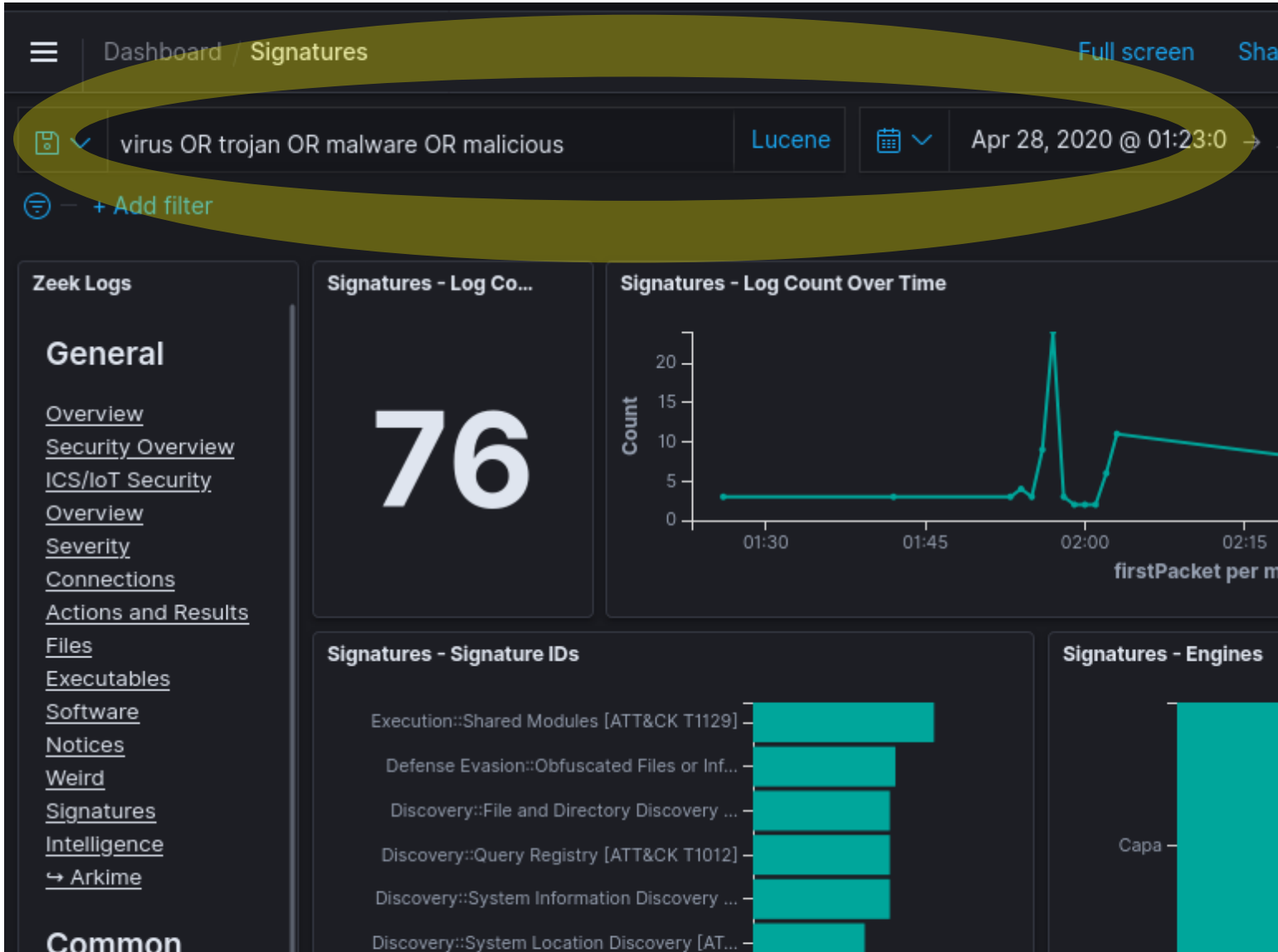
OpenSearch Dashboards

- Front end for Zeek logs
- Prebuilt visualizations for all protocols Malcolm parses
- WYSIWYG editors to create custom visualizations and dashboards
- Drill down from high-level trends to specific items of interest
- https://localhost/dashboards



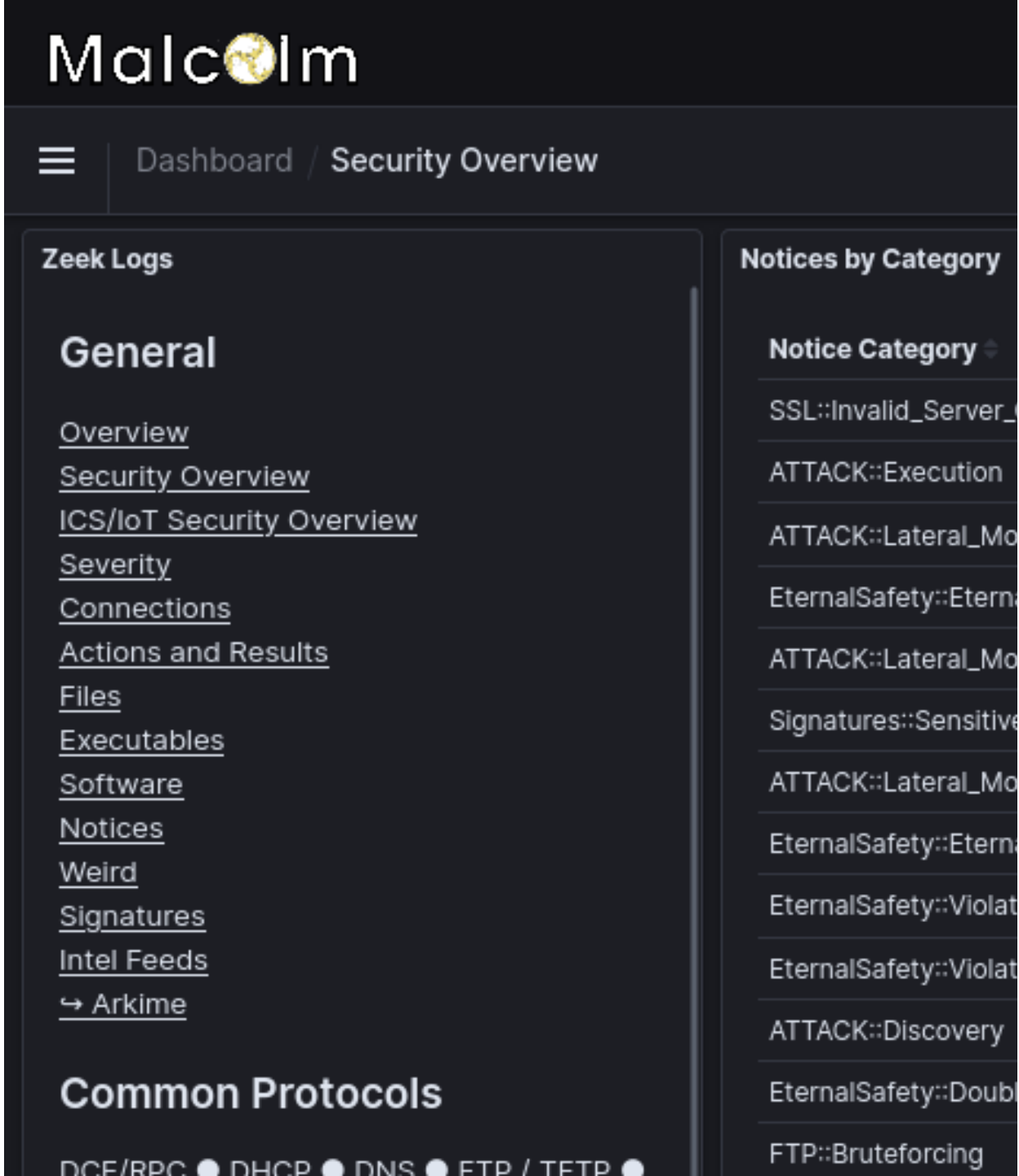
Dashboards Filters and Search

- Time filter: define search time frame
- Query bar: write queries in Lucene syntax or DQL (Dashboards Query Language)
- Filter bar: define filters using a UI
  - Pin filters as you move across dashboards
- Save queries and filters for reuse



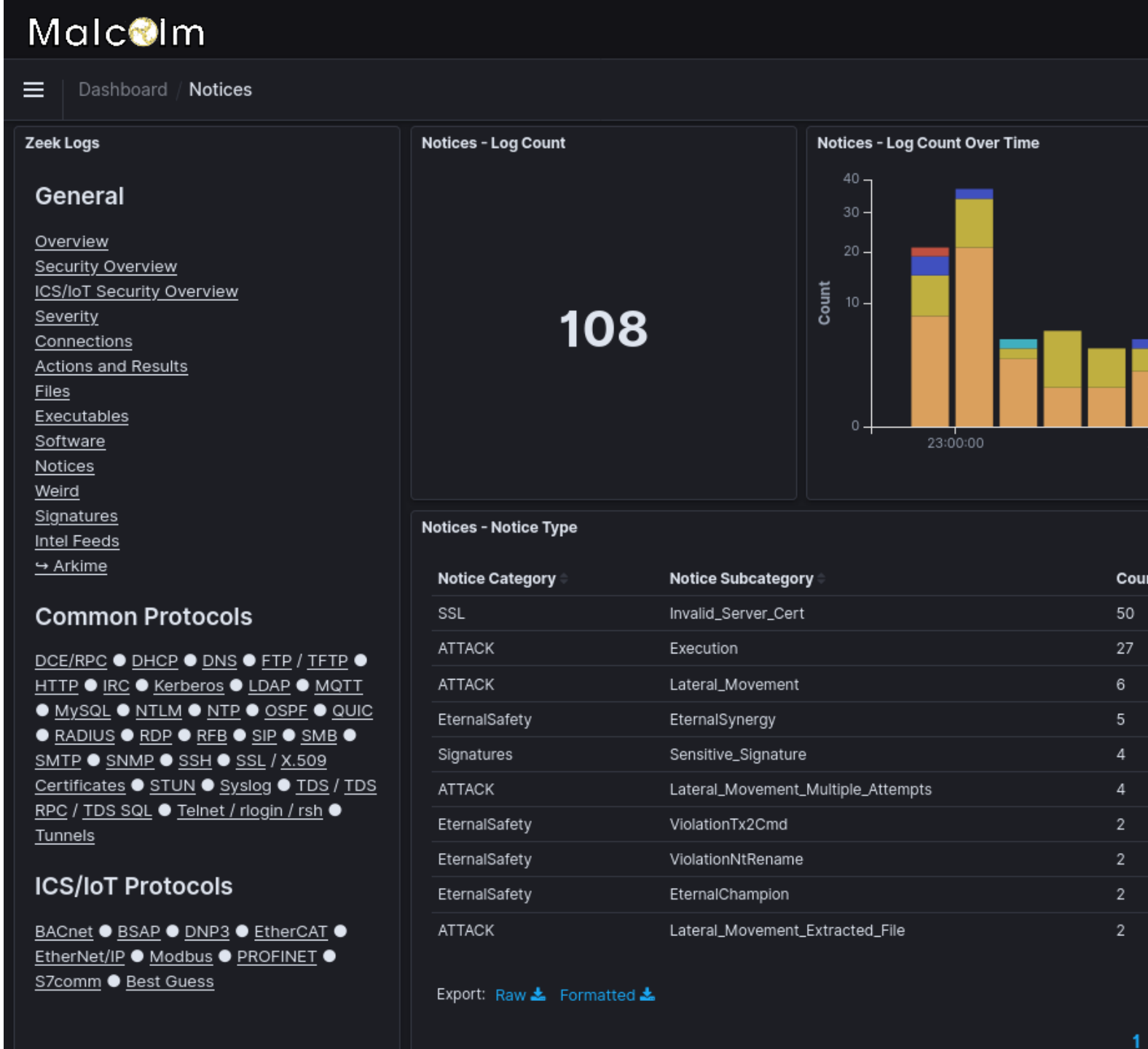
Overview Dashboards

- High-level view of trends, sessions and events
- Populated from logs across all protocols
- Good jumping-off place for investigation

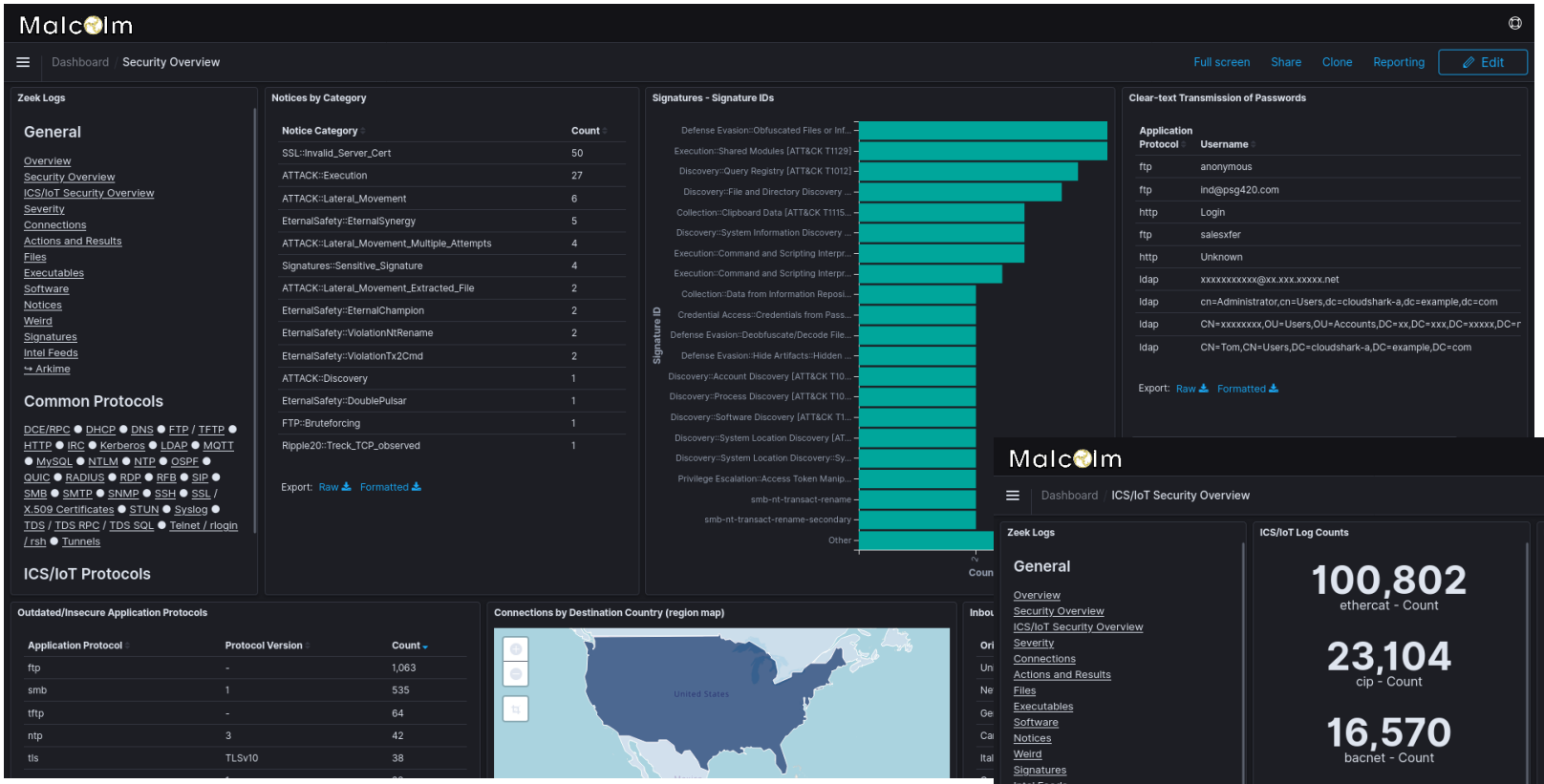


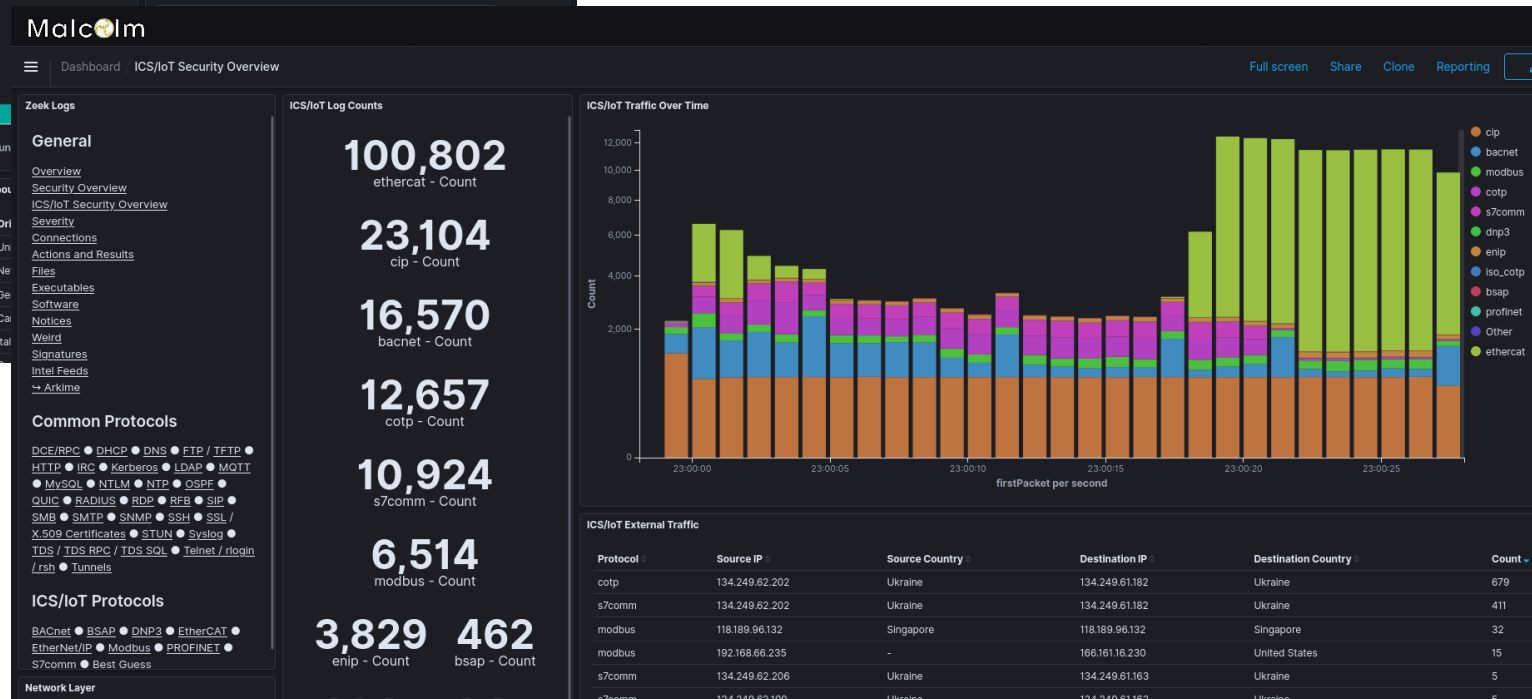
Notices

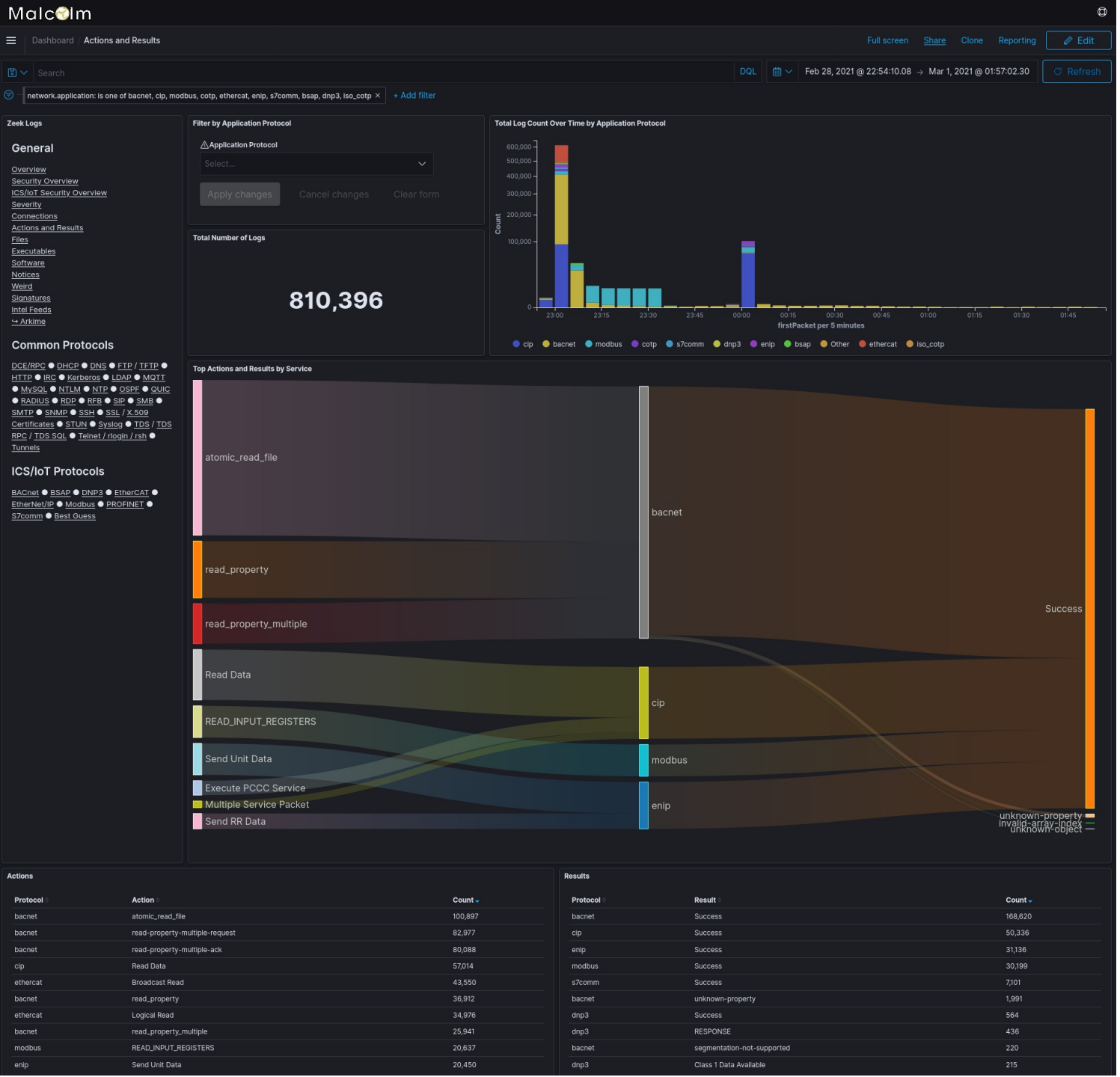
- Zeek notices are things that are odd or potentially bad
- In addition to Zeek's defaults, Malcolm raises notices for recent critical vulnerabilities and attack techniques



Security & ICS/IoT Security Overviews





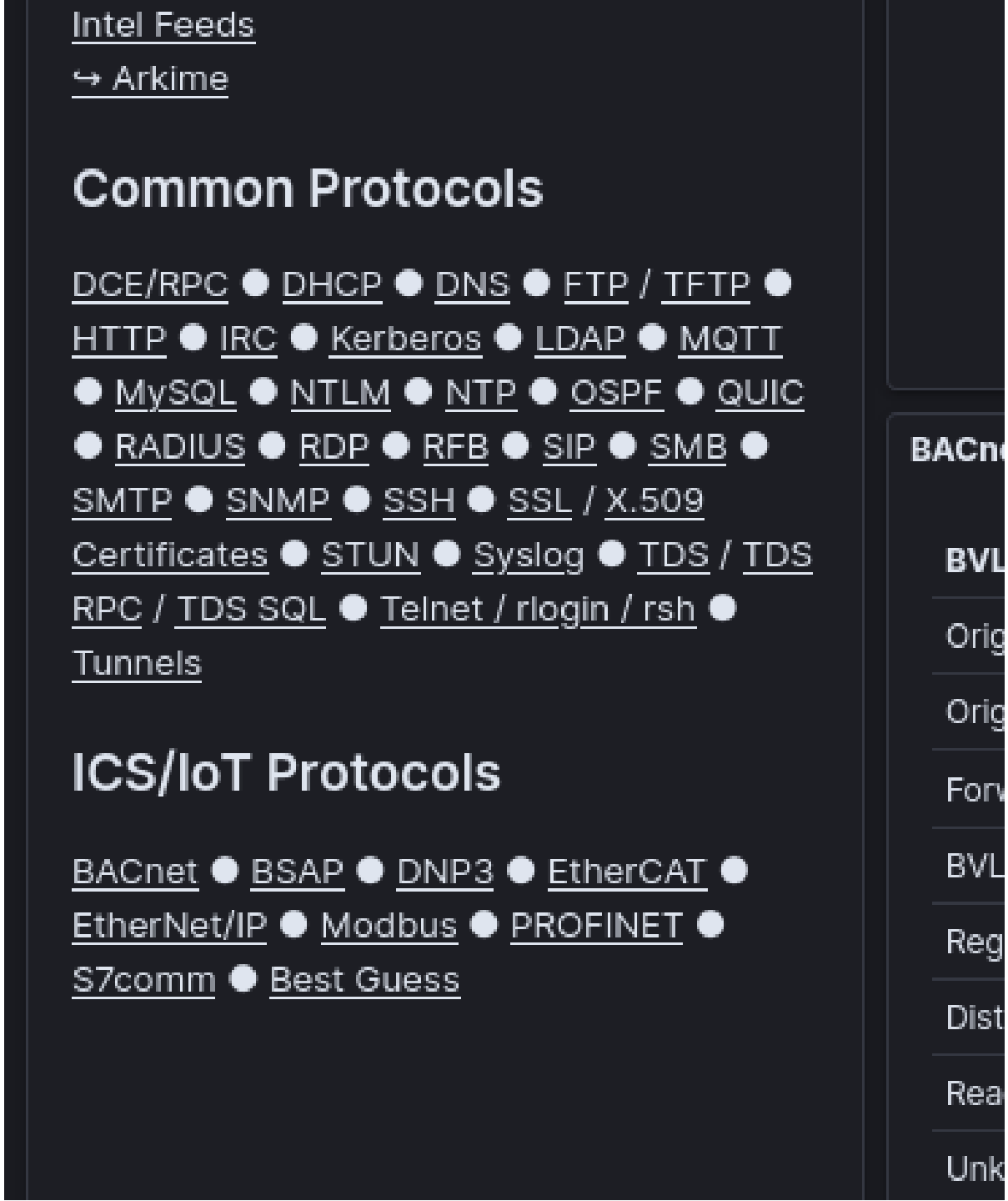


Actions and Results

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

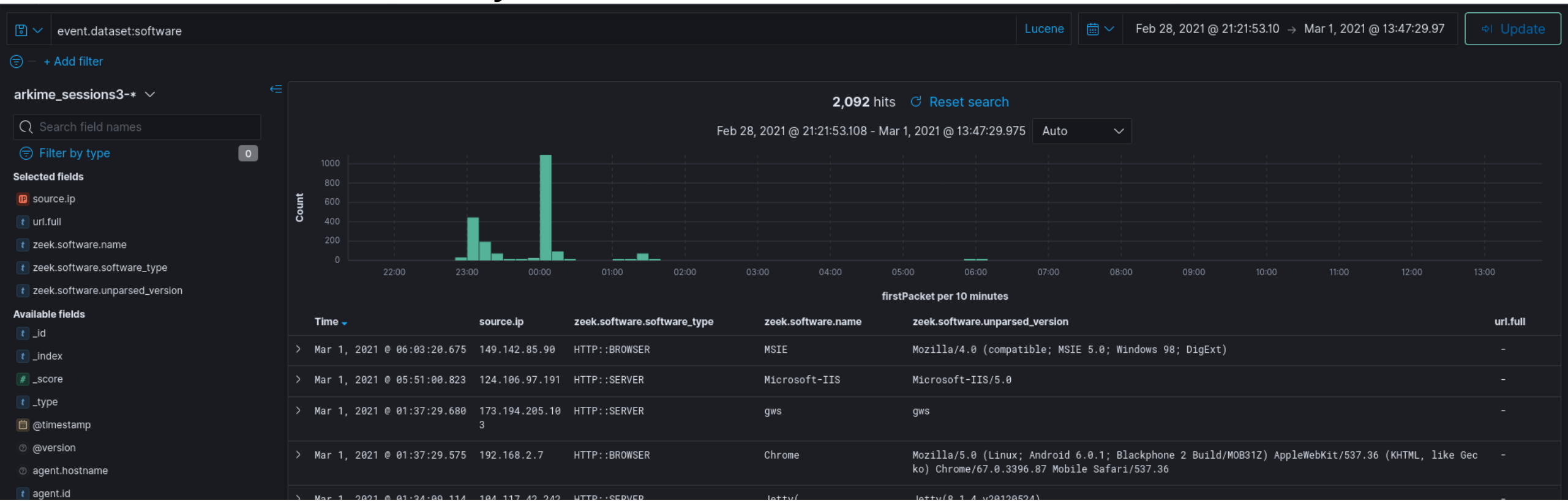
Protocol Dashboards

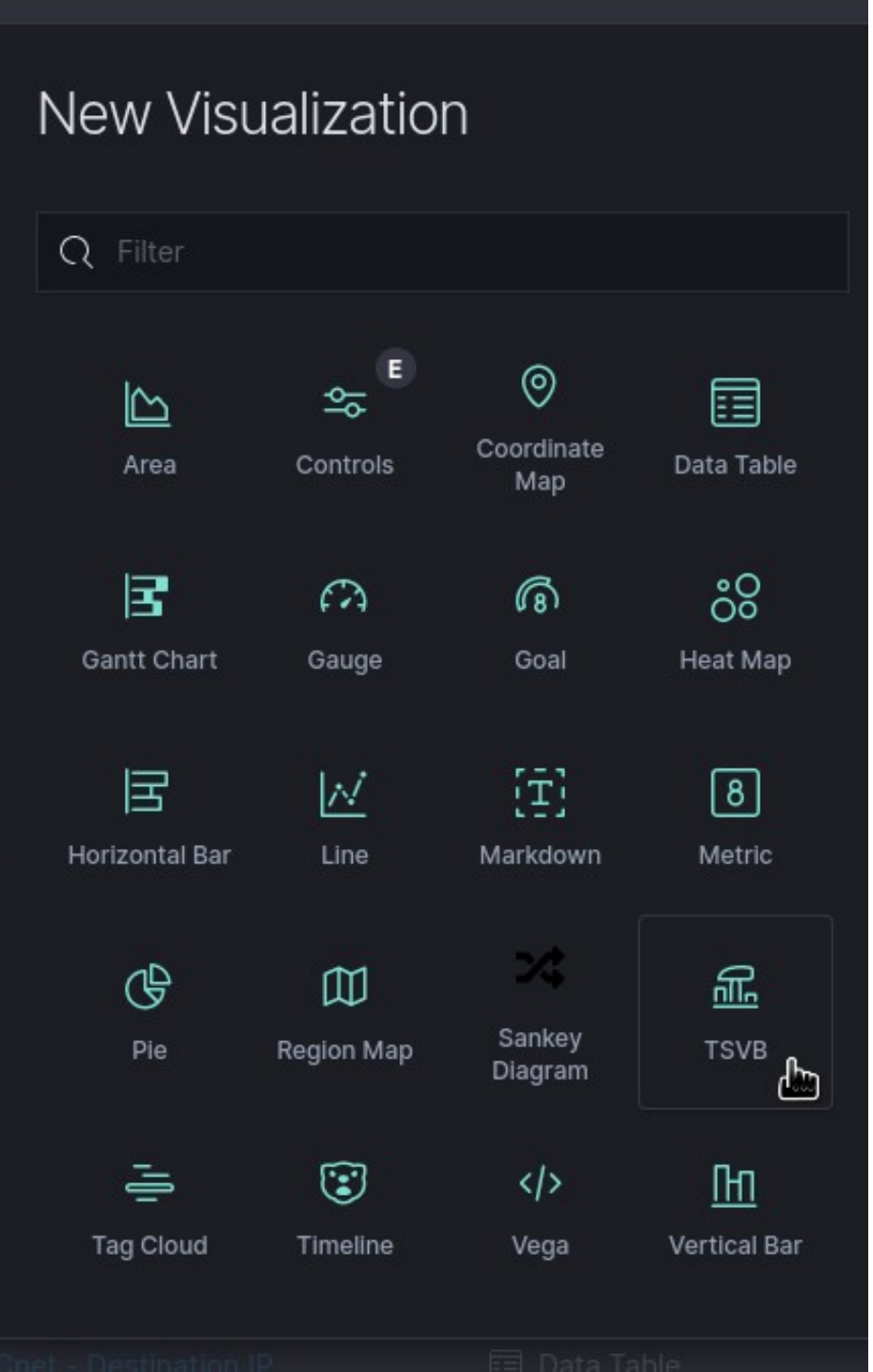
- Highlight application-specific fields of interest
- Grouped by common IT protocols and ICS/IoT protocols
- ICS protocols
  - BACnet
  - BSAP
  - DNP3
  - EtherCAT
  - EtherNet/IP
  - Modbus
  - PROFINET
  - S7comm



Discover

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





Custom Visualizations

- Create new visualizations from scratch or based on existing charts or dashboards

Search Syntax Comparison

	Arkime	Dashboards (Lucene)	Dashboards (DQL)
Field exists	`zeek.logType == EXISTS!`	`_exists_:zeek.logType`	`zeek.logType:*`
Field does not exist	`zeek.logType != EXISTS!`	`NOT _exists_:zeek.logType`	`NOT zeek.logType:*`
Field matches a value	`port.dst == 22`	`dstPort:22`	`dstPort:22`
Field does not match a value	`port.dst != 22`	`NOT dstPort:22`	`NOT dstPort:22`
Field matches at least one of a list of values	`tags == [external_source, external_destination]`	`tags:(external_source OR external_destination)`	`tags:(external_source or external_destination)`
Field range (inclusive)	`http.statuscode >= 200 && http.statuscode <= 300`	`http.statuscode:[200 TO 300]`	`http.statuscode >= 200 and http.statuscode <= 300`

Search Syntax Comparison (cont.)

	Arkime	Dashboards (Lucene)	Dashboards (DQL)
Field range (exclusive)	`http.statuscode > 200 && http.statuscode < 300`	`http.statuscode:{200 TO 300}`	`http.statuscode > 200 and http.statuscode < 300`
Field range (mixed exclusivity)	`http.statuscode >= 200 && http.statuscode < 300`	`http.statuscode:[200 TO 300}`	`http.statuscode >= 200 and http.statuscode < 300`
Match all search terms (AND)	`(tags == [external_source, external_destination]) && (http.statuscode == 401)`	`tags:(external_source OR external_destination) AND http.statuscode:401`	`tags:(external_source or external_destination) and http.statuscode:401`
Match any search terms (OR)	`(zeek_ftp.password == EXISTS!) || (zeek_http.password == EXISTS!) || (zeek.user == "anonymous")`	`_exists_:zeek_ftp.password OR _exists_:zeek_http.password OR zeek.user:"anonymous"`	`zeek_ftp.password:* or zeek_http.password:* or zeek.user:"anonymous"`

Search Syntax Comparison (cont.)

	Arkime	Dashboards (Lucene)	Dashboards (DQL)
Global string search (anywhere in the document)	all Arkime search expressions are field-based	`microsoft`	`microsoft`
Wildcards	`host.dns == "*micro?oft*"` (`?` for single character, `*` for any characters)	`dns.host:*micro?oft*` (`?` for single character, `*` for any characters)	`dns.host:*micro*ft*` (`*` for any characters)
Regex	`host.http == /.*www\.f.*k\.com.*/`	`zeek_http.host: /.*www\.f.*k\.com.*/`	Dashboards Query Language does not currently support regex
IPv4 values	`ip == 0.0.0.0/0`	`srcIp:"0.0.0.0/0" OR dstIp:"0.0.0.0/0"`	`srcIp:"0.0.0.0/0" OR dstIp:"0.0.0.0/0"`
IPv6 values	`(ip.src == EXISTS! || ip.dst == EXISTS!) && (ip != 0.0.0.0/0)`	`(_exists_:srcIp AND NOT srcIp:"0.0.0.0/0") OR (_exists_:dstIp AND NOT dstIp:"0.0.0.0/0")`	`(srcIp:* and not srcIp:"0.0.0.0/0") or (dstIp:* and not dstIp:"0.0.0.0/0")`

Search Syntax Comparison (cont.)

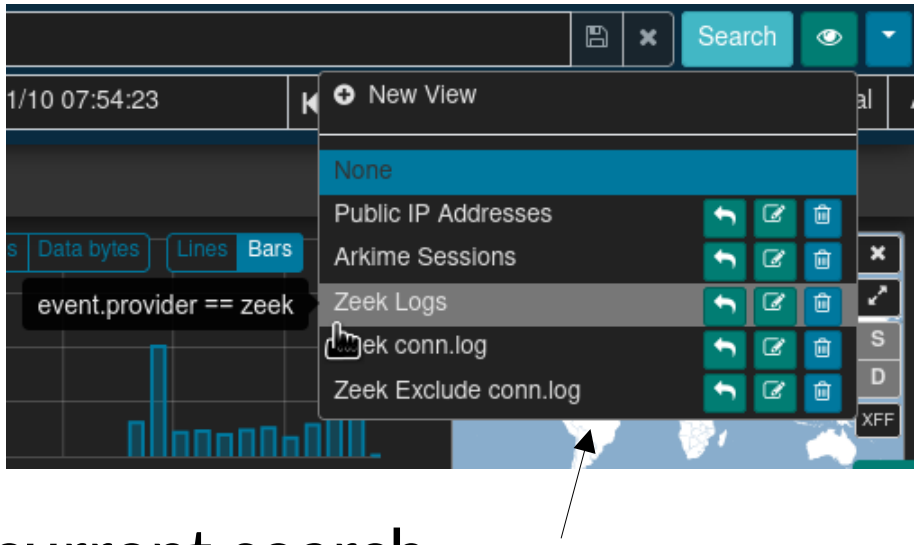
	Arkime	Dashboards (Lucene)	Dashboards (DQL)
GeoIP information available	`country == EXISTS!`	`_exists_:zeek.destination_geo OR _exists_:zeek.source_geo`	`zeek.destination_geo:* or zeek.source_geo:*`
Zeek log type	`zeek.logType == notice`	`zeek.logType:notice`	`zeek.logType:notice`
IP CIDR Subnets	`ip.src == 172.16.0.0/12`	`srcIp:"172.16.0.0/12"`	`srcIp:"172.16.0.0/12"`
Search time frame	Use Arkime time bounding controls under the search bar	Use Dashboards time range controls in the upper right-hand corner	Use Dashboards time range controls in the upper right-hand corner
GeoIP information available	`country == EXISTS!`	`_exists_:zeek.destination_geo OR _exists_:zeek.source_geo`	`zeek.destination_geo:* or zeek.source_geo:*`

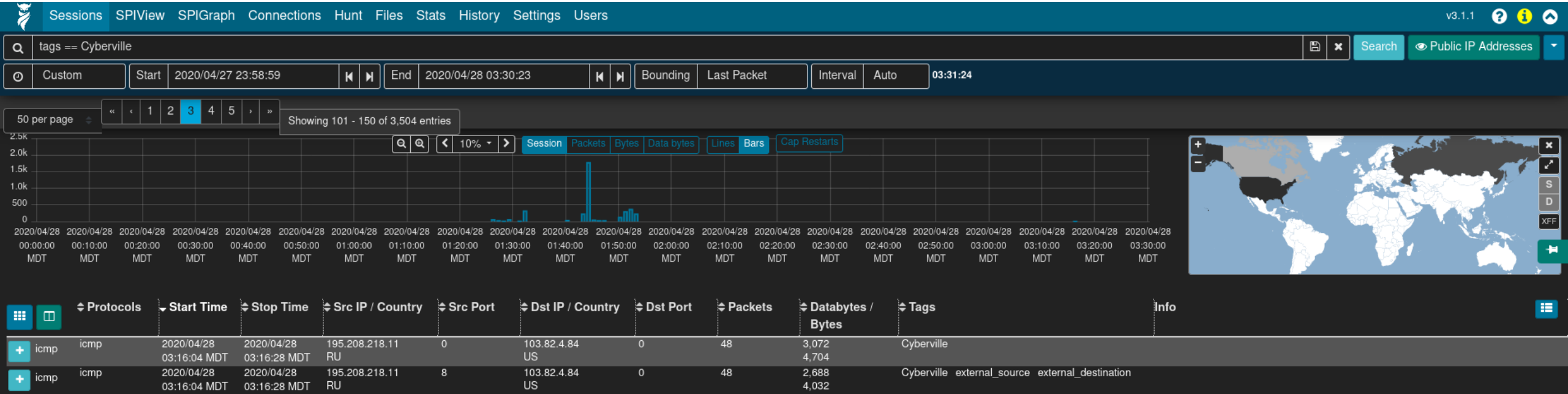
Arkime

- Front end for **both** enriched Zeek logs and Arkime sessions
  - Malcolm's custom Arkime Zeek data source adds full support for Zeek logs to Arkime, including ICS protocols
- Filter by Zeek logs or Arkime sessions; or, view both together
- "Wireshark at scale": full PCAP availability for
  - viewing packet payload
  - exporting filtered and joined PCAP sessions
  - running deep-packet searches
- https://localhost

Arkime Filters and Search

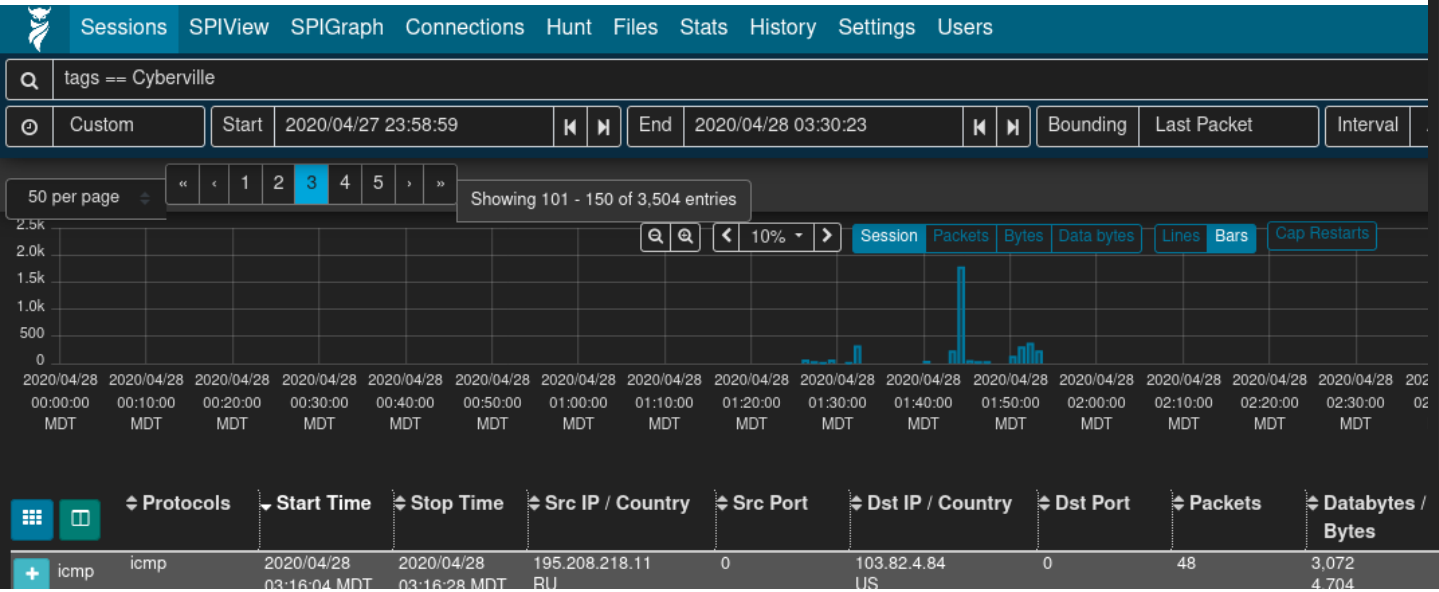
- Time filter: define search time frame
- Map filter: restrict results to geolocation
- Query bar: write queries in Arkime syntax
- Views: overlay previously-specified filters on current search

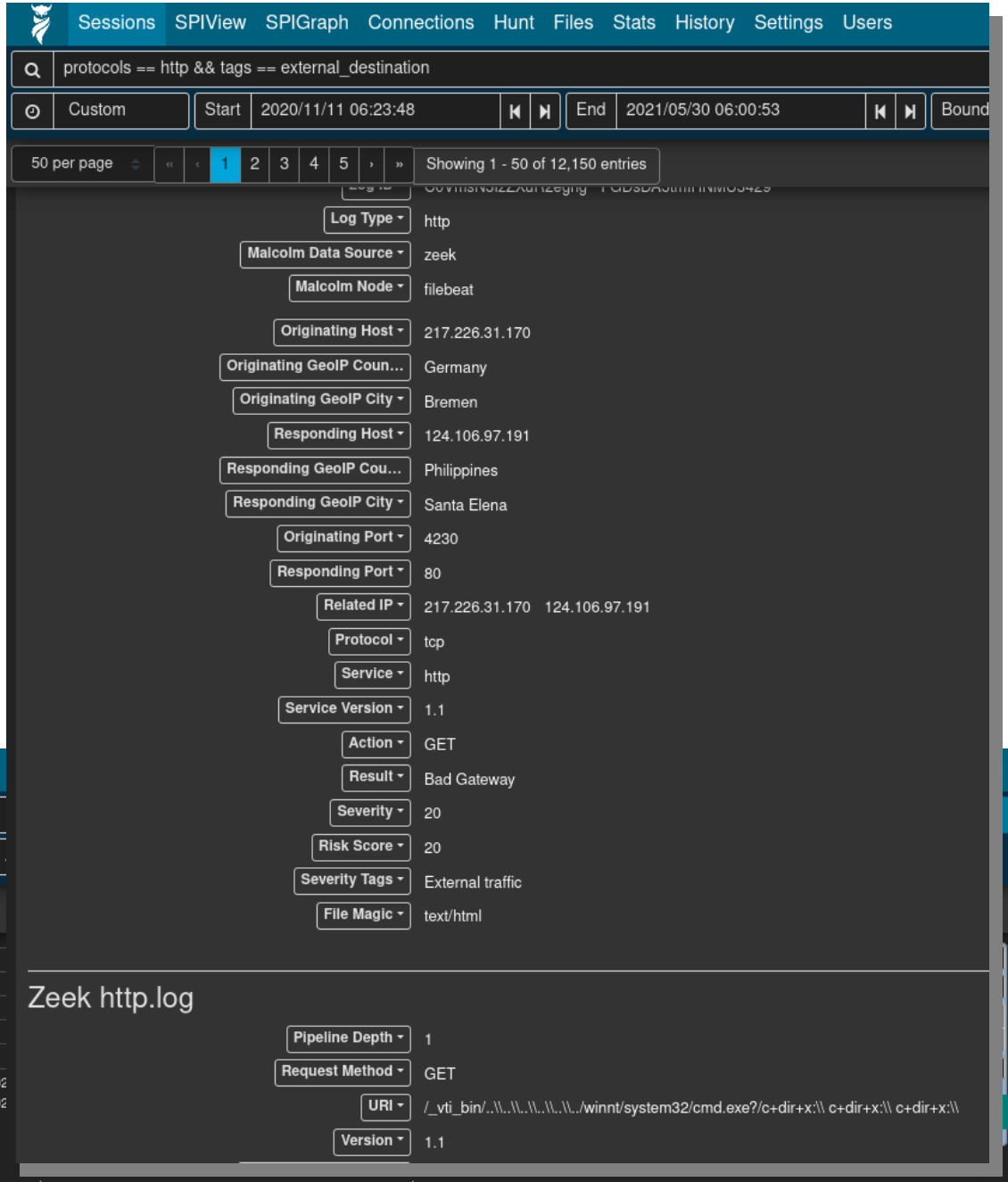




Sessions

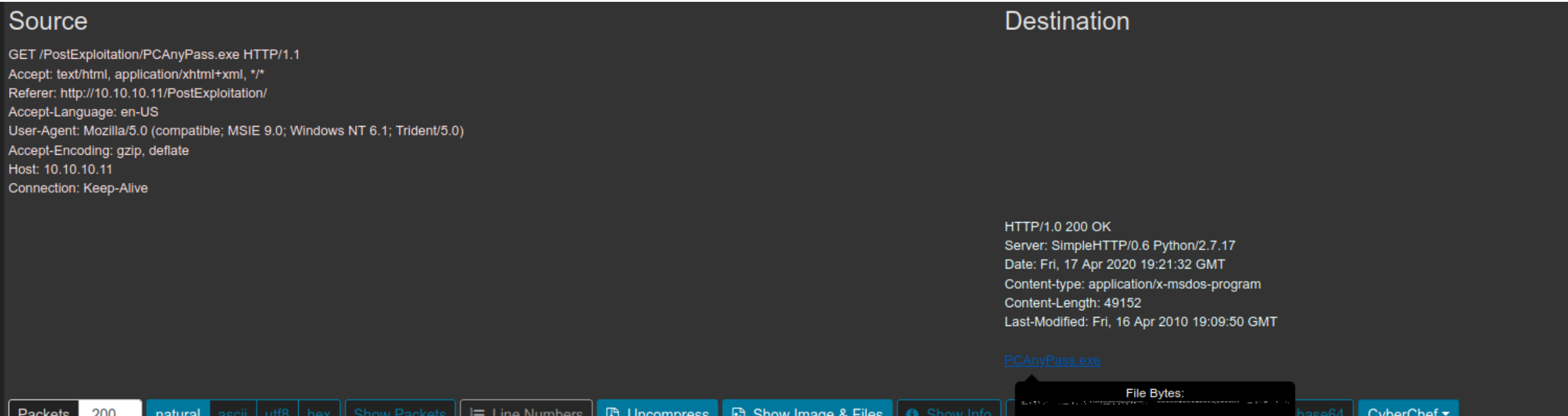
- Field-level details of sessions/logs matching filters
- Similar to Dashboards' Discover





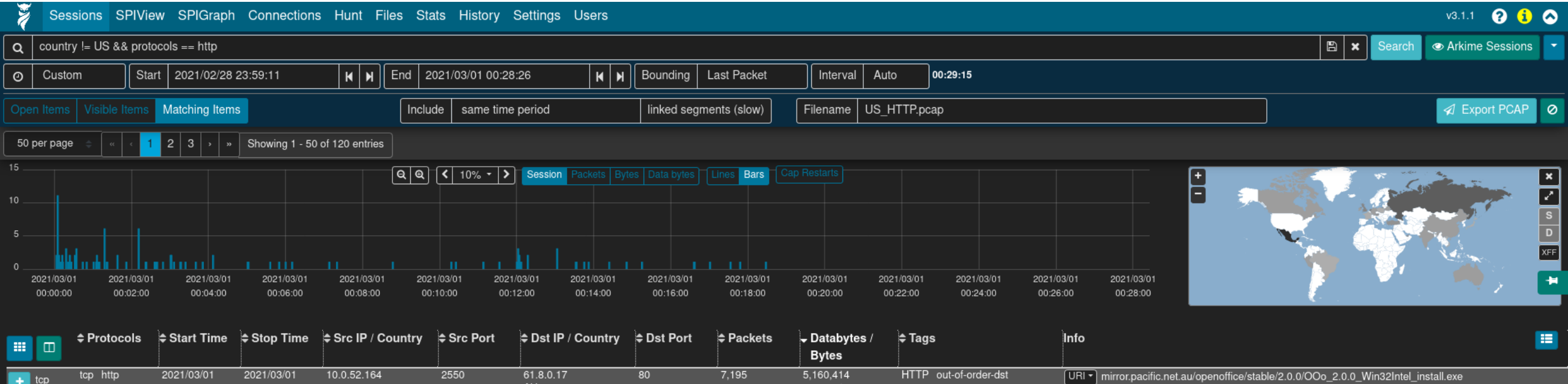
Packet Payloads

- Displayed for Arkime sessions with full PCAP (i.e., not Zeek logs)
- File carving on the fly
- Download session PCAP
- Examine payload with CyberChef



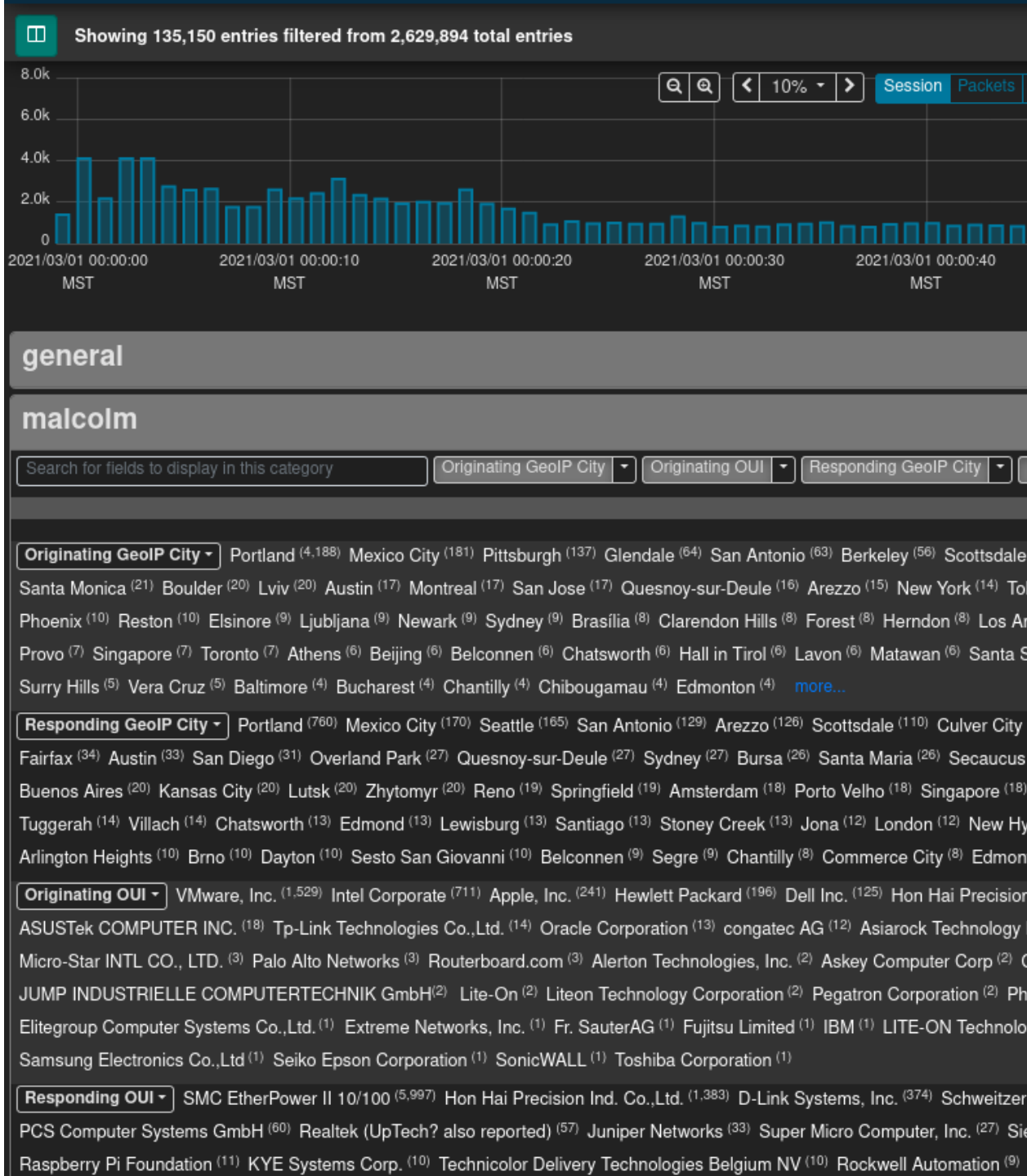
Export PCAP

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



SPIView

- Explore “top *n*” and field cardinality for all fields of both Arkime 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 (it runs many queries)

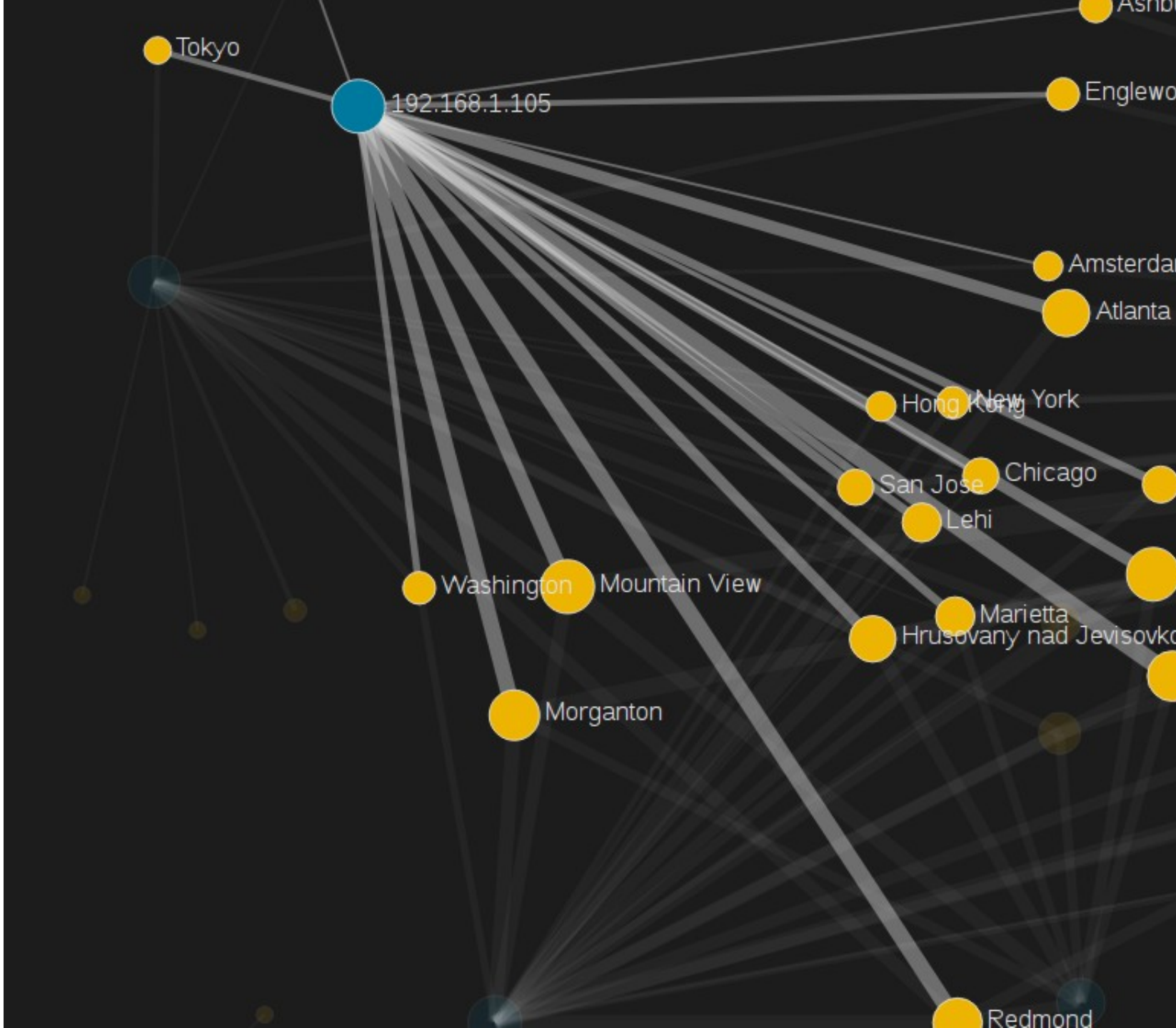


SPIGraph

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

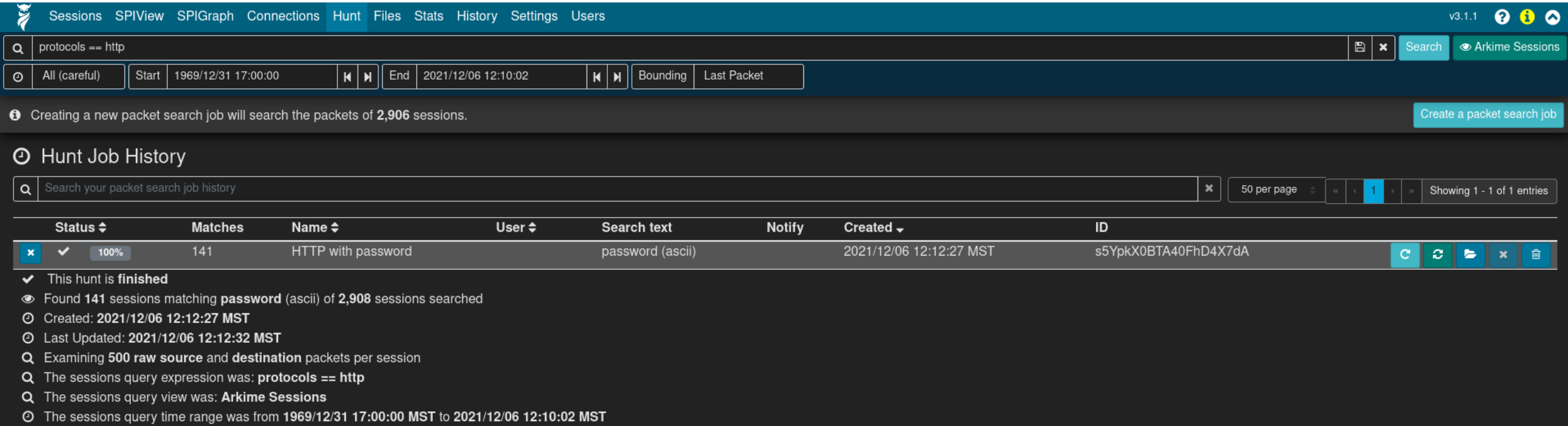
Connections

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



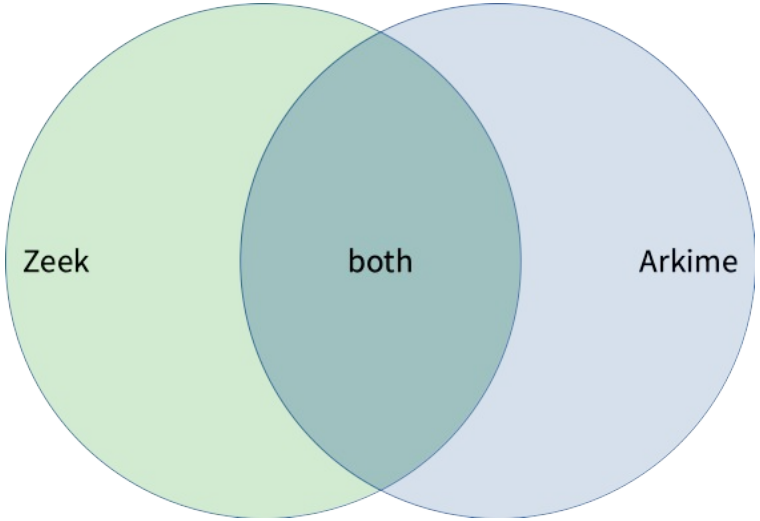
Packet Search (“Hunt”)

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



Data Source Correlation

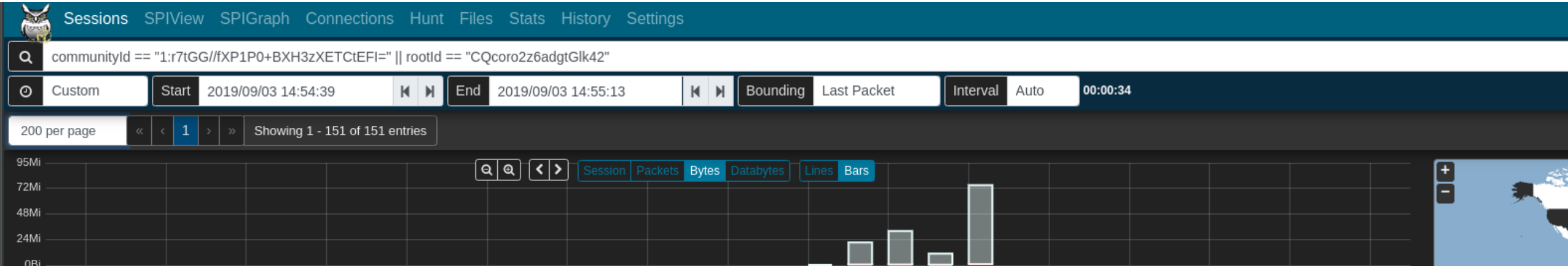
- Search syntax is different between Arkime and Dashboards (and in some cases, so are field names)
  - See search syntax comparison table, Malcolm and Arkime docs
- Despite considerable overlap, there are differences in protocol parser support between Zeek and Arkime
  - Learning the strengths of each will help you more effectively find the good stuff



Correlate Zeek Logs and Packet Payloads

- Correlate Zeek logs and Arkime sessions using common fields
- `communityId` fingerprints flows in both and can bridge the two
- `rootId` / `zeek.uid` filters Zeek logs for the same session
- Filter community ID OR'ed with Zeek UID to see all Arkime sessions and Zeek logs for the same traffic

```
communityId == "1:r7tGG//fXP1P0+BXH3zXETCtEFI=" || rootId == "CQcoro2z6adgtGlk42"
```

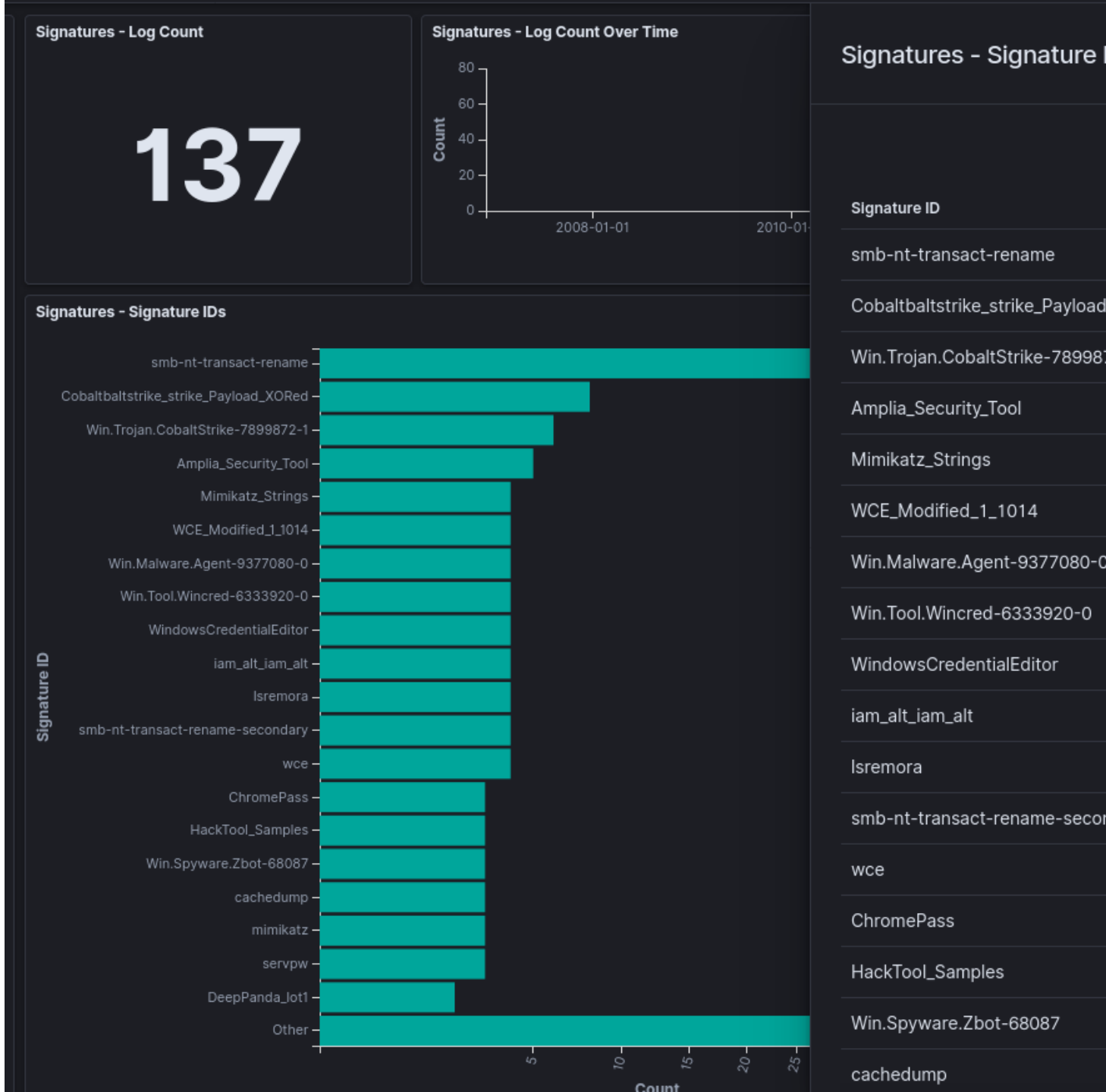


File Analysis

- Zeek can “carve” file transfers from common protocols
- Malcolm can examine carved files and flag hits
  - ClamAV – open source antivirus engine
  - YARA – pattern matching swiss army knife
  - Capa – portable executable capabilities analyzer
  - VirusTotal – online database of file hashes
    - requires API token and internet connection
- Triggering files can be saved to `zeek-logs/extract_files` under Malcolm directory for further analysis
  - Be careful! Carved files may contain live malware!

Signatures

- Signatures dashboard in Dashboards shows scanned file hits
- Use `zeek.fuid` field in *Signatures – Logs* table to pivot to connection UID (`zeek.uid`) and other logs with pertinent session details



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 Arkime and Zeek regardless of data source (e.g., `protocol:http` in Dashboards and `protocols == http` in Arkime)
- Use tags

Malcolm

Thank you!

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