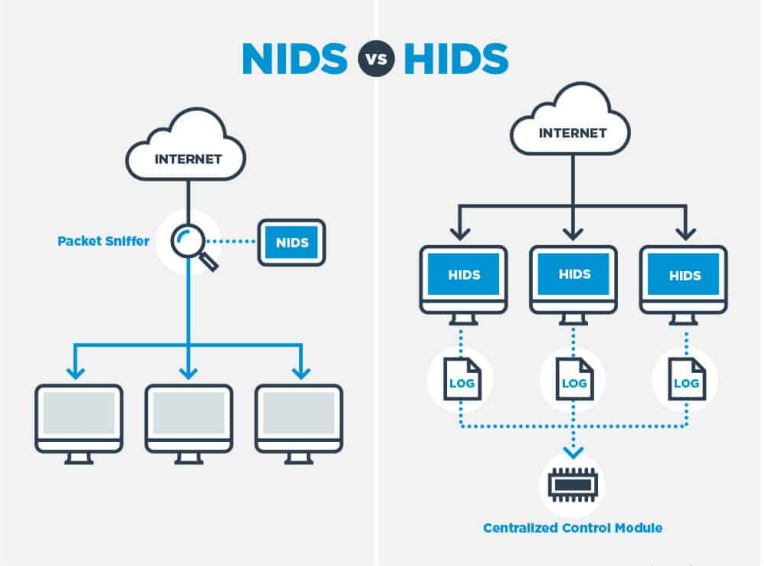
Network Traffic Analysis with Malcolm

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

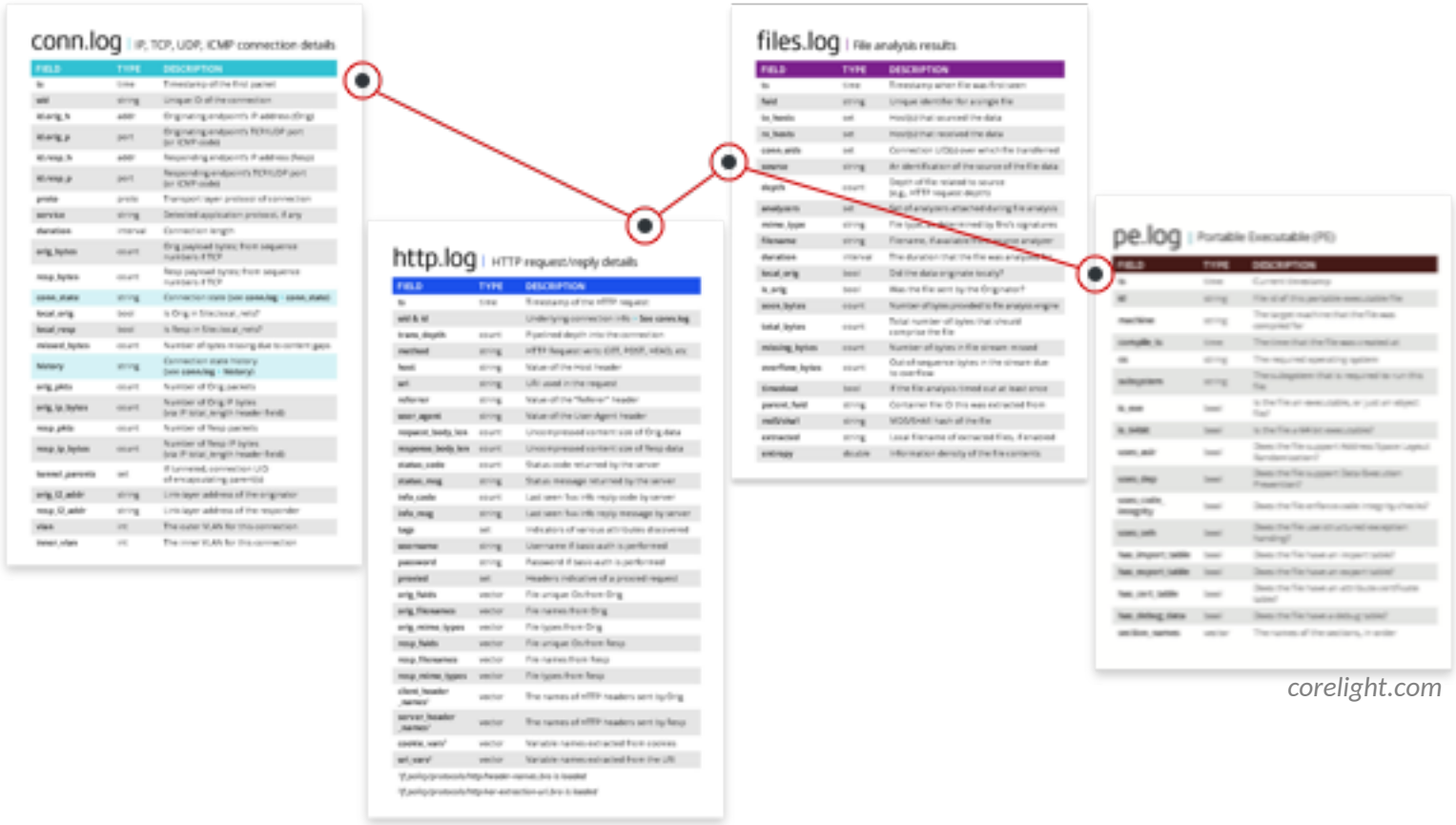
- Scanning Attack
  - Determine network topology
  - IDS highlights connections from one host to many other hosts in the network, or connection attempts to sequential IP addresses and/or ports
- Denial of Service Attack
  - Interrupt service by flooding requests or flaws in protocol implementations
  - 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

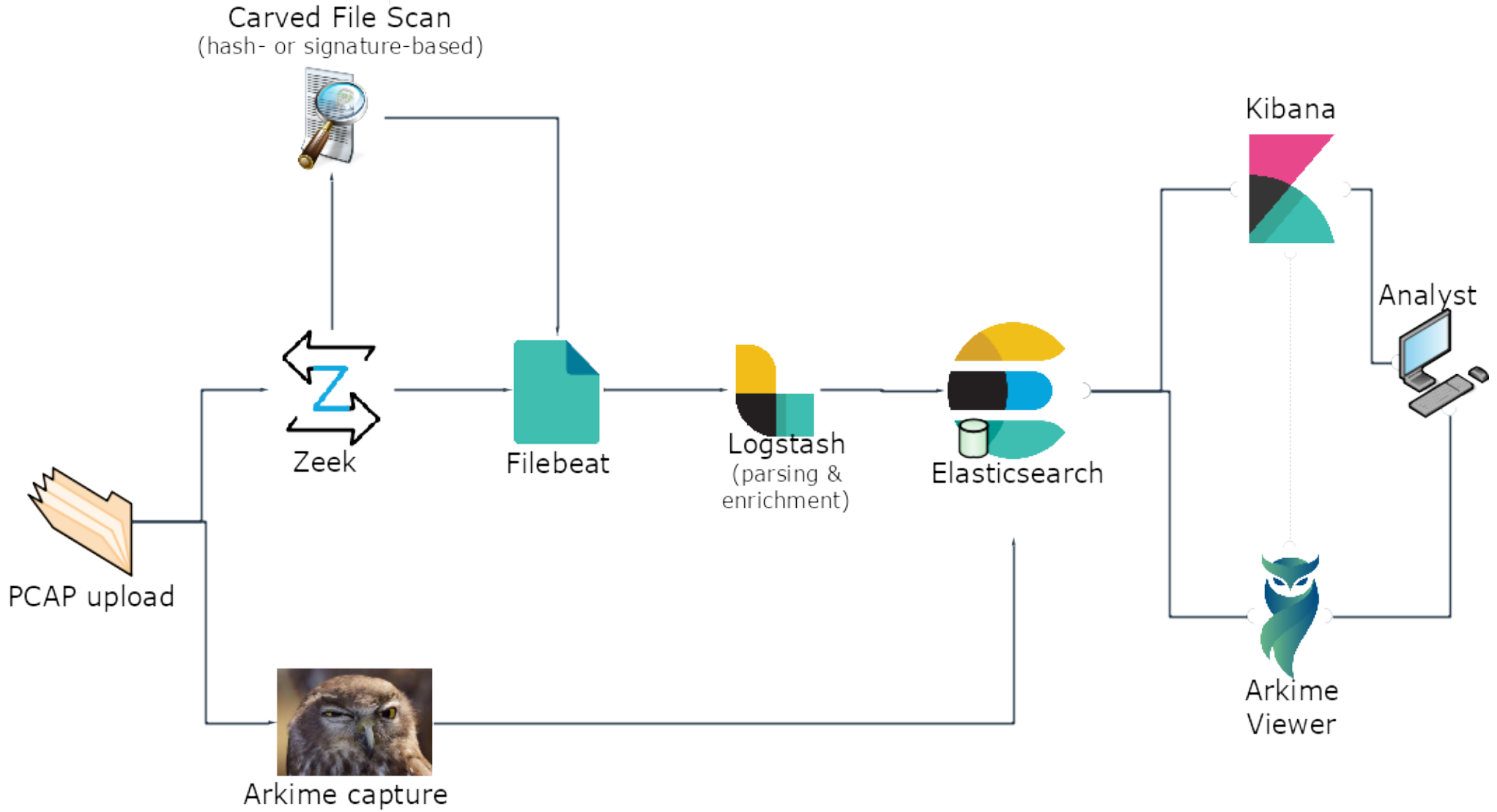
https://github.com/idaholab/Malcolm

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
OpenVPN
PostgreSQL
Process Field Net (PROFINET)
Remote Authentication Dial-In User Service (RADIUS)
Remote Desktop Protocol (RDP)
Remote Framebuffer (RFB / VNC)
S7comm / Connection Oriented Transport Protocol (COTP)
Session Initiation Protocol (SIP)
Server Message Block (SMB) / Common Internet File System (CIFS)
Simple Mail Transfer Protocol
Simple Network Management Protocol
SOCKS
Secure Shell (SSH)
Secure Sockets Layer (SSL) / Transport Layer Security (TLS)
Syslog
Tabular Data Stream
Telnet / remote shell (rsh) / remote login (rlogin)
TFTP (Trivial File Transfer Protocol)
WireGuard
tunnel protocols (e.g., GTP, GRE, Teredo, AYIYA, IP-in-IP, etc.)

elastic stack zeek Arkime CyberChef yara ClamAV docker NGINX …

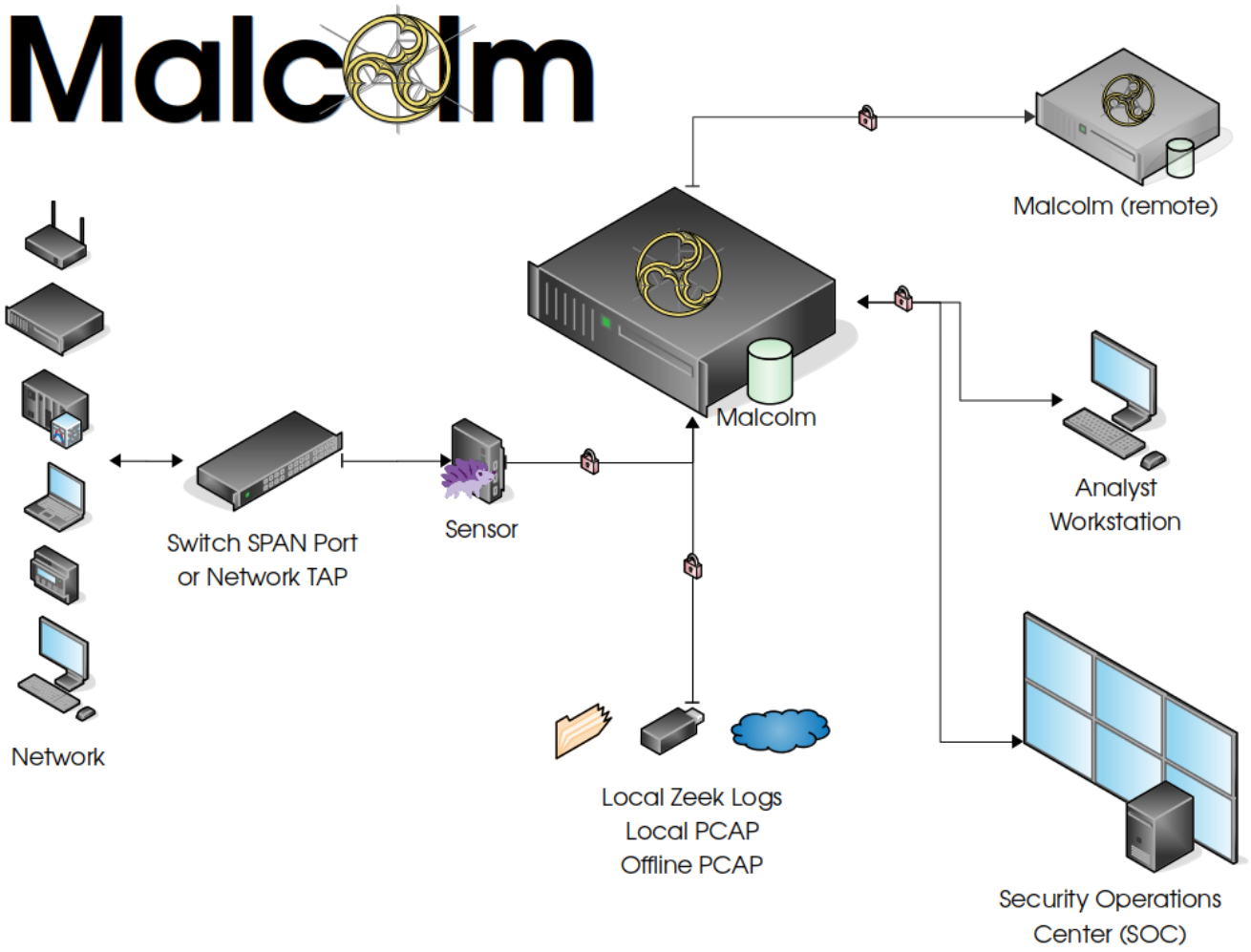
Malcolm

https://github.com/idaholab/Malcolm



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”

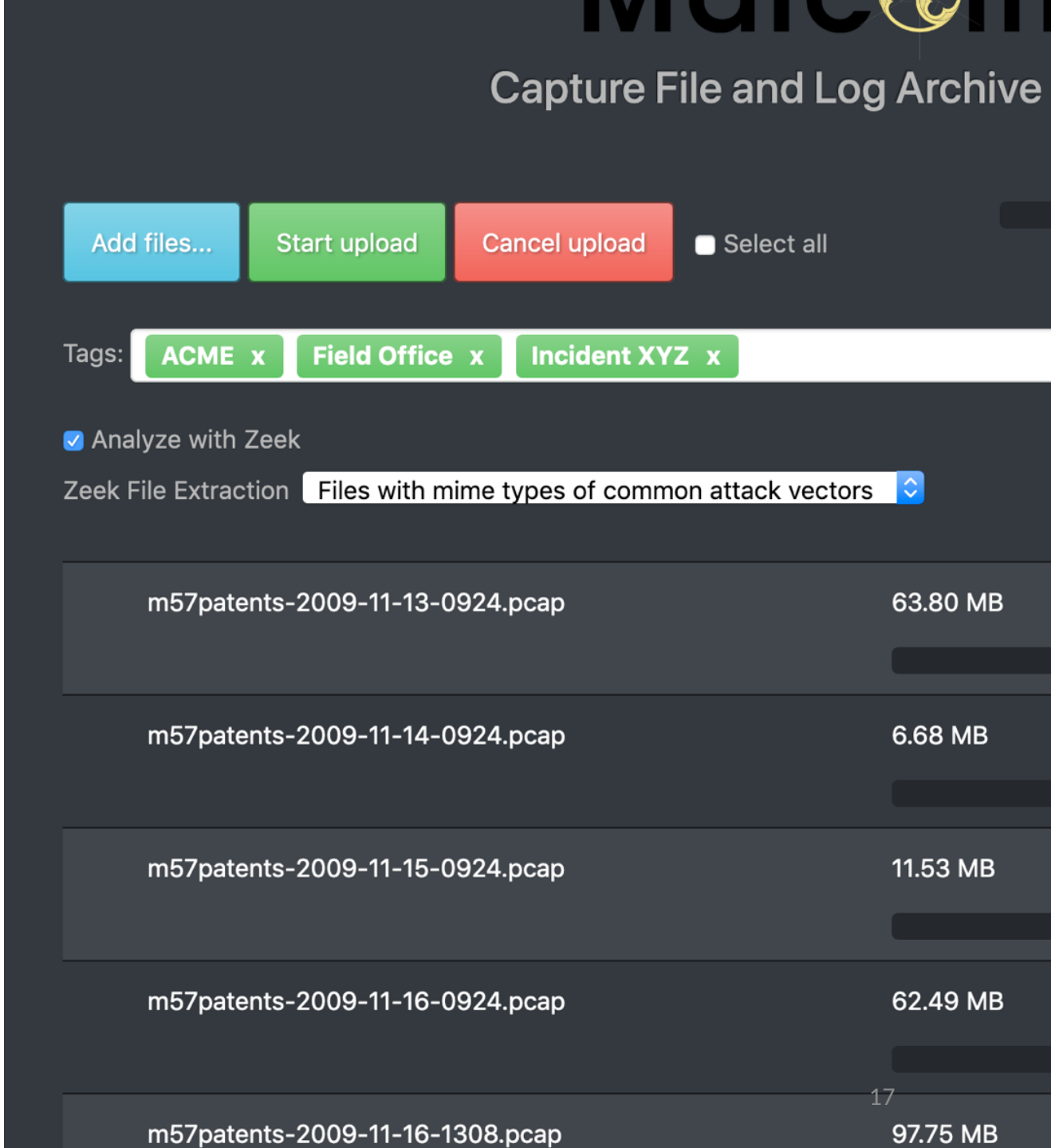


Identifying Network Hosts and Subnets

- 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



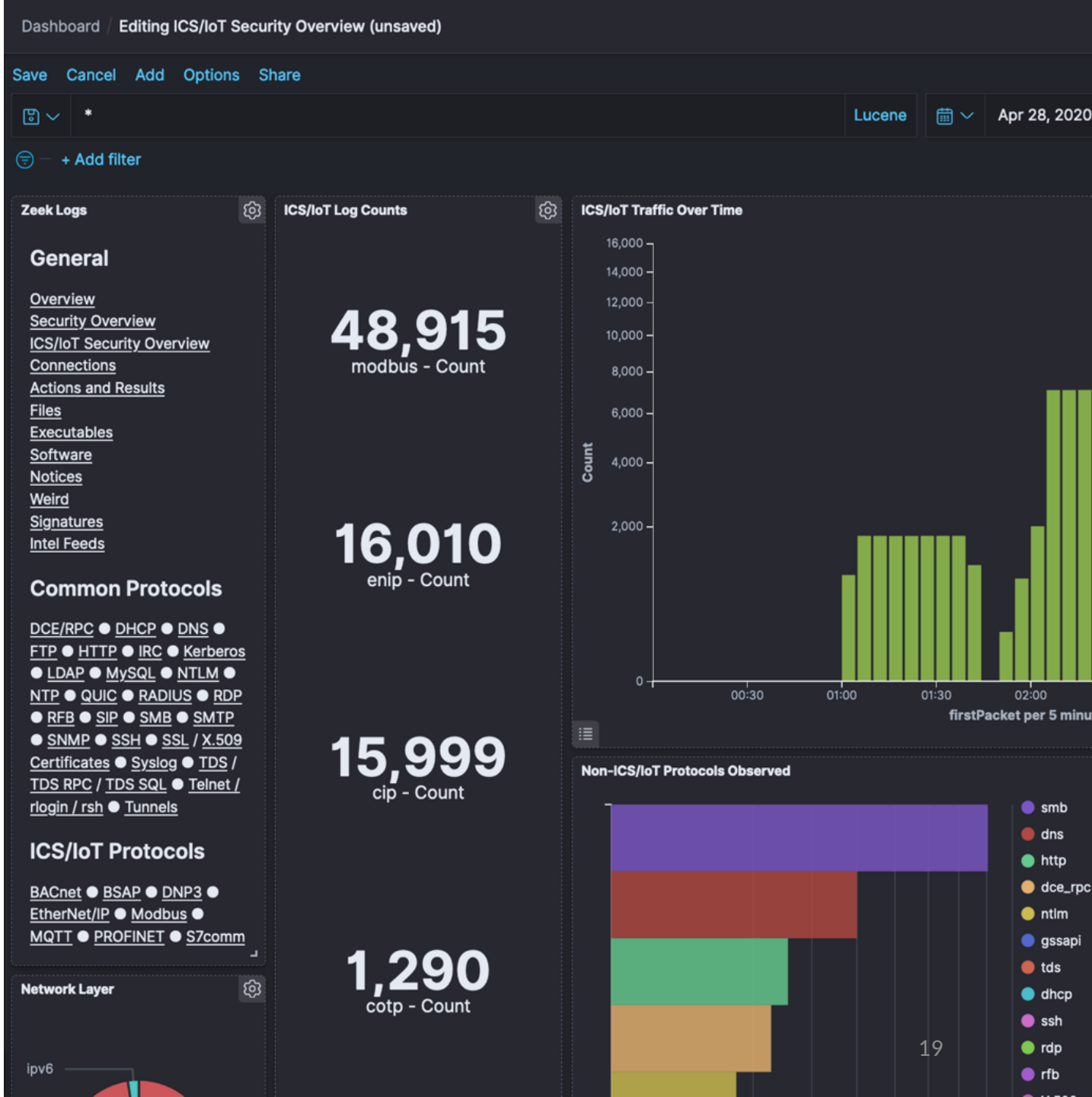
Data Tagging and Enrichment

- 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`

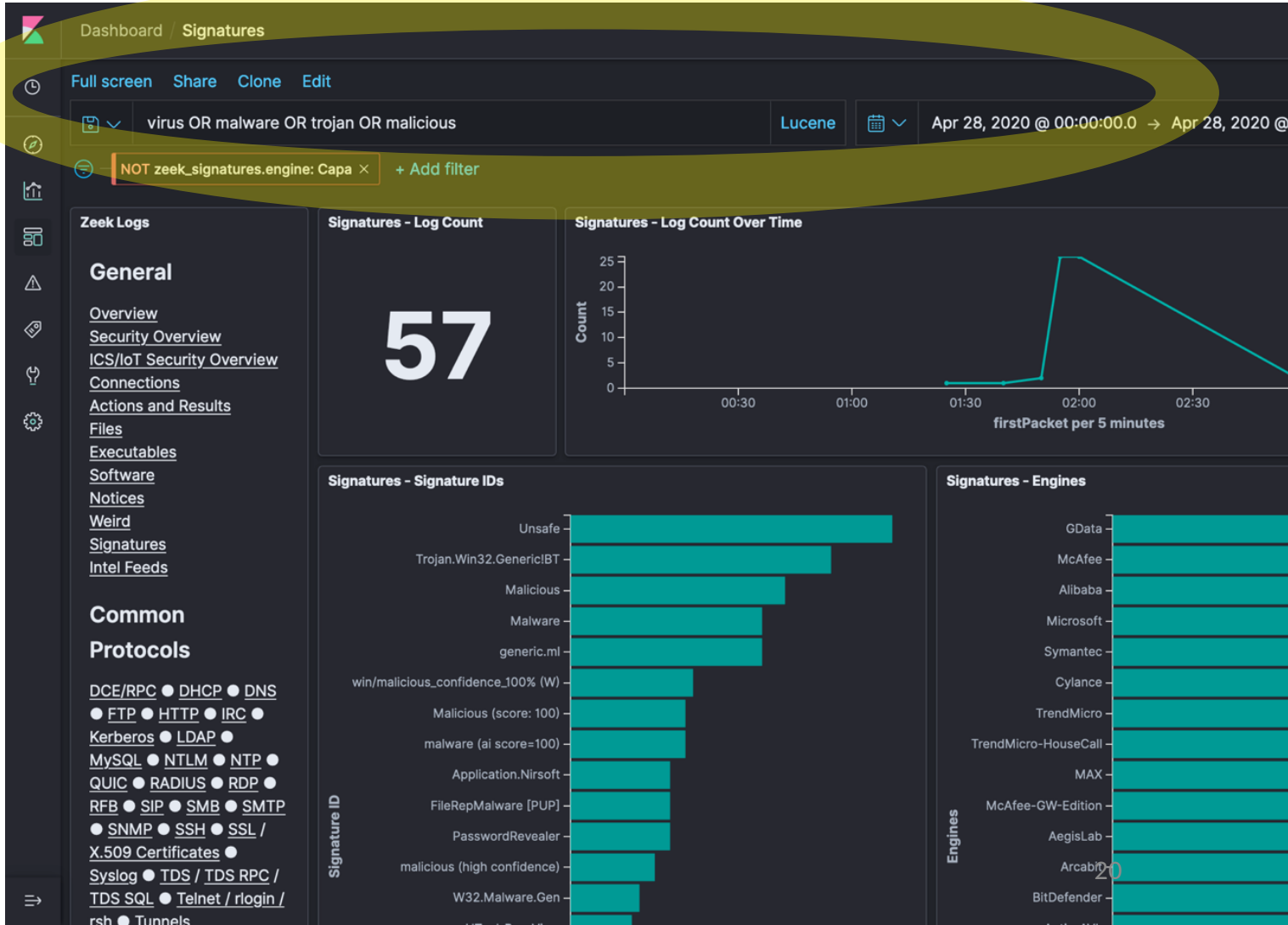
Kibana

- 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/kibana



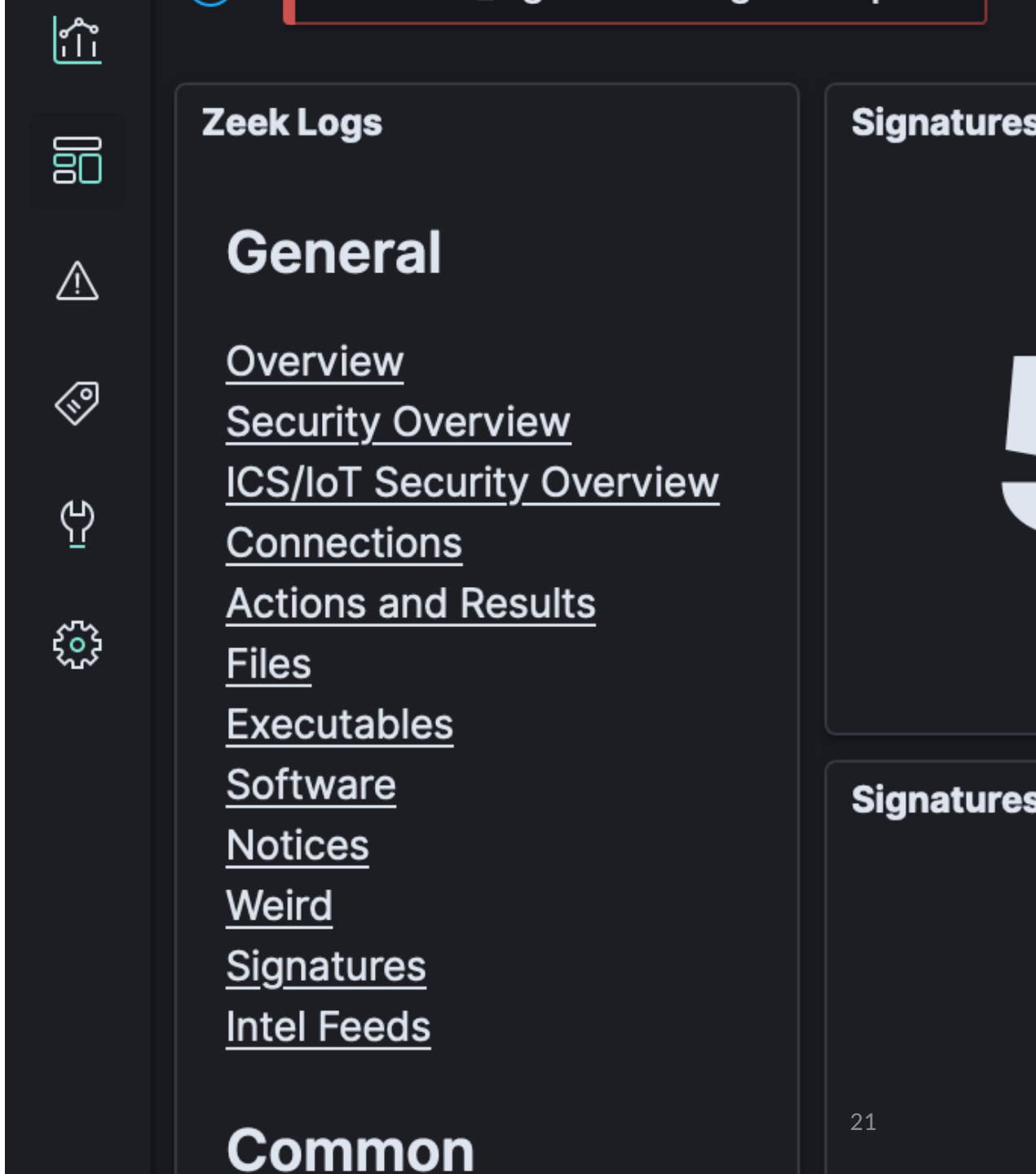
Kibana Filters and Search

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



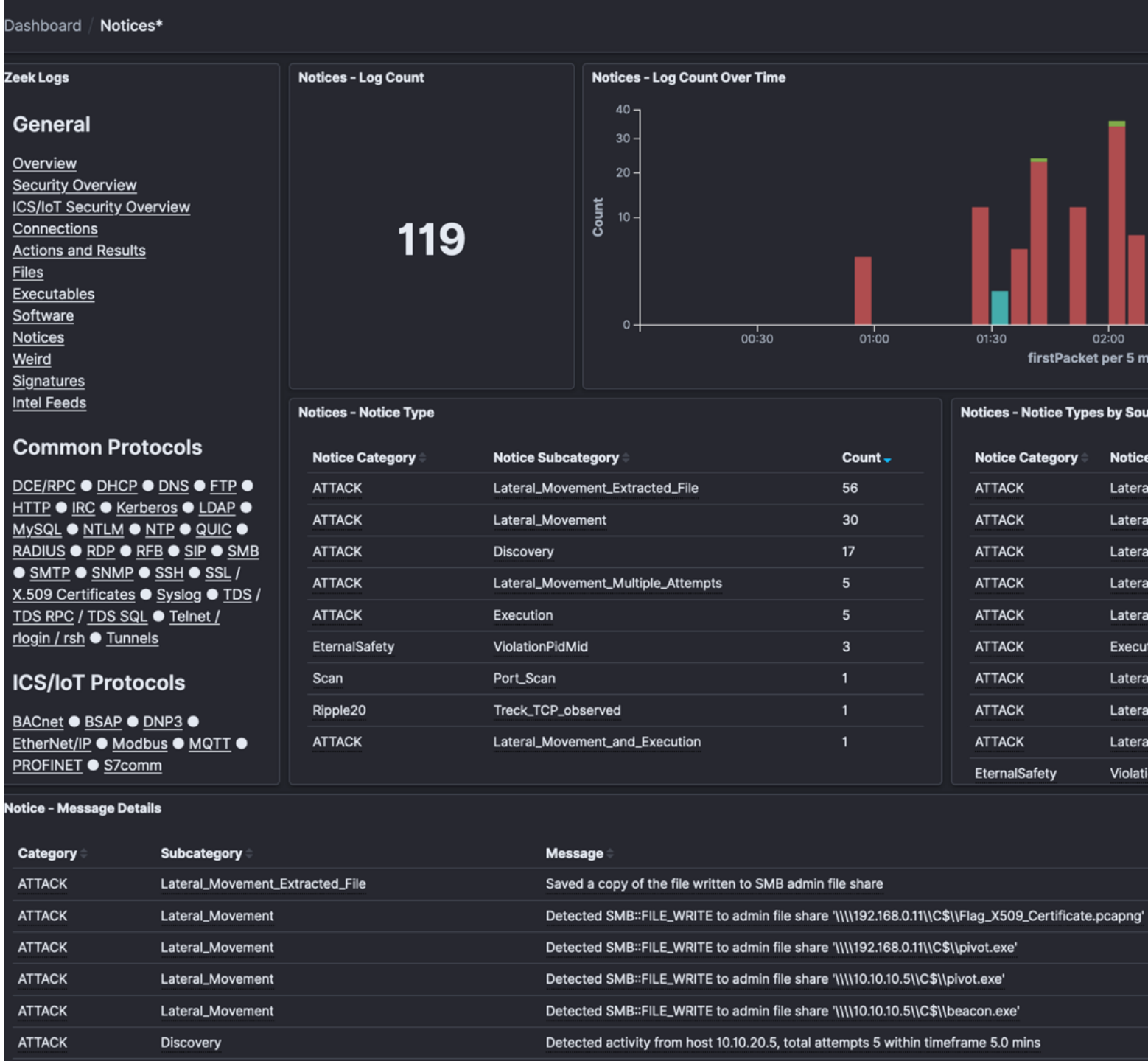
Overview Dashboards

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

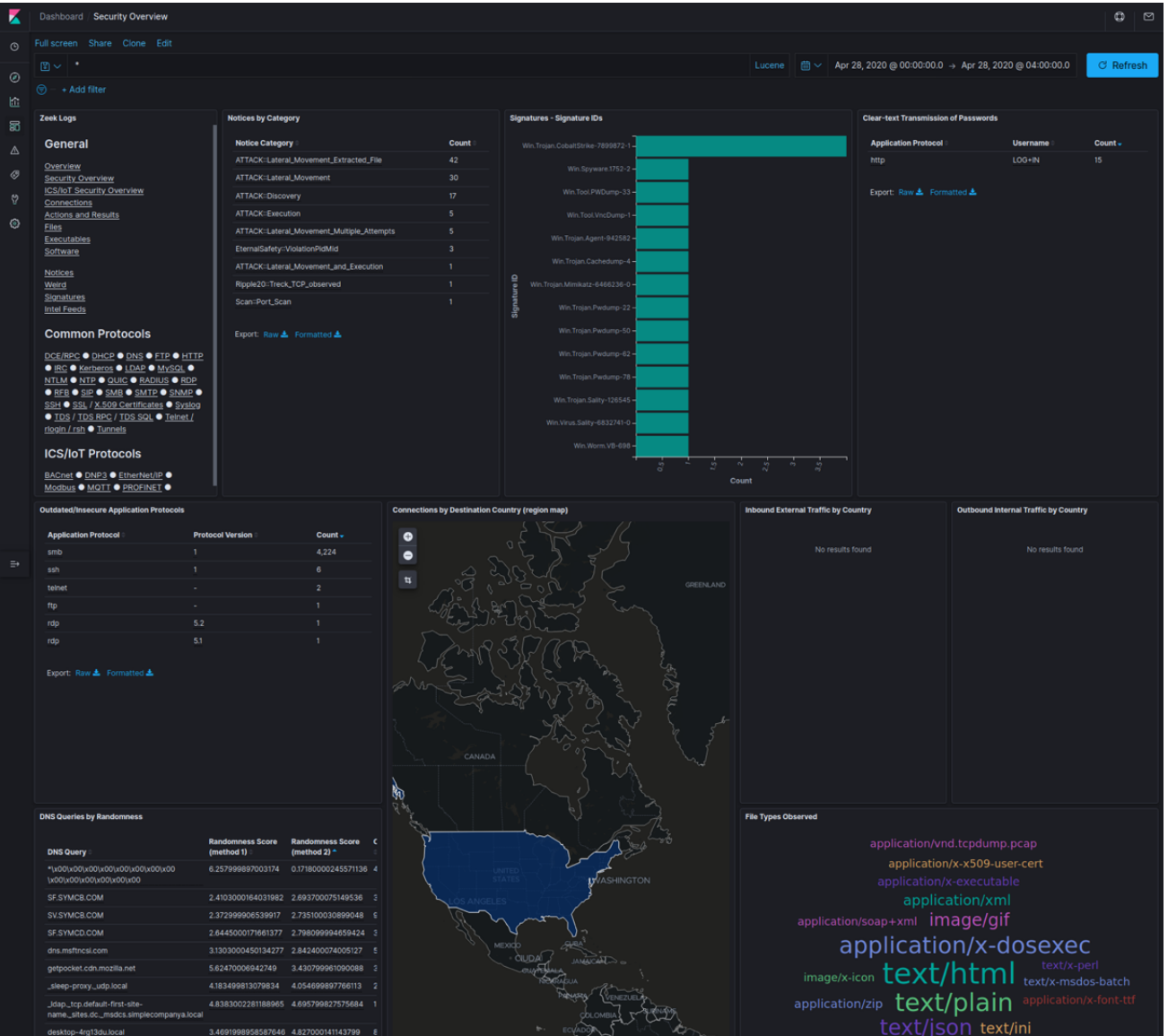


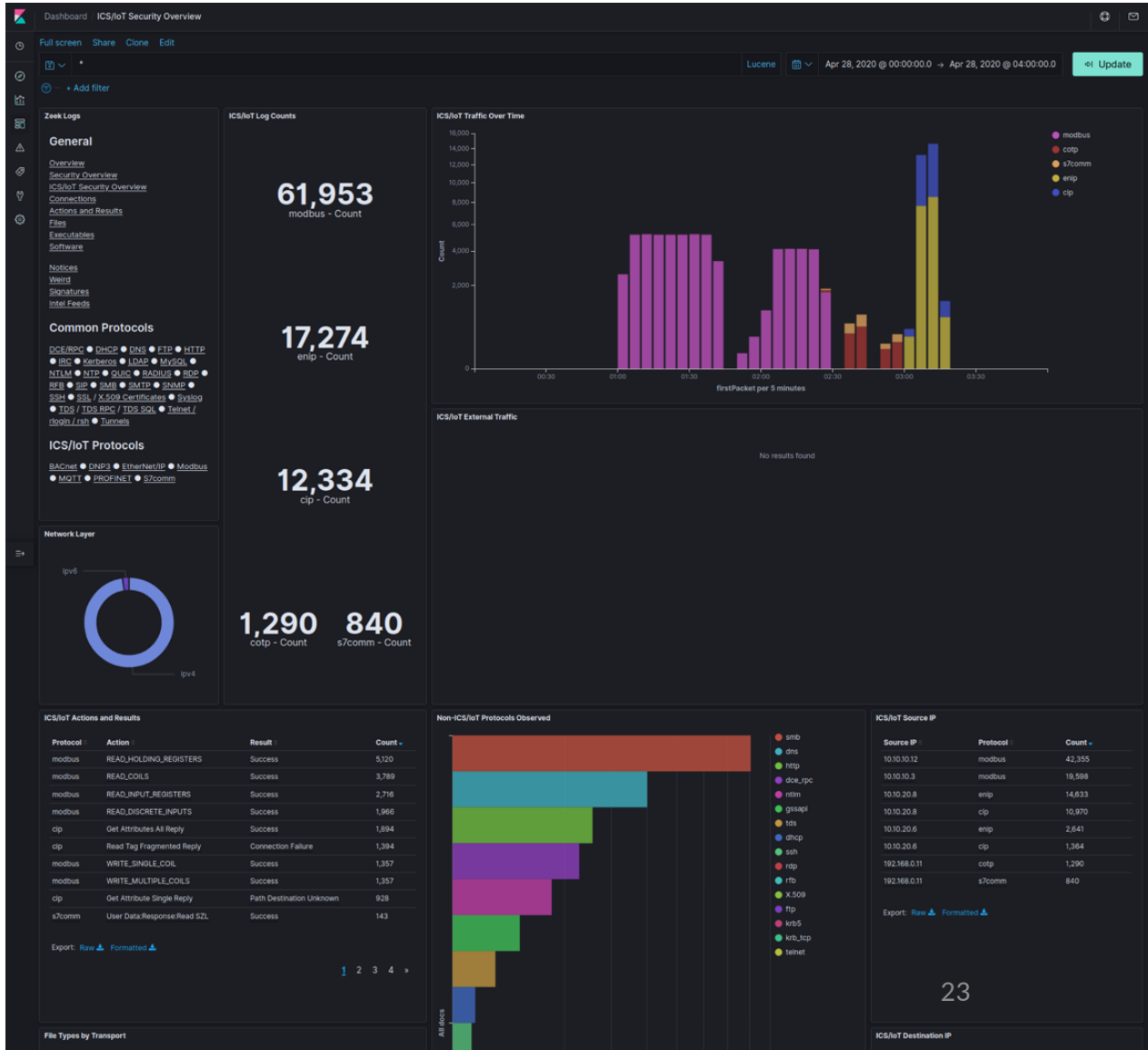
Notices

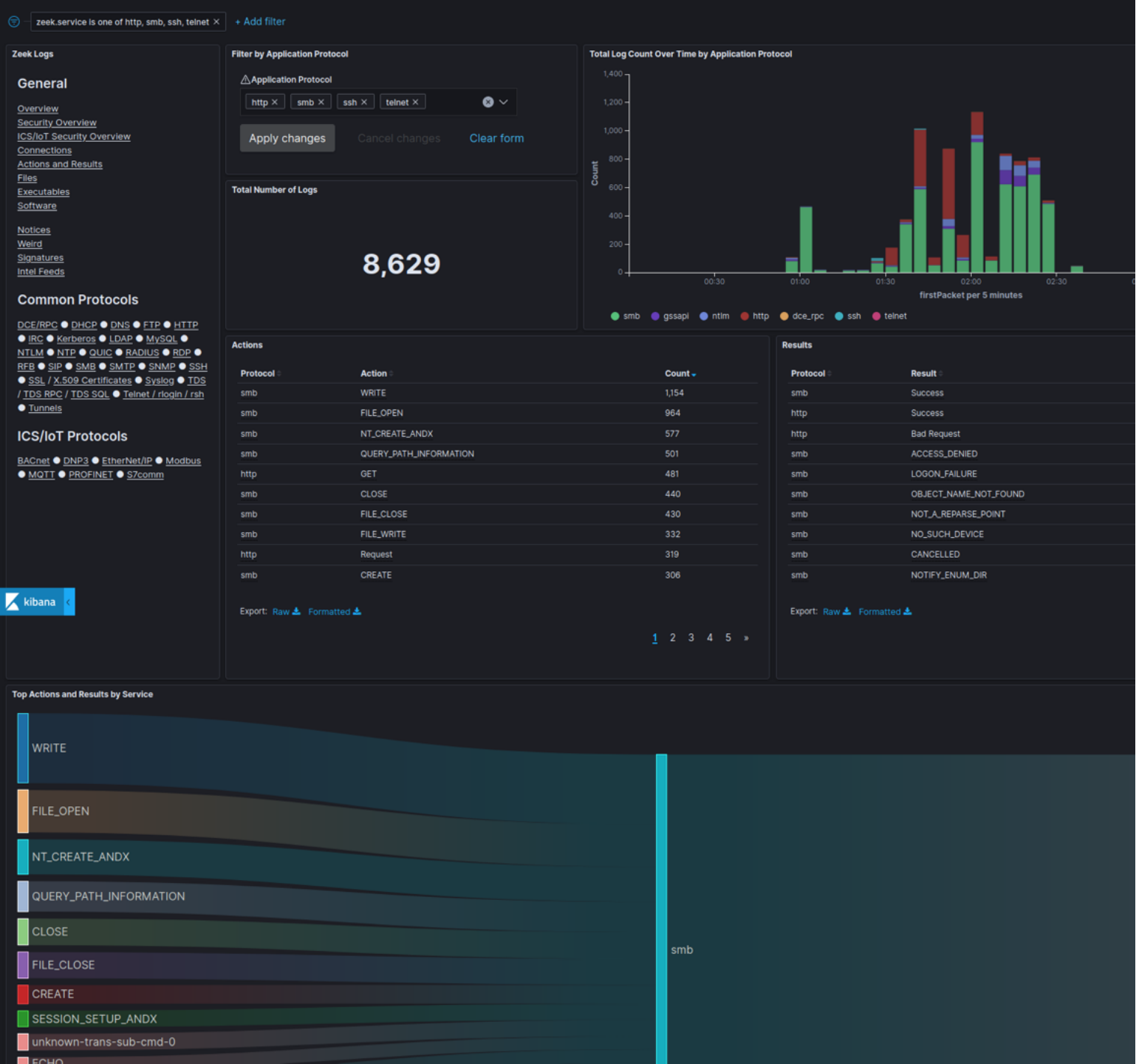
- Zeek notices are things that are odd or potentially bad
- In addition to Zeek's defaults, Malcolm raises notices for recent critical vulnerabilities and attack techniques



Security & ICS/IoT Security Overview





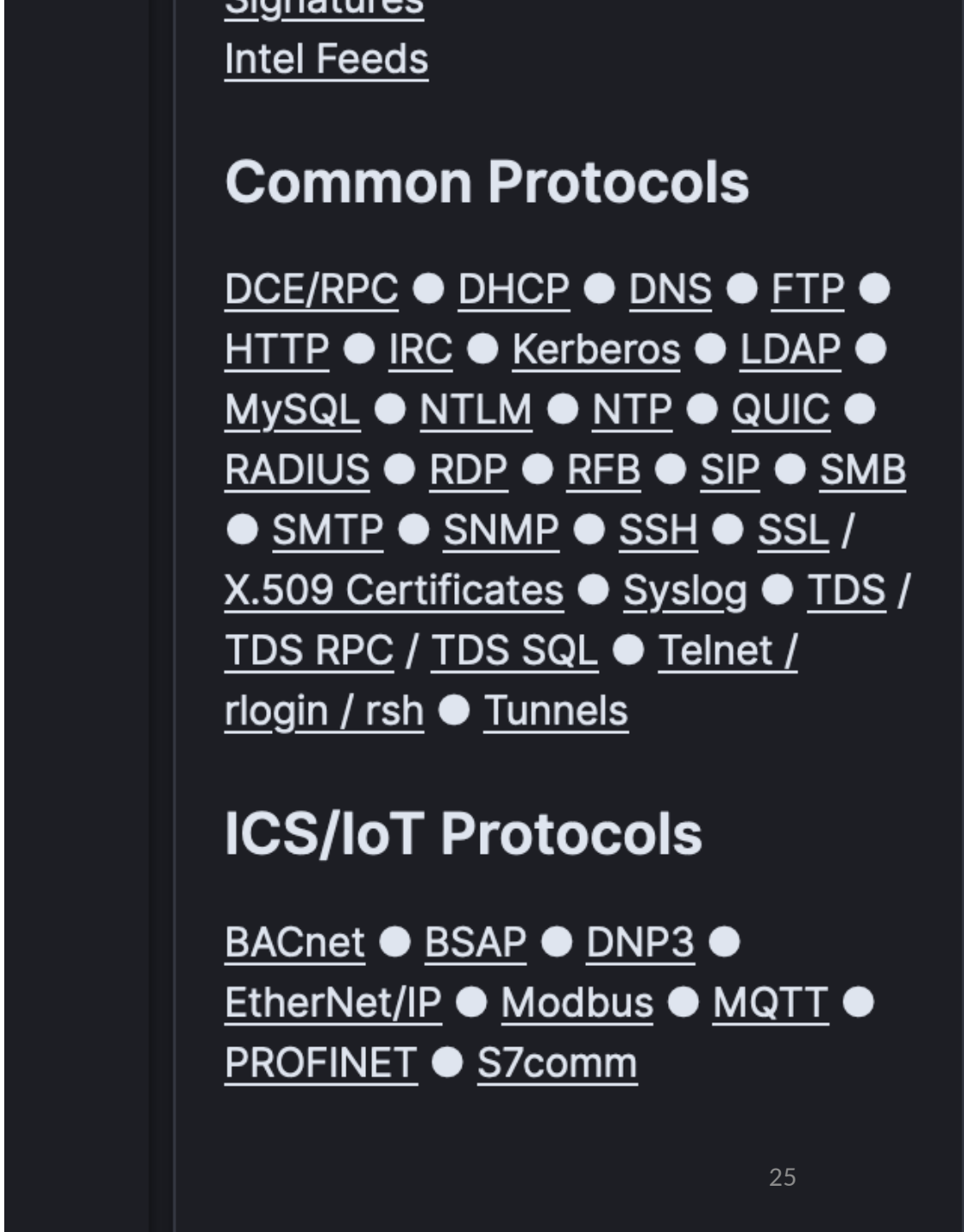


Actions and Results

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

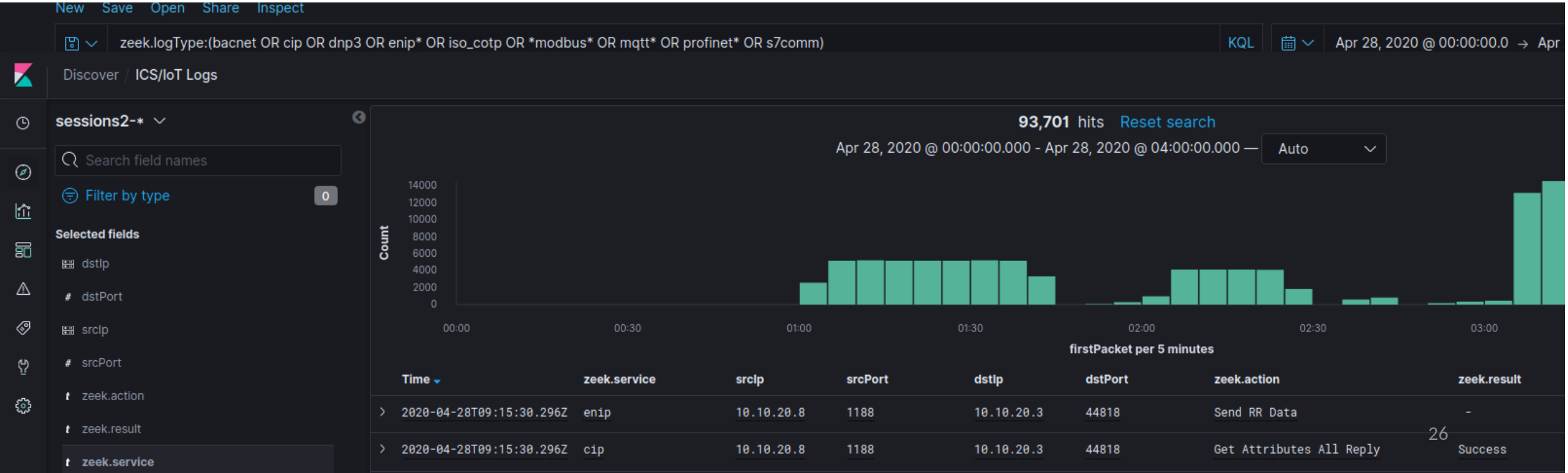
Protocol Dashboards

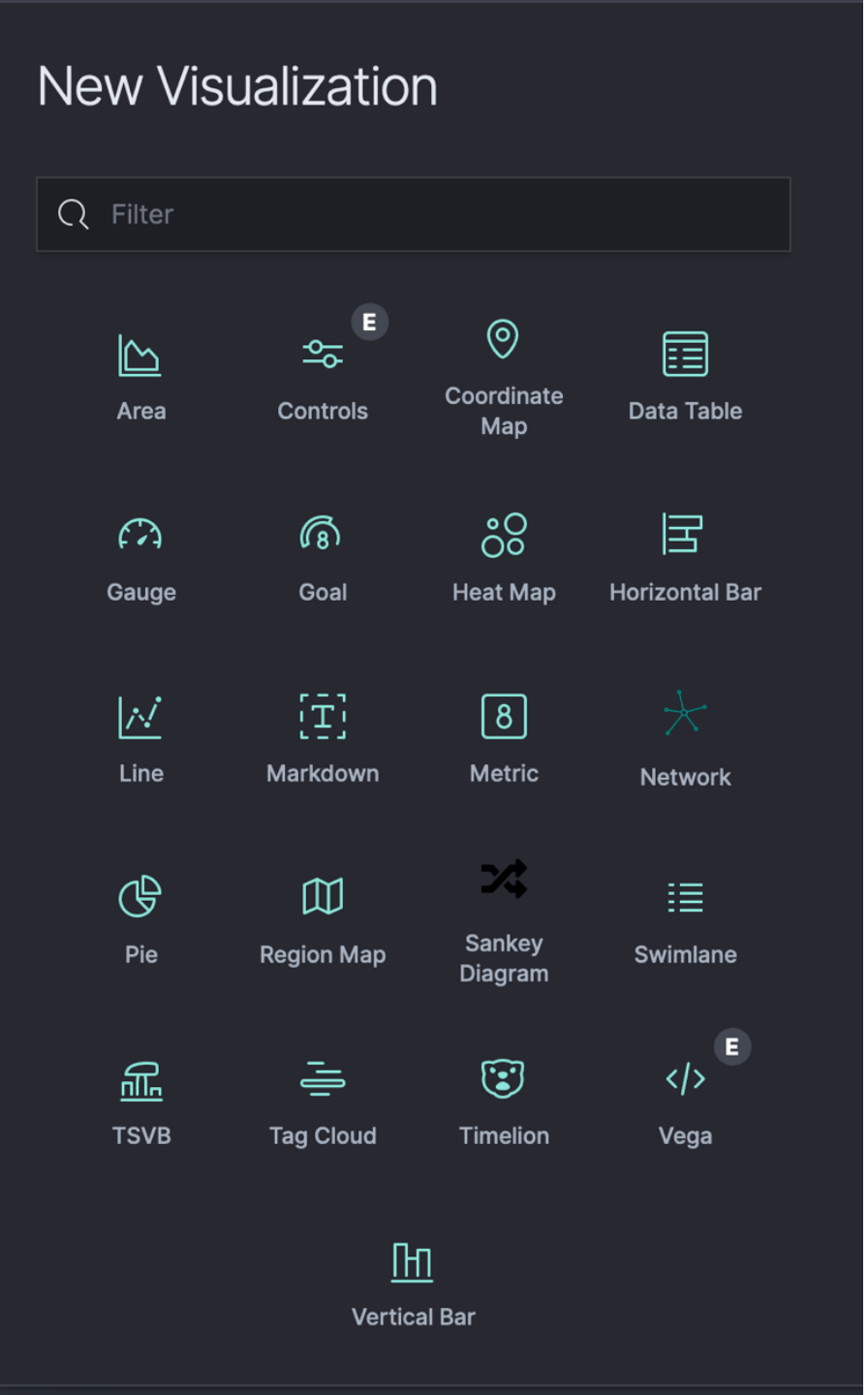
- Highlight application-specific fields of interest
- Grouped by common IT protocols and ICS/IoT protocols
- OT protocols
  - BACnet
  - BSAP
  - DNP3
  - EtherCAT
  - EtherNet/IP
  - Modbus
  - MQTT
  - PROFINET
  - S7comm
  - TDS



Discover

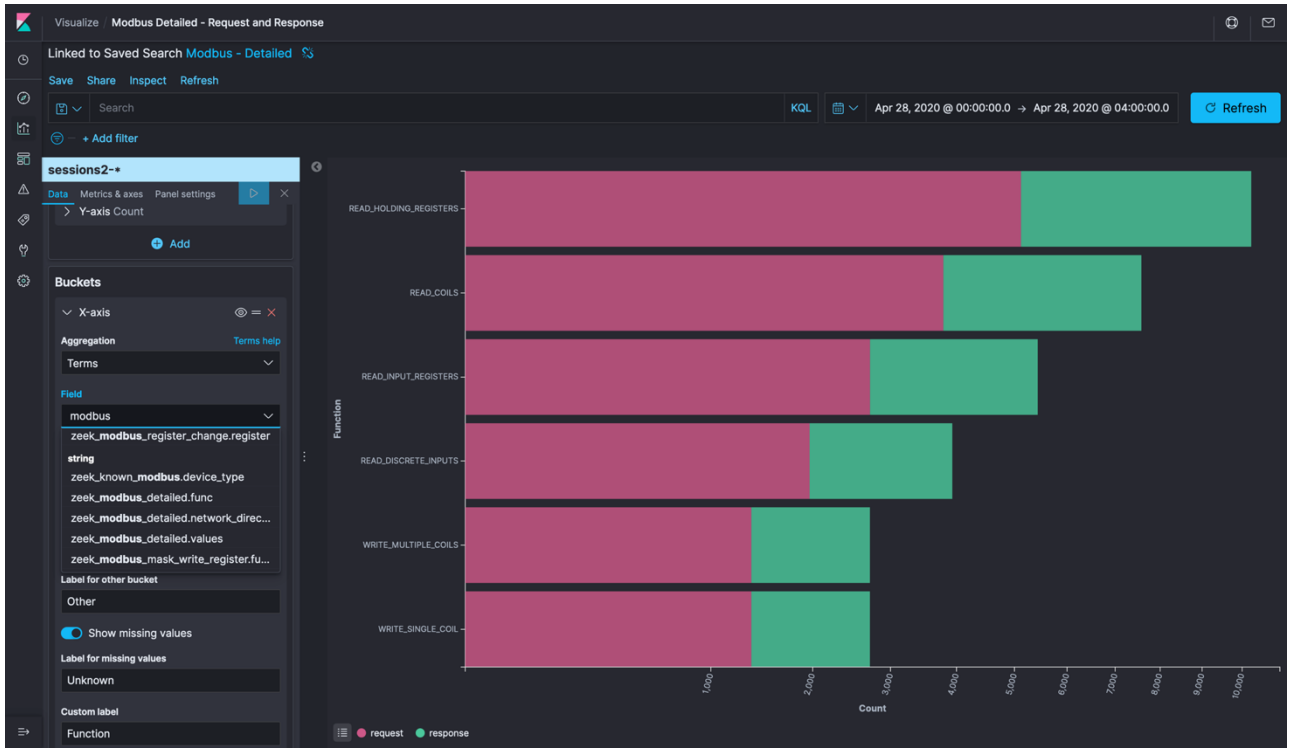
- Field-level details of logs matching filter criteria
- Create and view saved searches and column configurations
- View other events just before and after an event





Custom Visualizations

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



Search Syntax Comparison

	Arkime	Kibana (Lucene)	Kibana (KQL)
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	Kibana (Lucene)	Kibana (KQL)
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	Kibana (Lucene)	Kibana (KQL)
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.*/`	Kibana 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.)

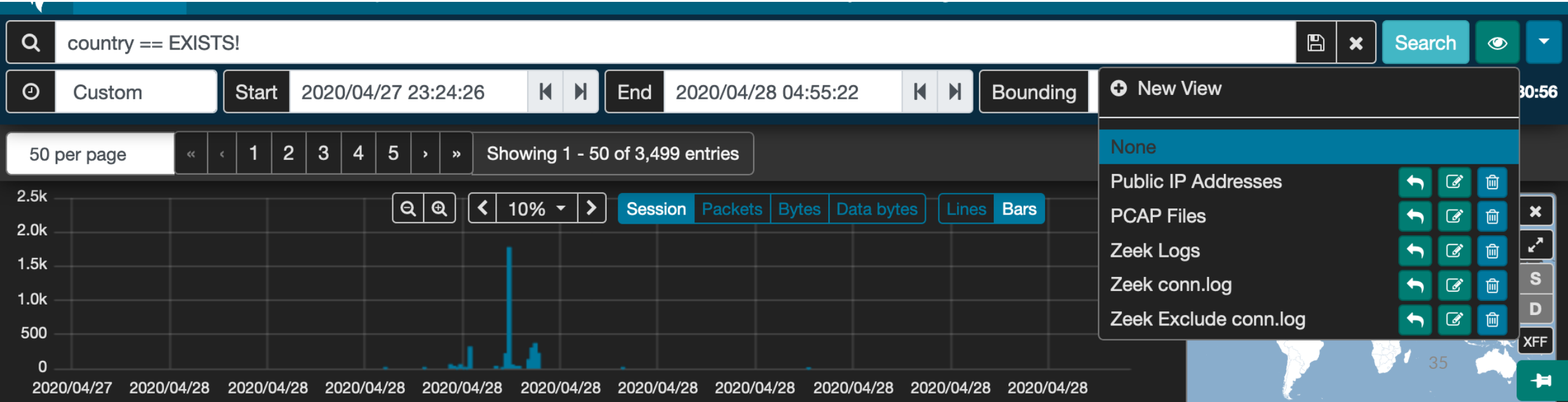
	Arkime	Kibana (Lucene)	Kibana (KQL)
GeoIP information available	`country == EXISTS!`	`_exists_:zeek.destination_geo OR _exists_:zeek.source_geo`	`zeek.destination_geo:* or zeek.source_geo:*`
Zeek log type	`zeek.logType == notice`	`zeek.logType:notice`	`zeek.logType:notice`
IP CIDR Subnets	`ip.src == 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 Kibana time range controls in the upper right-hand corner	Use Kibana 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

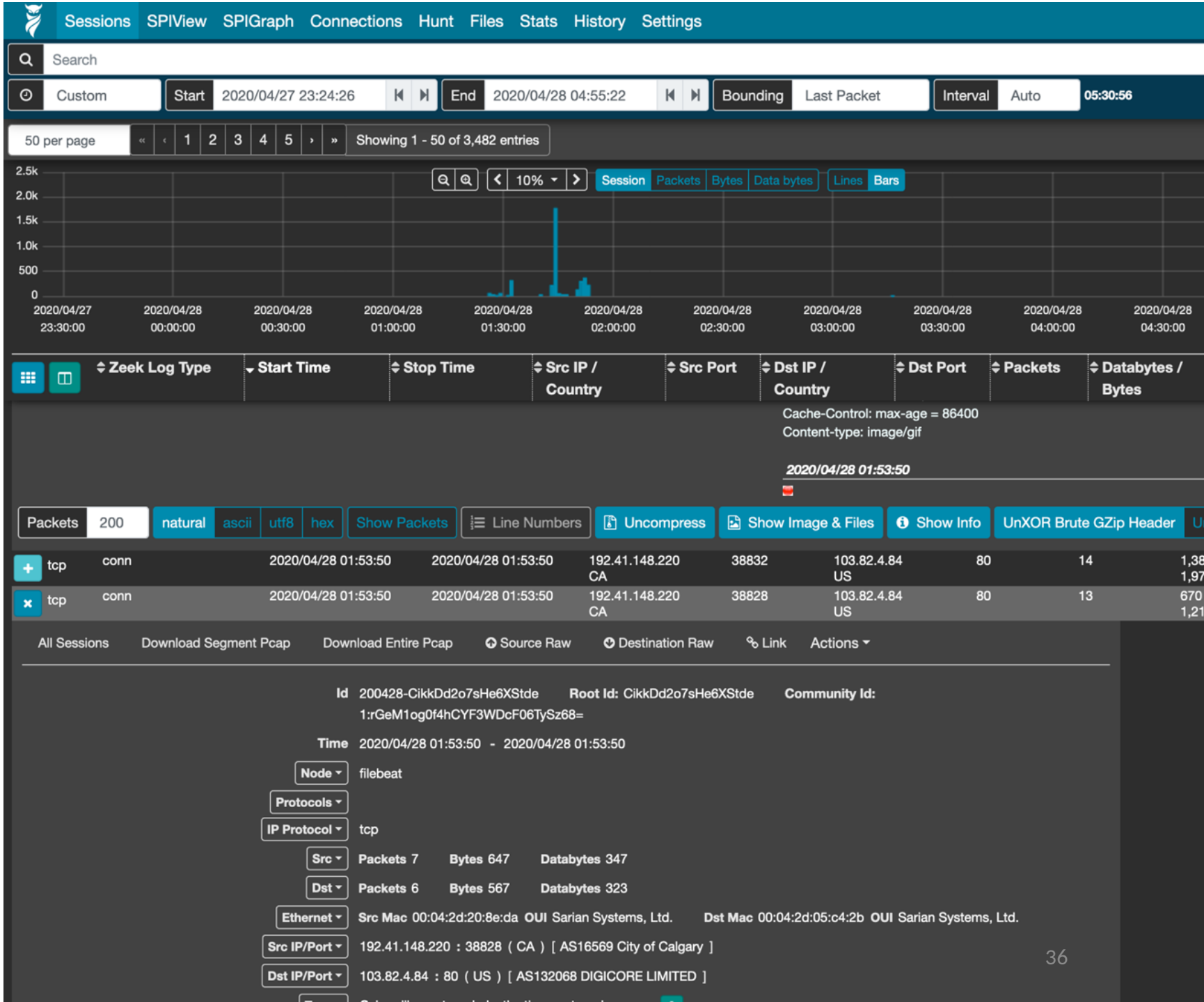
Arkime Filters and Search

- 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



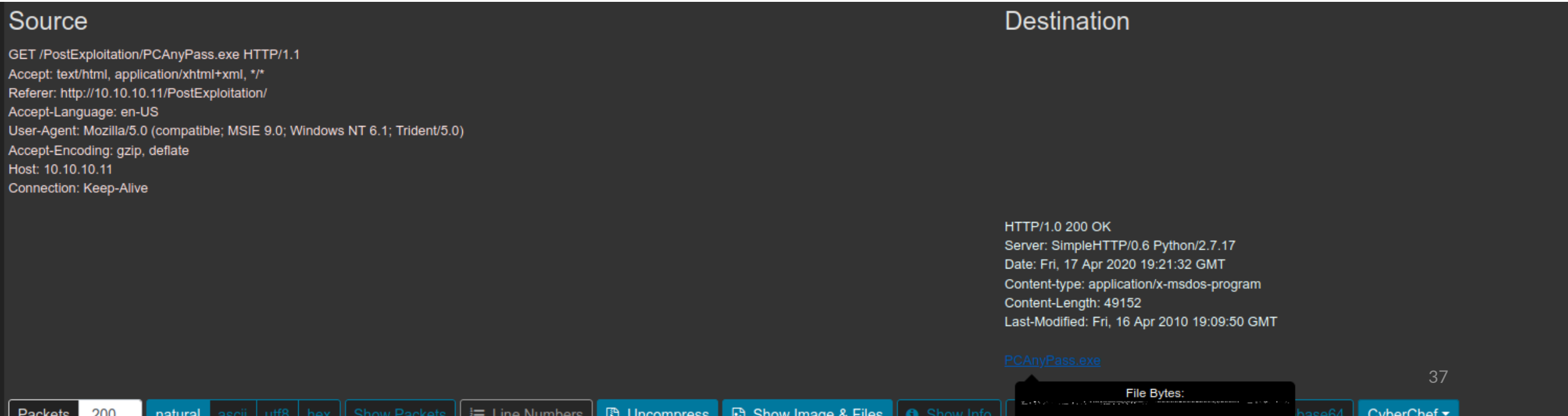
Sessions

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



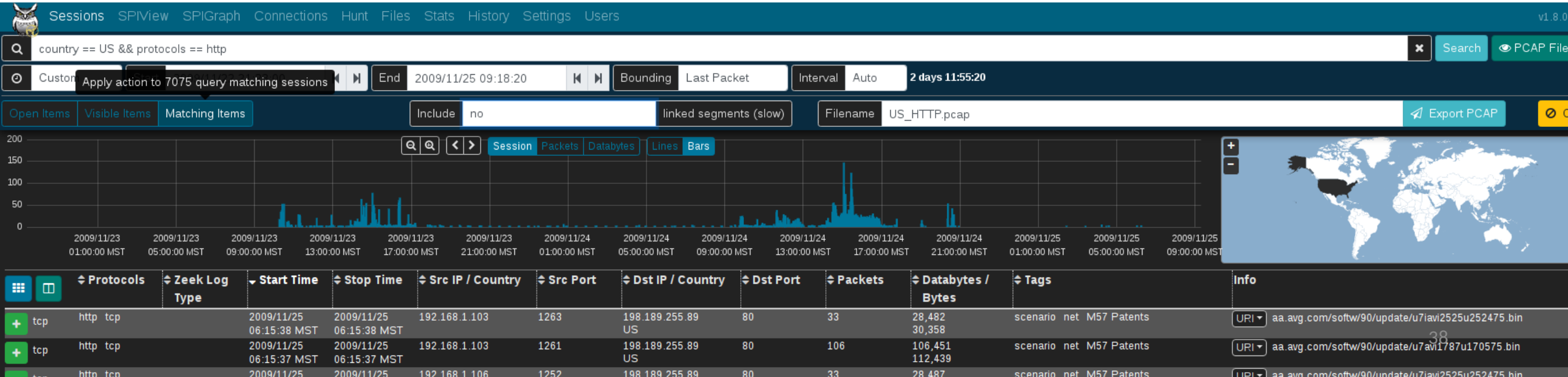
Packet Payloads

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



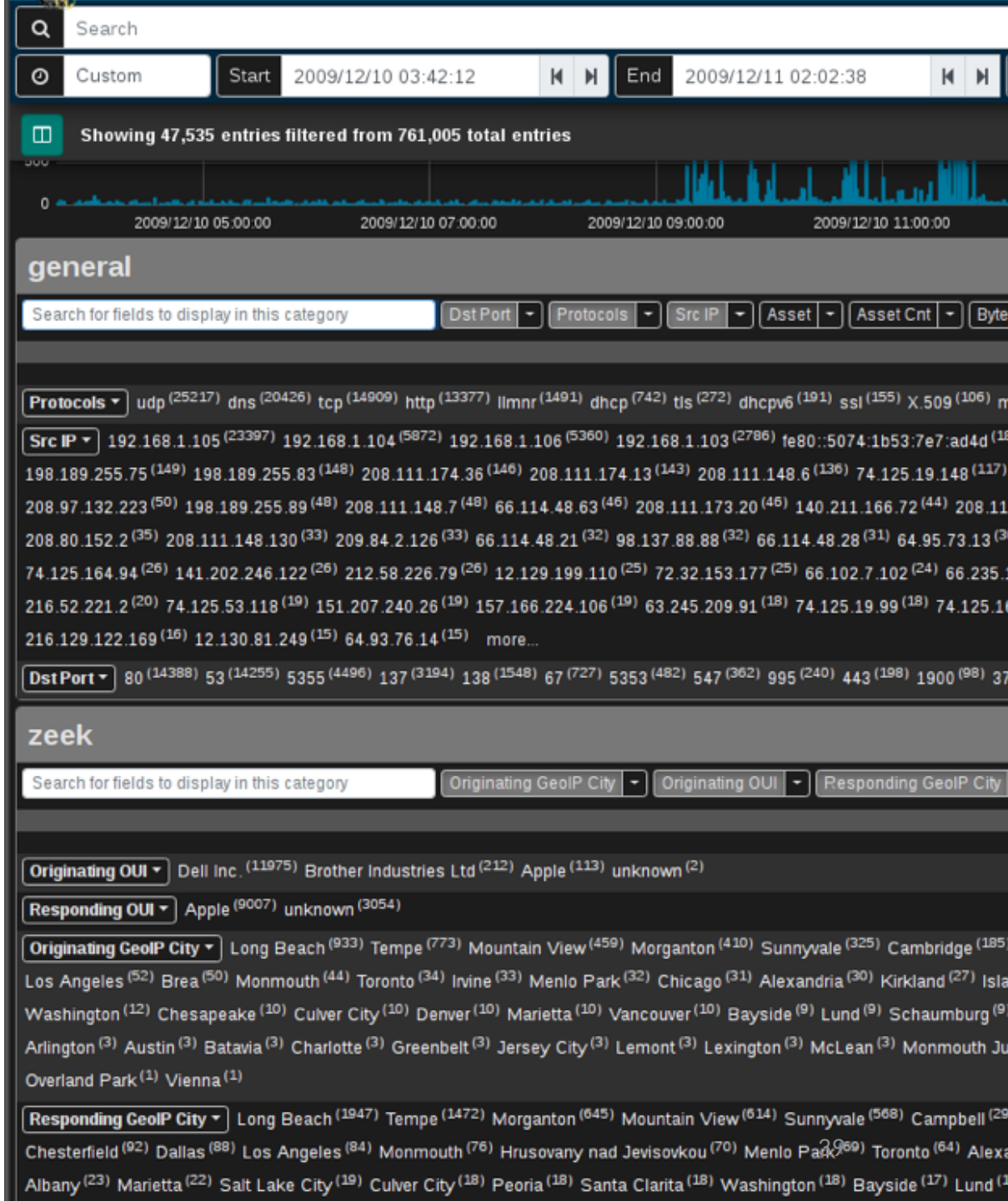
Save and Export PCAP

- 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)



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)

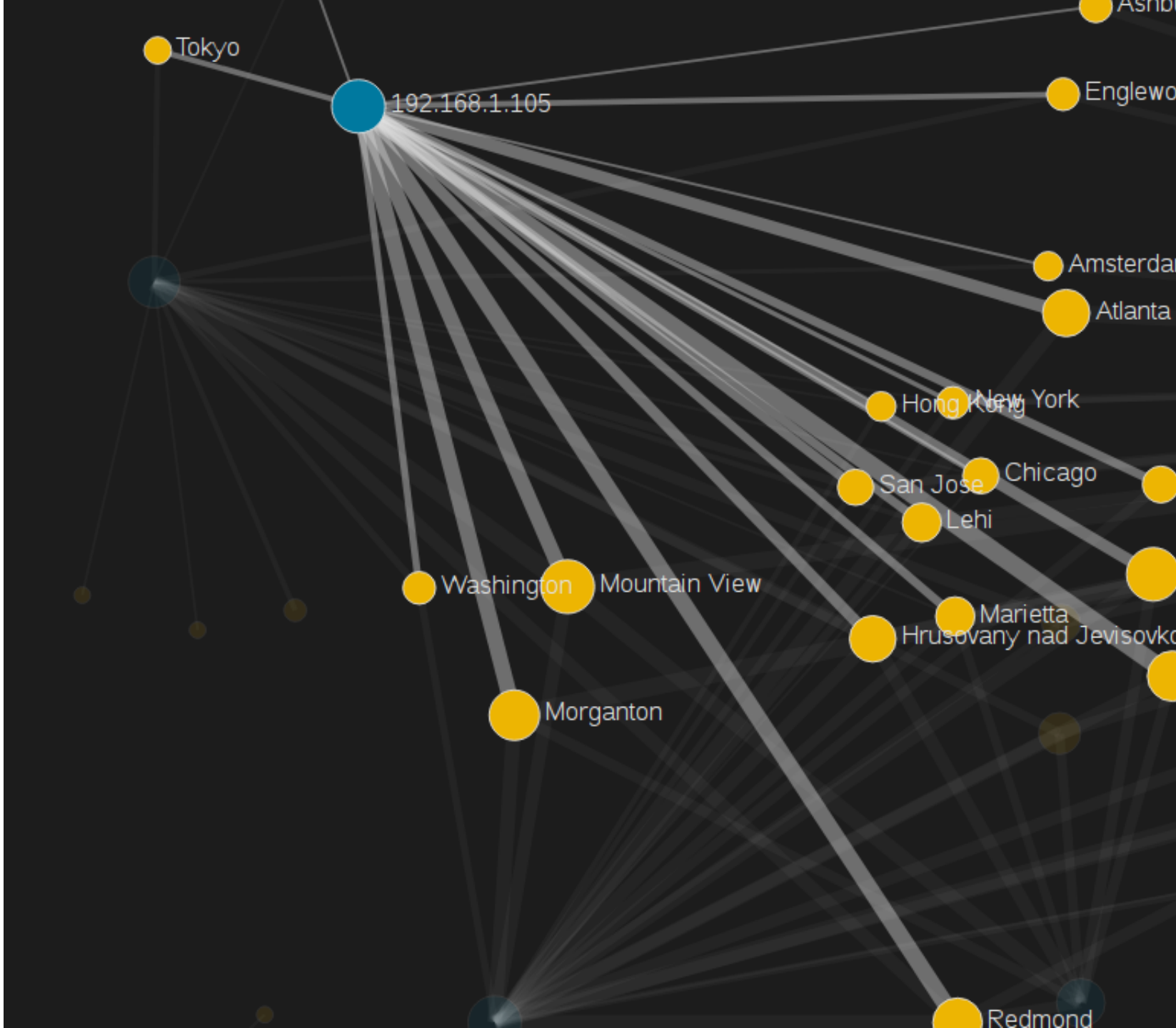


SPIGraph

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

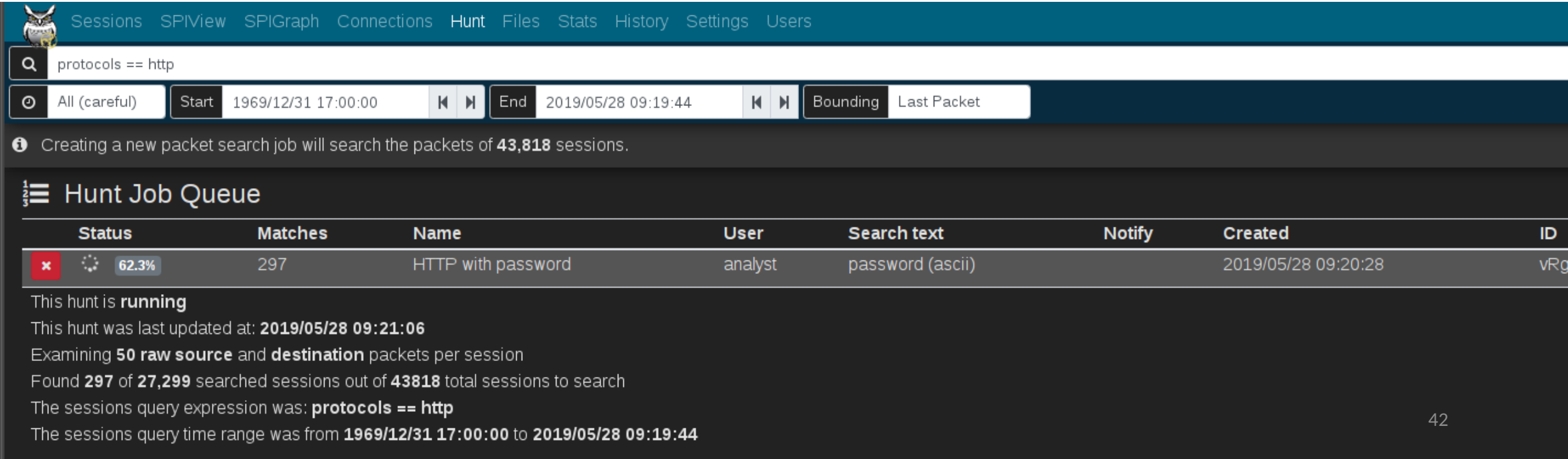
Connections

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



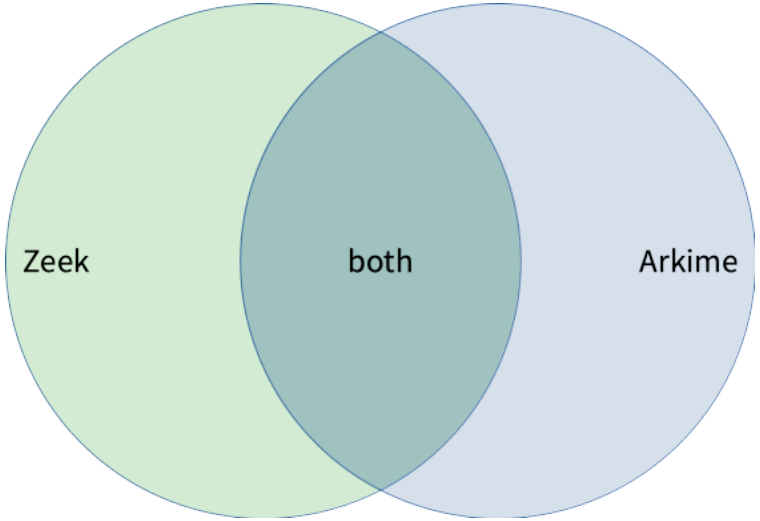
Packet Search (“Hunt”)

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



Data Source Correlation

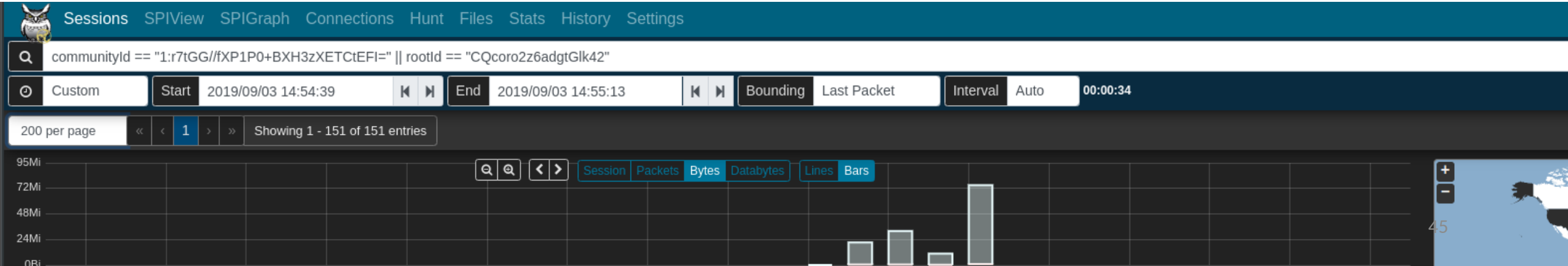
- Search syntax is different between Arkime and Kibana (and in some cases, so are field names)
  - See search syntax comparison table, Malcolm and Arkime docs
- Despite considerable overlap, there are differences in protocol parser support between Zeek and Arkime
  - Learning the strengths of each will 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"
```

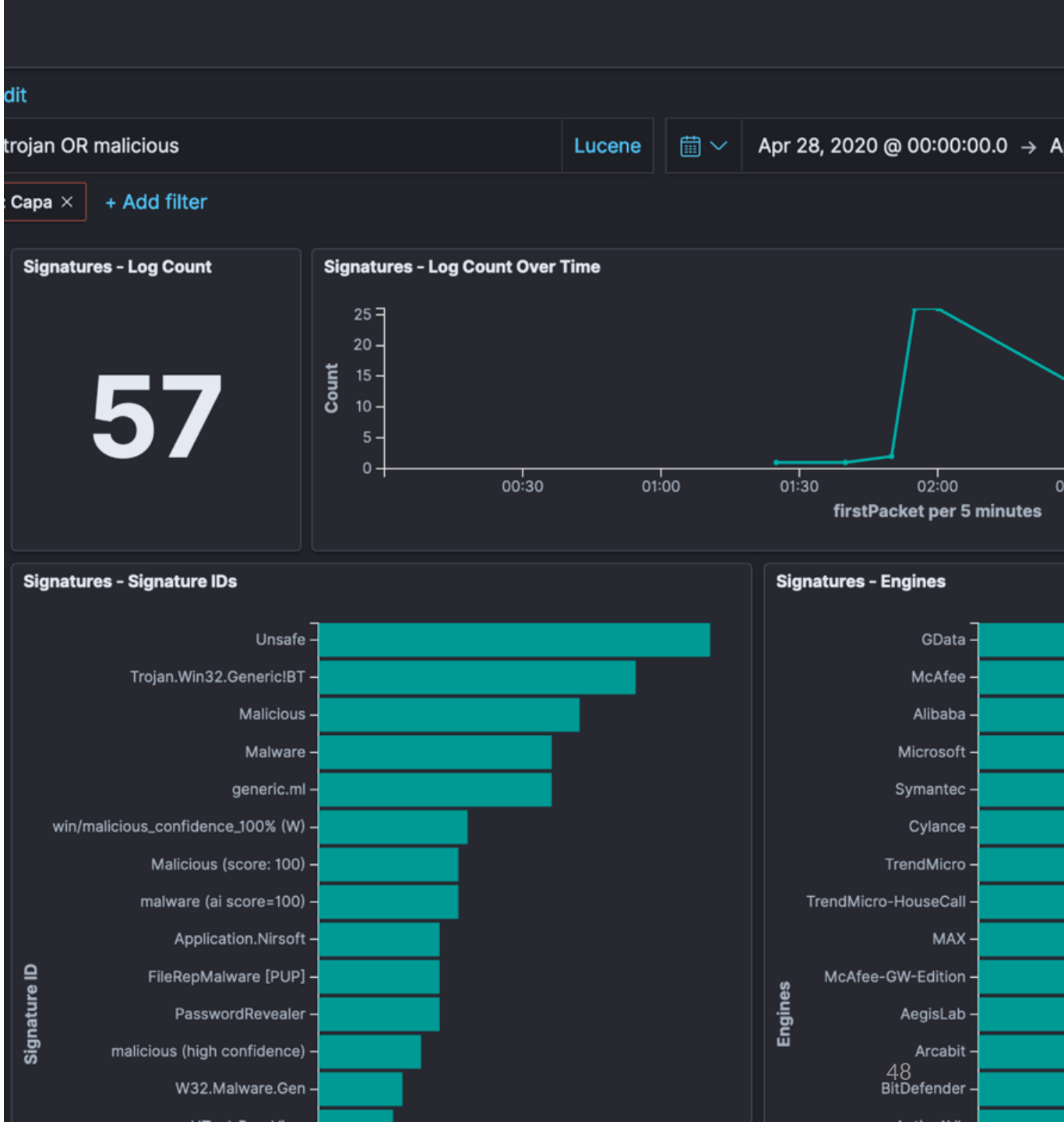


File Analysis

- 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 Kibana 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 Kibana 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!

Network Traffic Analysis with Malcolm

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

Network traffic analysis is all about getting to the “important stuff” as quickly as possible.

There are many open source and proprietary tools available for analyzing raw packet capture (PCAP) files: WireShark, Network Miner, GrassMarlin, etc.

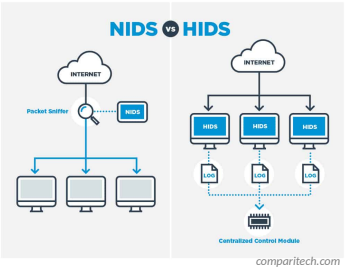
Analyzing PCAP sets that are large (or from complex networks) with many of these tools is difficult, as often they at best struggle and at worst outright fail to handle packet captures larger than a few hundred megabytes.

Today we’re going to talk about the open source network traffic analysis tool suite Malcolm, developed at the Idaho National Lab with support from the US Department of Homeland Security. You may be familiar with some or all of the open source tools which make up Malcolm: all are already available and in general use. What Malcolm provides a is a framework of interconnectivity which makes it greater than the sum of its parts, streamlining network traffic analysis and bringing that “important stuff” to the foreground as painlessly as possible.

This morning we’re going to talk about using Malcolm to gain insight **into both link-layer and application-layer** network traffic.

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

2

Before we jump into our discussion about Malcolm and some of its primary components (like Zeek, the Elastic Stack and Arkime), let's take a minute and talk about intrusion detection systems so we can get an understanding of these tools fit into the threat detection landscape.

When talking about Intrusion Detection Systems, you're usually going to be talking about tools in one of two categories:

- Host Intrusion Detection Systems utilize a native agent that runs locally on individual hosts and network devices. These agents often monitor not only network traffic at the device NIC level, but also track modifications to critical system files, monitor user authentication events, or configuration changes, and report these events to a central manager for alerting and reporting. Host Intrusion Detection Systems are **not** what we are talking about today.

- Network Intrusion Detection Systems are generally passive or out-of-band programs or devices that capture and analyze network traffic at strategic points within the network in order to monitor traffic among devices in the network or between network devices and the outside world.
  - This monitoring and analysis can be done concurrently (in other words, analyzing the traffic as it is captured), or the network traffic can be captured with other tools for later offline analysis. We are taking the latter of these approaches in this presentation.
  - An IDS is generally passive, meaning that it should not alter the network traffic as a side-effect of its analysis. Systems which actively drop suspicious network traffic are called Intrusion Prevention Systems

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)

3

What type of intrusions or attacks might we hope to uncover using intrusion detection system?

A scanning attack is used to assimilate information about a system or network being attacked. By attempting connections to a range of IP addresses within a network and scanning for open ports (responding services) on those hosts, an attacker puts together a map of the topology of the network: types of network traffic allowed through a firewall, active hosts on the network, the operating system, kernel, and software versions running on those hosts, etc. This information can then be used to launch attacks aimed at specific exploits. A good IDS should be able to notice these types of accesses (possibly seen as sequential connections from one host to a range of IP address or ports) and alert that a host scan or port scan took place.

Denial of service attacks work by flooding a network or host with an overwhelming number of connections or requests. This could be something as simple as sending a large number of "ping" packets (ping flood), or by forging the initiation of a TCP connection (SYN flood) causing the host to be unable to respond to legitimate connections. Intrusion detection systems are good at categorizing traffic from or to a particular host or service, and can often track things like connection state for various network protocols, making identifying this type of attack easier to identify.

Finally, a penetration attack is any type of attack which give an unauthorized attacker the ability to access system resources, privileges, or data by exploiting a misconfigured system or a software flaw. These types of attacks are more difficult to identify because once an attacker has a foothold on a system in a network it becomes much easier for them to cover their tracks and mask commands as normal network

- 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)

4

Zeek (formerly Bro) is one of two PCAP analyzing engines used by Malcolm to generate ”metadata” about network traffic, which metadata is indexed and made searchable through Malcolm’s visualization tools. Let’s discuss Zeek’s capabilities to better understand what it offers analysts as a Malcolm data source.

So where does “Zeek” come into the picture? What is Zeek? While it’s sometimes referred to as “Zeek IDS,” and it incorporates some of the techniques from the previous slide, Zeek is more than just an intrusion detection system.

Zeek is an extensible open-source passive network analysis framework, featuring:

- packet capture
- traffic inspection
- intrusion detection
- Flow log recording
- a scripting and data structure framework for log enrichment

If I had to categorize Zeek itself into one of the three detection method categories from the previous slide, I’d categorize it in the “Stateful protocol analysis detection” camp: Zeek’s network traffic parsers examine network traffic at the application layer and reports the behaviors of hosts communication over those protocols. These logs can then be used to do more in-depth manual or automated analysis, as we’ll see throughout our discussion.

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

5

Zeek is fundamentally different from other IDS, in that it goes beyond pure signature matching in favor of analyzing the **application-layer behavior** of hosts themselves (although it does have also have signature matching capabilities similar to Yara or Snort).

Zeek's features can be combined in powerful ways to provide insight into network traffic. With Zeek logs, a network analyst can perform:

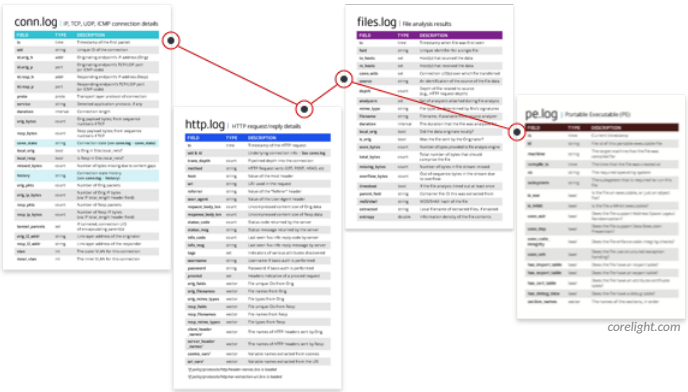
- **content extraction** – for example, extract exfiltrated files from PCAPs for further examination
- **behavioral analysis and session correlation** – as Zeek is highly stateful, extensively tracking application-layer network state, it can be used to determine "what else" took place in the communication between two hosts, or what preceded a suspicious event
- **extensible** – support for uncommon protocols (for example, OT protocols) can be added via scripts and plugin architecture

Zeek is a powerful tool commonly used in network traffic analysis, but it does have its own set of hurdles. In a minute we'll see how Malcolm helps overcome these hurdles.

While Zeek does provide its own packet capture abilities (for example, sniffing for traffic on an interface connected to a network tap on a switch), in the context of this presentation we will be focusing on the use of Zeek (as a component of Malcolm) to perform post-capture network analysis, i.e., against PCAP files gathered previously.

Zeek Log Files

- Network Protocols
- Files
- Detection
- Network Observations



6

As Zeek analyzes network traffic, it generates a number of .log files containing the events it detected.

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

7

- **conn.log** is the "backbone" of a Zeek analysis. Each line of this file represents a unique network session, identified by a **4-tuple** consisting of originating (i.e.., source) IP and port and response (i.e.., destination) IP and port.
  - Each connection in conn.log is assigned a random 18-character unique identifier (**UID**).
  - A particular session's UID from conn.log will be referenced in any other Zeek log files generated for that same PCAP.
  - So, for example, in the case of an HTTP session between a web browser and a web site, there may be one line representing the entire session in conn.log, and many lines in http.log, each representing a different HTTP request belonging to that same session, referencing that connection's UID.

There are many protocol-specific log files Zeek will generate. Taking note of which log files are generated from a network trace can provide insight into a potential breach even before you begin analyzing the files' contents.

For example, if **ssh.log** was generated for a PCAP captured on a network where no SSH servers should be exposed, that could indicate that an attacker has opened an SSH service as a backdoor into the network after having compromised it through some other means.

The **dhcp.log** may provide useful in an assessment when identifying approved network devices, particularly those that communicate wirelessly, by their MAC and associated IP addresses.

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

8

Zeek has a **file analysis engine** that attempts to detect and identify when file transfers occur. Similar to connections, each file is assigned a random **file unique identifier** (FUID) that can be referenced in other log files. For example, a file transferred over HTTP may be referenced by its FUID in http.log, then more information could be found in files.log about that file. Entries in files.log may also be **linked to the sessions** during which they were transferred in conn.log by connection UID.

When possible, Zeek will identify the filename, mime type, file size, and other **attributes of the file** during file analysis.

Two specific types of files are Zeek out into their own log file: **pe.log** contains entries for portable executable files (as these may be of special interest when it comes to network security analyses), and x509.log which contains information about X509-formatted public key certificates.

x509.log, along with ssl.log, and other log files detailing events occurring over encrypted channels can help identify encryption schemes employed.

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

9

The primary log file of interest from this list is **notice.log**.

- A "notice" is Zeek's concept of an alarm: a way to draw extra attention to an event.
- Notices can be generated from any other Zeek script as it is processing traffic.
- Zeek currently implements about 50 notices by default, and Malcolm adds several more, ranging from brute-force SSH login attempts to SQL injection attacks to expired SSL certificates and more.

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

10

weird.log, along with notice.log, is often a good place to begin when looking for **anomalies in network traffic**. This log's contents is varied: Zeek identifies over 150 types of "weird" behavior across many protocols.

Of course, what is weird or invalid in one network may be perfectly normal in another. As such, through setting script variables (covered in a later slide) Zeek can be configured to ignore particular hosts or weird entries if you have determined them to be false positives or normal traffic.

signatures.log is used to flag hits from Zeek's signature-based engine and is also used by Malcolm to log hits from file scanning engines on transferred files extracted by Zeek (more on that later).

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

11

At a configurable **interval** (defaulting to one day), Zeek will dump **summary lists** of various entities it has seen over the course of that period.

These lists may include SSL certificates, MAC addresses seen communicating over the network, hosts that have performed TCP handshakes, **modbus** hosts, and TCP services.

known_hosts.log, along with **conn.log**, is an essential part of using Zeek to build a network diagram.

Zeek may also generate **software.log** when it can detect and identify software communicating across the network, and, when possible, will indicate the version of that software.

- Examples may include identifying Windows operating system versions and clients and servers communicating over HTTP, FTP, SSH, SMTP, and MySQL protocols.
- This may be particularly useful during an assessment to identify network hosts or devices running **software or firmware** with known vulnerabilities, and when identifying servers by operating system, type of application running and version

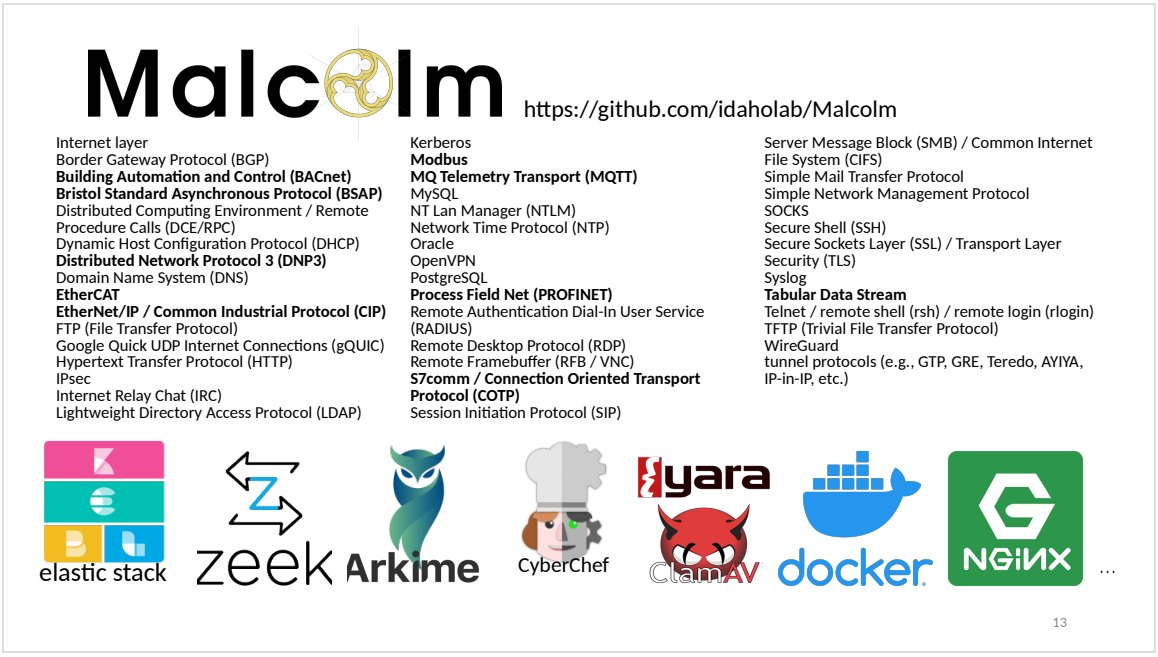
Arkime

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

12

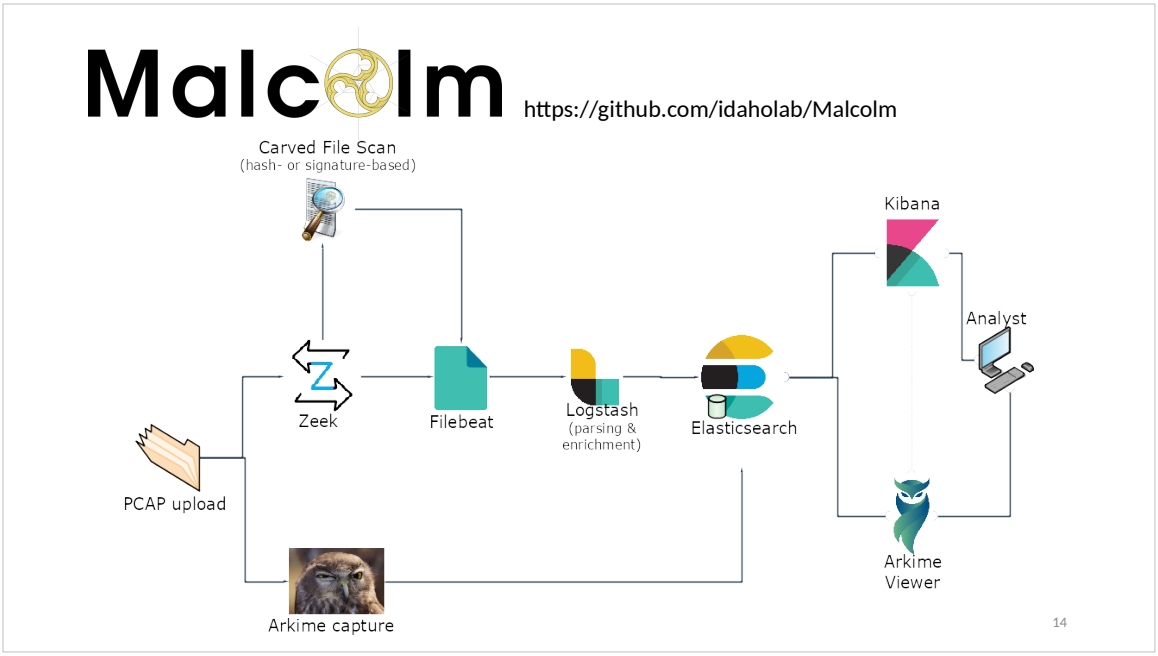
Arkime (formerly Moloch) is the other PCAP analyzer used to populate Malcolm's network session metadata database.

Arkime, similarly to Zeek, parses network traffic data to generate network session metadata. These Arkime "session" logs are written into an Elasticsearch database where they are indexed and become searchable. What is unique and powerful about Arkime is that these network sessions are able to tie back into the original packet payload. This allows for deeper packet inspection and searching that is not just limited to packet headers.



The components comprising Malcolm are industry-standard open source tools, which makes it easy to integrate Malcolm with other solutions in those tools' respective ecosystems.

Malcolm can interpret network traffic across dozens of application protocols, including several protocols commonly seen in OT networks. Much of Malcolm's development is dedicated to improving Malcolm's coverage of protocols used by ICS devices.



An uploaded PCAP file goes through several steps on its way to becoming enriched, indexed and user-searchable.

Upon upload, Malcolm generates metadata for the network traffic represented in a PCAP file using both Zeek and moloch-capture.

Arkime's moloch-capture aggregates metadata for a particular network connection into what it calls a "session" record, which written to Elasticsearch for indexing.

Zeek generates several log files, primarily broken out by application protocol, which contains metadata similar to that generated by moloch-capture.

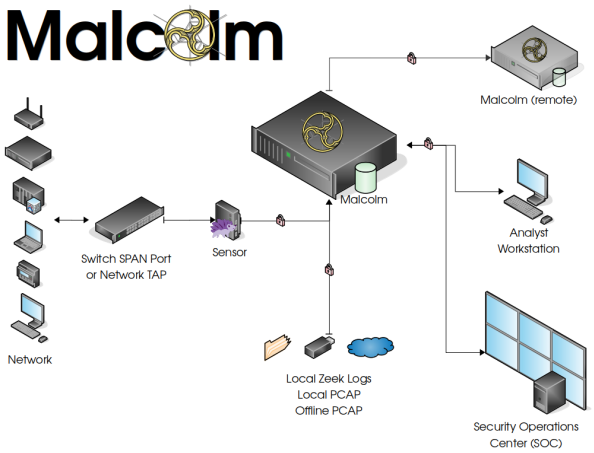
Malcolm can also leverage Zeek's ability to "carve" out files transferred over the network. These files can be scanned (for example, by an antivirus tool) or preserved for analysis with external tools.

The Zeek logs are forwarded by Filebeat to Logstash for further enrichment, normalized to the same field schema as the corresponding Arkime sessions and then indexed into Elasticsearch.

Once ingested by Elasticsearch, Malcolm provides two interfaces for visualizing network traffic: Kibana and Arkime Viewer.

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”



15

Malcolm can be installed and run on any system that supports Docker, a containerization platform for software services.

Most modern commodity hardware (including laptops, desktops and servers) are configured with the resources needed to run Malcolm. Available system memory tends to be the most crucial, with 12GB as a bare minimum and 16GB or more recommended.

This presentation does not cover installation and configuration of Malcolm. Please refer to the documentation and examples found at the links provided.

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

The Host and Network Segment Name Mapping interface allows you to assign names for network segments and hosts based on IP and/or MAC addresses in Zeek logs.

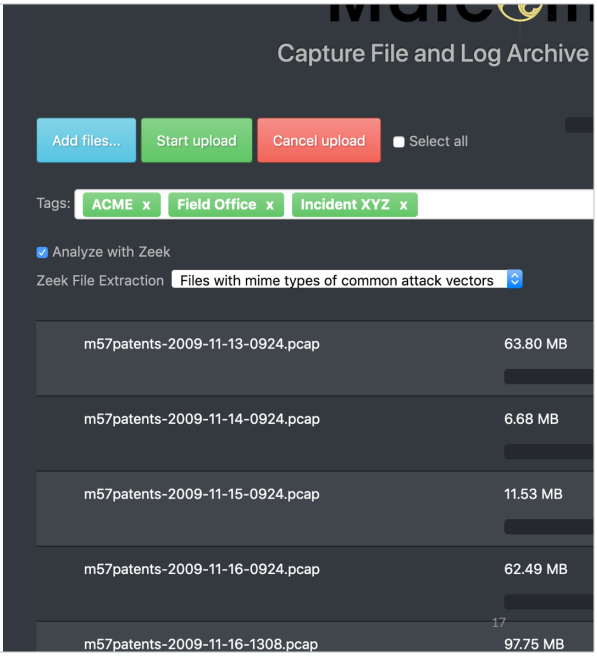
As Zeek logs are processed into Malcolm's Elasticsearch instance, the log's source and destination IP and MAC address fields (zeek.orig_h, zeek.resp_h, zeek.orig_l2_addr, and zeek.resp_l2_addr, respectively) are compared against the list of “host” addresses provided. When a match is found, a new field is added to the log: zeek.orig_hostname or zeek.resp_hostname, depending on whether the matching address belongs to the originating or responding host. For traffic matching the list of “segment” addresses provided, zeek.orig_segment and zeek.resp_segment fields are added. If both zeek.orig_segment and zeek.resp_segment are added to a log, and if they contain different values, the tag cross_segment will be added to the log's tags field for convenient identification of cross-segment traffic.

If the "required tag" field is specified, a log must also contain that value in its tags field in addition to matching the IP or MAC address specified in order for the corresponding name assignment to be made.

These mappings can also be defined in a delimited format in cidr-map.txt and host-map.txt in the Malcolm installation directory.

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



Malcolm must be provided with captured network traffic to interpret, index and present for analysis. While this may be accomplished by dedicated network sensors, more often in assessments PCAP files will have been previously captured and provided for analysis.

PCAP files can be uploaded into Malcolm for processing by accessing /upload on the host on which Malcolm is running.

Prior to starting the upload, “tags” may be added which will allow the data from the PCAP file(s) being uploaded to be searchable using those tags later on. Other behavior relating to how the PCAP file is parsed can also be customized on this page.

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`

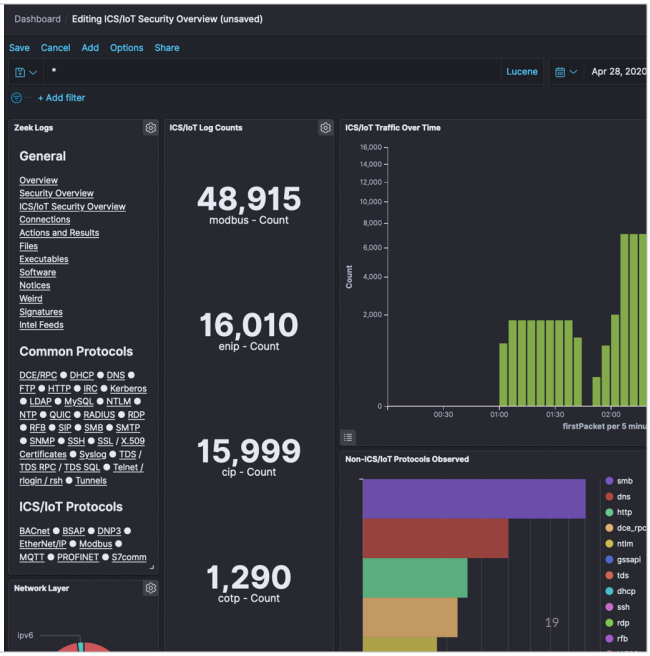
18

Before taking a harder look at the Kibana and Arkime UIs, let's talk a moment about fields Logstash can use to enrich log data before it is written to the database.

- MAC addresses are mapped to hardware manufacturers where possible (to indicate, for example, whether a device was manufactured by Schneider Electric, Schweitzer Engineering or Rockwell Automation)
- GeoIP, ASN and (optionally) reverse DNS lookups are performed for routable IP addresses
- Character frequency analysis is performed for DNS responses and some other hostnames to detect DGA (domain generation algorithm) hostnames often used by malware
- Community-standard fingerprinting algorithms are applied where applicable to allow Malcolm's data to be cross-referenced with other tools

Kibana

- 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/kibana



Kibana is one of Malcolm's two user interfaces for visualizing log data.

Where Kibana really shines is in providing intuitive interactive representations of log data that simplify the process of recognizing and narrowing in on important network events: starting from a high-level overview and being able to quickly "drill-down" to the traffic of an individual host or connection of interest.

Malcolm comes with dozens of prebuilt visualizations specifically for data ingested from Zeek logs. Its dashboards fall into two categories: overview dashboards and protocol-specific dashboards. We'll review some of these in a moment.

Aside from its prebuilt offerings, Kibana provides an easy drag-and-drop WYSIWYG editor for creating new visualizations on the fly.

Kibana Filters and Search

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

The first step to analyzing network traffic with Kibana is to identify the time range of interest. This can be done using the time filter controls in the upper right-hand corner of the interface.

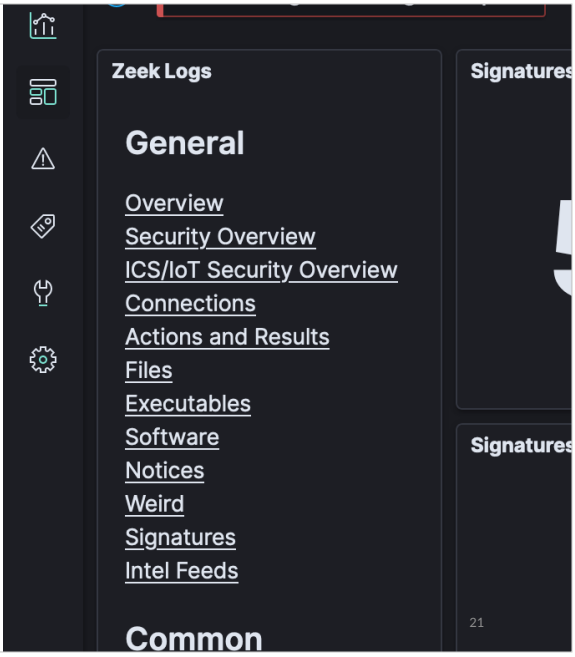
The query bar allows you to specify search constraints, using Lucene query syntax. Modifying the contents of this bar and hitting Enter or clicking the Search icon to the right will run the search and update the results displayed. The “Search Syntax Comparison” table in a few slides gives some examples of the syntax that can be used in the query bar.

The filter bar is another way of specifying search constraints, although it provides more of a GUI-type interface to do so. In most cases there’s not really a meaningful distinction between putting query terms in via the query bar vs. the filter bar, although using the filter bar does allow you to more easily pin filters across different dashboards and is somewhat more intuitive. Filters may also be populated by clicking on values in charts and graphs and choosing the magnifying glass icon with either the plus sign (+) or minus sign (-) to restrict to or exclude that value from the result set.

A future release of Kibana will merge them into a single search component.

Overview Dashboards

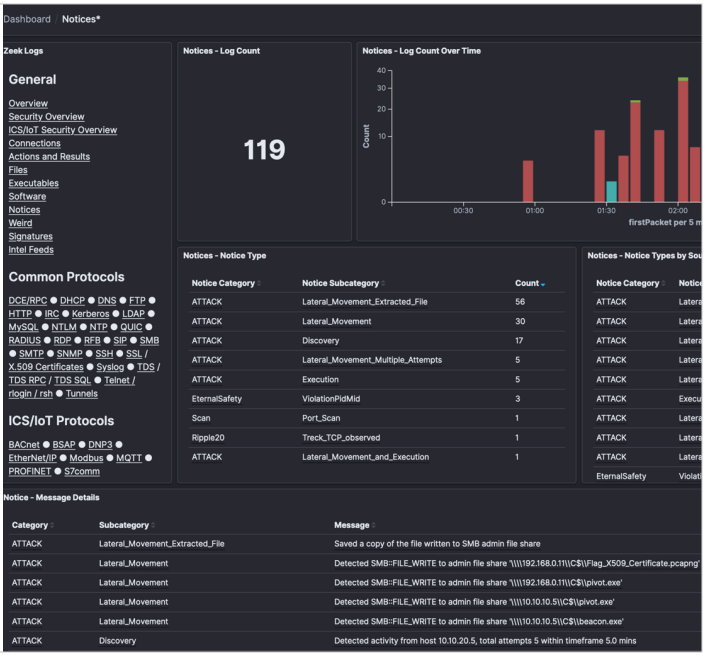
- High-level view of trends, sessions and events
- Populated from logs across all protocols
- Good jumping-off place for investigation



The dashboards under the General section of Malcolm's Kibana navigation panel provide a high-level overview of network traffic from across all of the logs generated by Zeek. These dashboards are a good jumping-off point for investigation when trying to get a "feel" for the network and the application protocols used by the hosts that comprise it.

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



As discussed earlier, Zeek notices are the tool's way of raising some event to the forefront of an analyst's attention. To quote the Zeek notice framework's documentation:

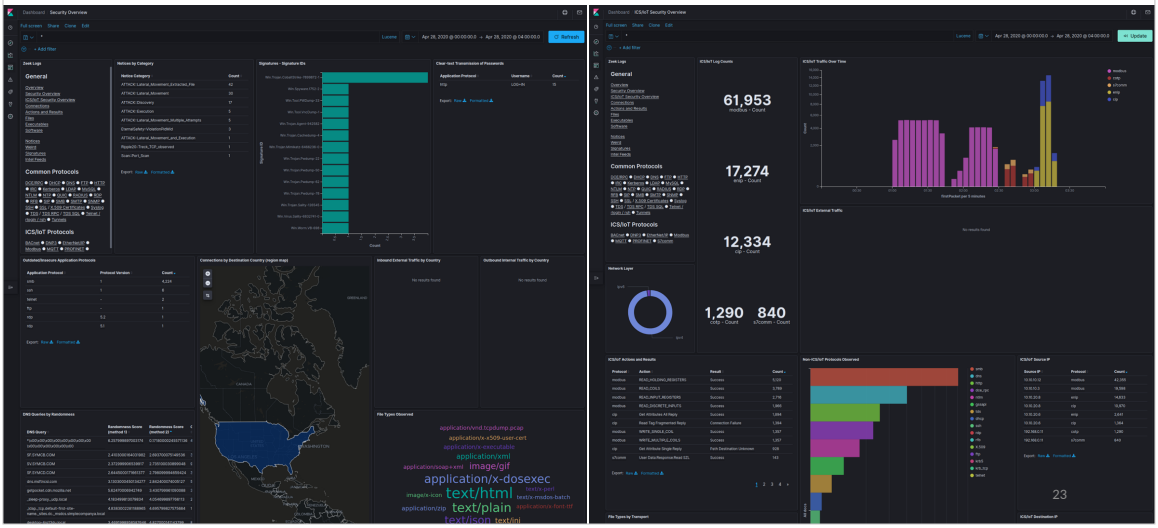
Zeek ships with a large number of policy scripts which perform a wide variety of analyses. Most of these scripts monitor for activity which might be of interest for the user. However, none of these scripts determines the importance of what it finds itself. Instead, the scripts only flag situations as potentially interesting, leaving it to the local configuration to define which of them are in fact actionable. This decoupling of detection and reporting allows Zeek to address the different needs that sites have. Definitions of what constitutes an attack or even a compromise differ quite a bit between environments, and activity deemed malicious at one site might be fully acceptable at another.
(https://docs.zeek.org/en/current/frameworks/notice.html)

Zeek currently implements about 50 notices by default, and Malcolm adds several more, ranging from brute-force SSH login attempts to SQL injection attacks to expired SSL certificates and more. A list of Zeek's built-in notices can be found at https://docs.zeek.org/en/stable/zeek-noticeindex.html.

The third-party Zeek plugins used by Malcolm are listed in the Malcolm README at https://github.com/idaholab/Malcolm#Components. They include, but are not limited to, notices generated for:

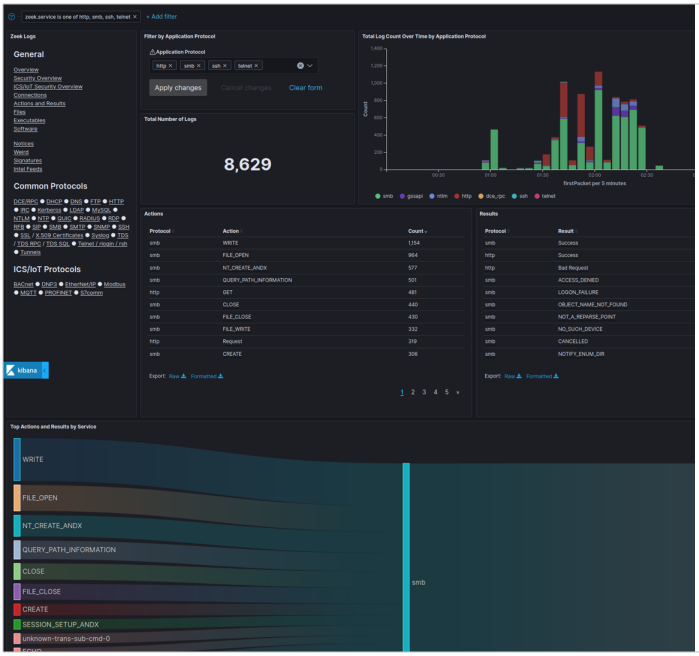
- cleartext passwords detected in in HTTP POST requests
- noncompliant HTTP requests (like those used for smuggling)
- XOR-obfuscated file transfers
- Behavior/techniques categorized according to the MITRE ATT&CK framework
- Various other CVEs and vulnerabilities, including
    - Bad Neighbor (CVE-2020-16898)
    - CallStranger (CVE-2020-12695)

Security & ICS/IoT Security Overview



The Security Overview and ICS/IoT Security Overview dashboards highlight events that may be of particular interest from a security standpoint, including Zeek notices, signatures triggered from file scans, clear-text transmission of passwords, outdated or insecure versions of application protocols, traffic originating from or directed to public IP addresses, file transfers and more.

These dashboards are a good place to start when looking for indicators of compromise or vulnerabilities in network traffic.



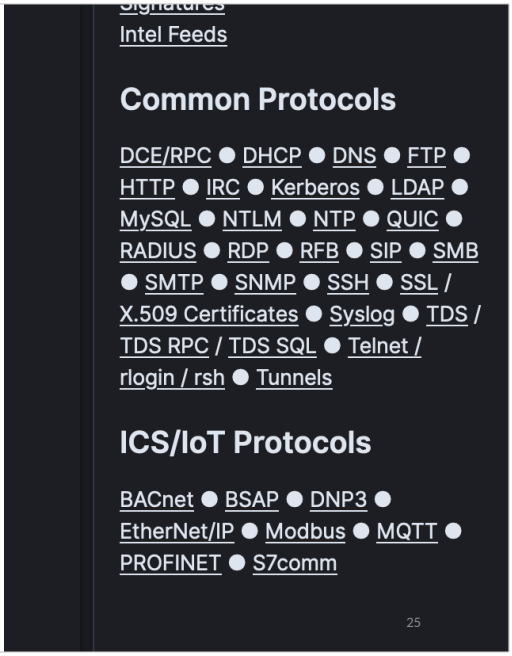
Where possible, Malcolm correlates common fields from across different protocols to allow you to view one device's or application's network traffic in the context of the other traffic occurring around it.

For example, multiple failed HTTP authentication attempts, followed by a successful authenticated HTTP POST operation followed by successful reads and writes to a file server could indicate that a foothold was obtained in an HTTP server that allowed an adversary to pivot to another service in the network.

A good example of this is the Actions and Results dashboard, in which actions (such as "a file was written," "a logon was attempted," "a web page was requested") and "results" ("success," "access denied," "page not found") can be inspected together regardless of protocol.

Protocol Dashboards

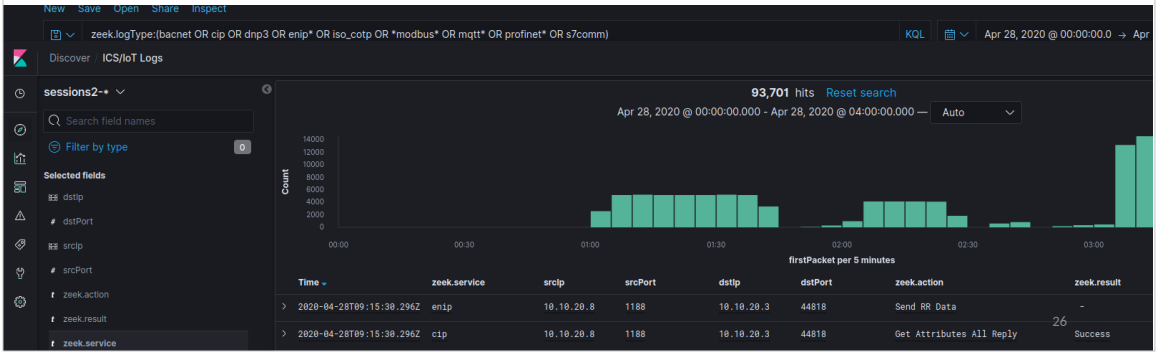
- Highlight application-specific fields of interest
- Grouped by common IT protocols and ICS/IoT protocols
- OT protocols
  - BACnet
  - BSAP
  - DNP3
  - EtherCAT
  - EtherNet/IP
  - Modbus
  - MQTT
  - PROFINET
  - S7comm
  - TDS



In addition to the overview dashboards, Malcolm provides dozens of dashboards tailored to specific application protocols, including protocols commonly used in industrial control systems networks.

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

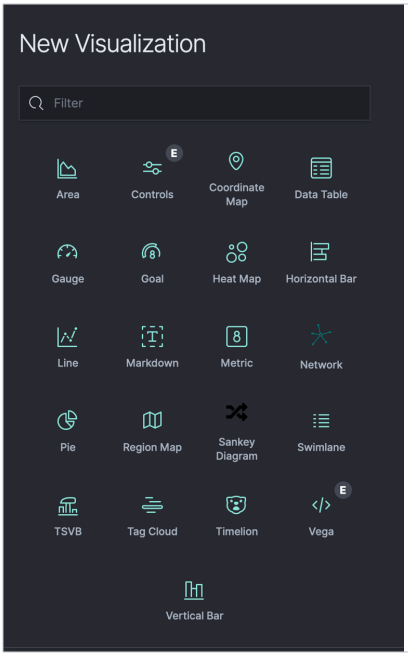


The Discover view enables you to view events on a record-by-record basis, similar to a session record in Arkime, which we'll discuss in a moment, or to an individual line from a Zeek log.

The data table in the Discover view can be customized to display only the fields relevant to the traffic you're interested in: for example, a "play-by-play" of an HTTP session could be reviewed by filtering on zeek.logType:http, sorting by Time and including the following fields in the table:

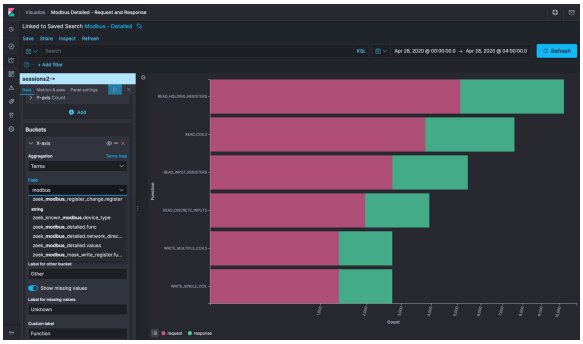
- srcIp
- zeek_http.user_agent
- zeek_http.referrer
- dstIp
- zeek_http.host
- zeek_http.uri
- zeek_http.status_msg

This configuration could be stored as a saved search and returned to for future investigation.



Custom Visualizations

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



27

The visualizations page allows you to view and manage visualization components, which are like “graphical building blocks” to be used in dashboards. Kibana includes many different kinds of charts, tables, and maps for displaying your data.

Search Syntax Comparison

	Arkime	Kibana (Lucene)	Kibana (KQL)
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`

30

As Malcolm is a collection of open source tools, an analyst must be aware of and become familiar with the query languages understood by its varied components.

This table compares common types of query strings across Arkime, Lucene and Kibana Query Language syntaxes.

Search Syntax Comparison (cont.)

	Arkime	Kibana (Lucene)	Kibana (KQL)
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	Kibana (Lucene)	Kibana (KQL)
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.*/`	Kibana 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	Kibana (Lucene)	Kibana (KQL)
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 Kibana time range controls in the upper right-hand corner	Use Kibana 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

34

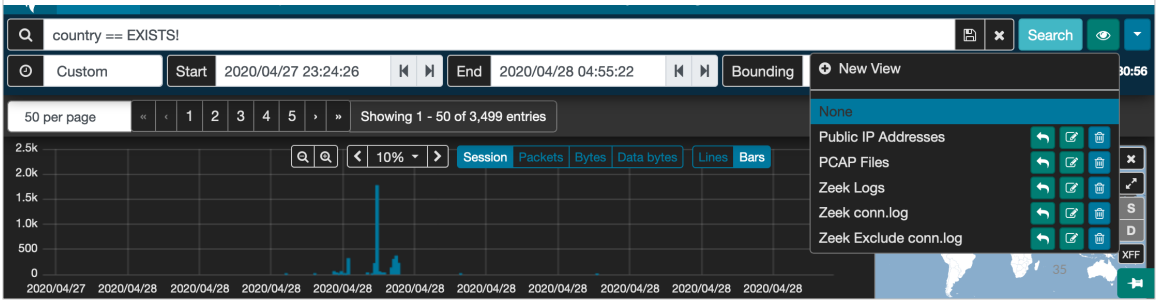
While Kibana is great for "at-a-glance" views and for creating custom visualizations, Arkime (formerly Moloch) provides another interface for examining network traffic that may be better suited to in-depth analysis and network forensics.

Earlier when we talked about the Malcolm PCAP processing pipeline, we mentioned two metadata representations of the same network traffic: the logs generated by Zeek and the session records generated by Arkime's moloch-capture. While Malcolm's Kibana dashboards are focused on the Zeek logs, its instance of Arkime can be used to view **both** Zeek logs and Arkime sessions together in the same interface.

Another strength of Arkime is its ability to tie the session metadata back to the original packets' payloads, allowing you to view, search and export the data deeper in the PCAP that may not be referenced in the metadata. Arkime is able to efficiently deal with very large PCAP file sets, something that Wireshark struggles to do.

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

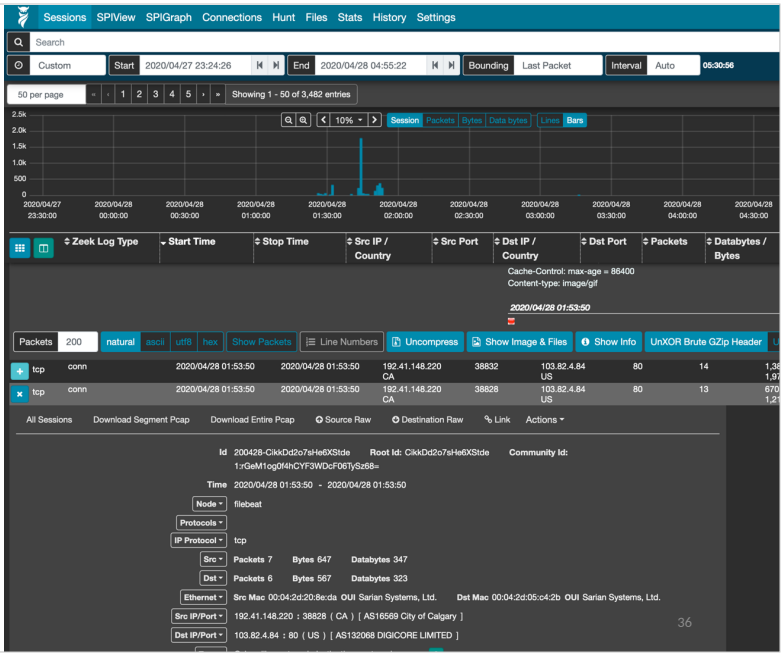


The Arkime interface has various controls for applying time, geographic and field value filters to narrow the set of matching sessions.

In addition, Arkime's views (indicated by the eyeball icon) allow overlaying additional previously-specified filters onto the current sessions filters. For convenience, Malcolm provides several Arkime preconfigured views including several on the zeek.logType field.

Sessions

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

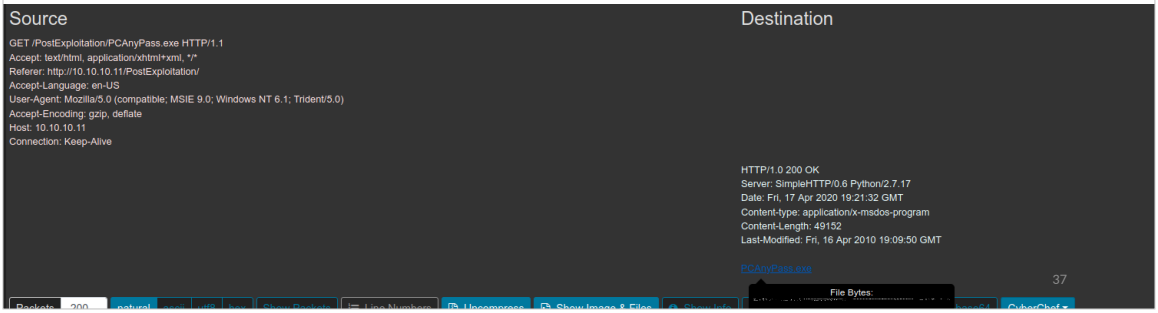


Arkime's Sessions tab provides low-level details of the sessions being investigated (similar to Kibana's Discover interface), whether they be Arkime sessions created from PCAP files or Zeek logs mapped to the Arkime session database schema.

The set of fields present in the sessions table can be also customized, saved and later recalled.

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

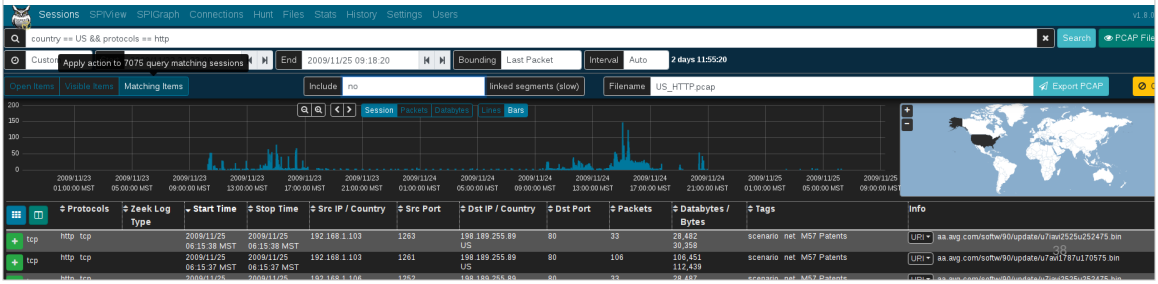


As mentioned, Arkime's ability to tie a session record back to its original packet(s) is one of its greatest strengths. Details for individual sessions/logs can be expanded by clicking the plus icon on the left of each row. For Arkime session records, an additional *Packets* section will be visible underneath the metadata sections.

When the details of a session of this type are expanded, Arkime will read the packet(s) comprising the session for display here. Various controls can be used to adjust how the packet is displayed (enabling natural decoding and enabling *Show Images & Files* may produce visually pleasing results), and other options (including PCAP download, carving images and files, applying decoding filters, and examining payloads in CyberChef) are available.

Save and Export PCAP

- 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)



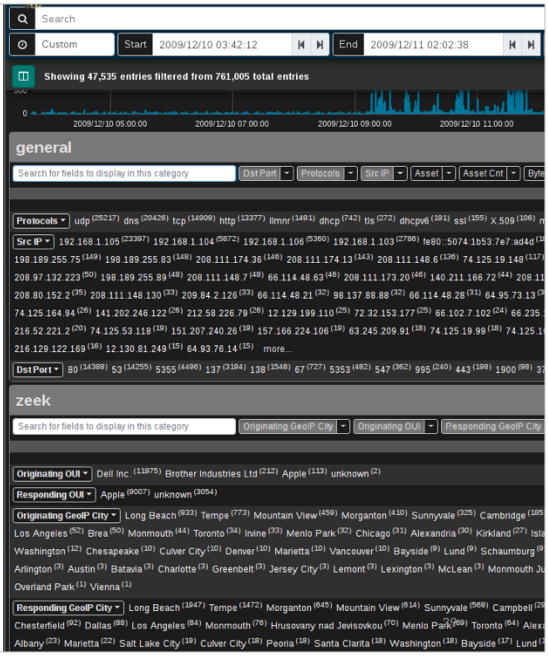
Clicking the down arrow ▼ icon to the far right of the search bar presents a list of actions including *PCAP Export*.

When full PCAP sessions are displayed, the *PCAP Export* feature allows you to create a new PCAP file from the matching Arkime sessions, including controls for which sessions are included (open items, visible items, or all matching items) and whether or not to include linked segments. Click *Export PCAP* button to generate the PCAP, after which you'll be presented with a browser download dialog to save or open the file.

Note that depending on the scope of the filters specified this might take a long time (or, possibly even time out).

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)



Arkime's SPI (Session Profile Information) View provides a quick and easy-to-use interface for exploring session/log metrics. The SPIView page lists categories for general session metrics (e.g., protocol, source and destination IP addresses, sort and destination ports, etc.) as well as for all of various types of network understood by Arkime and Zeek. These categories can be expanded and the top *n* values displayed, along with each value's cardinality, for the fields of interest they contain.

Click the the plus icon to the right of a category to expand it. The values for specific fields are displayed by clicking the field description in the field list underneath the category name. The list of field names can be filtered by typing part of the field name in the *Search for fields* dialog to display in this category text input. The *Load All* and *Unload All* buttons can be used to toggle display of all of the fields belonging to that category. Once displayed, a field's name or one of its values may be clicked to provide further actions for filtering or displaying that field or its values. Of particular interest may be the *Open [fieldname] SPI Graph* option when clicking on a field's name. This will open a new tab with the SPI Graph populated with the field's top values.

Note that because the SPIView page can potentially run many queries, SPIView limits the search domain to seven days (in other words, seven indices, as each index represents one day's worth of data). When using SPIView, you will have best results if you limit your search time frame to less than or equal to seven days.

SPIGraph

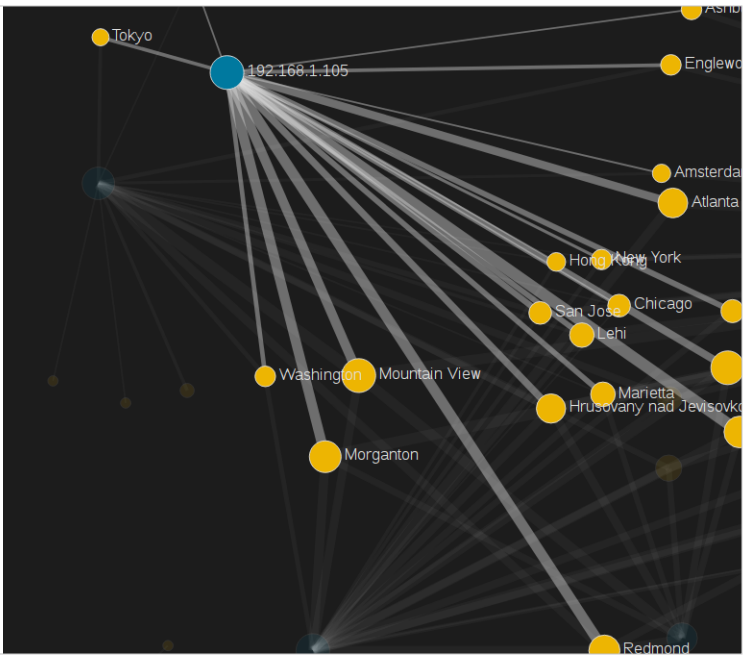
- View “top *n*” field values chronologically and geographically
- Identify trends and patterns in network traffic

Arkime's SPI (Session Profile Information) Graph visualizes the occurrence of some field's top n values over time, and (optionally) geographically. This is particularly useful for identifying trends in a particular type of communication over time: traffic using a particular protocol when seen sparsely at regular intervals on that protocol's date histogram in the SPIGraph may indicate a connection check, polling, or beaconing (for example, see the llmnr protocol in the screenshot).

Controls can be found underneath the time bounding controls for selecting the field of interest, the number of elements to be displayed, the sort order, and a periodic refresh of the data.

Connections

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



The Connections page presents network communications via a force-directed graph, making it easy to visualize logical relationships between network hosts.

Controls are available for specifying the query size (where smaller values will execute more quickly but may only contain an incomplete representation of the top n sessions, and larger values may take longer to execute but will be more complete), which fields to use as the source and destination for node values, a minimum connections threshold, and the method for determining the "weight" of the link between two nodes. As is the case with most other visualizations in Arkime, the graph is interactive: clicking on a node or the link between two nodes can be used to modify query filters, and the nodes themselves may be repositioned by dragging and dropping them. A node's color indicates whether it communicated as a source/originator, a destination/responder, or both.

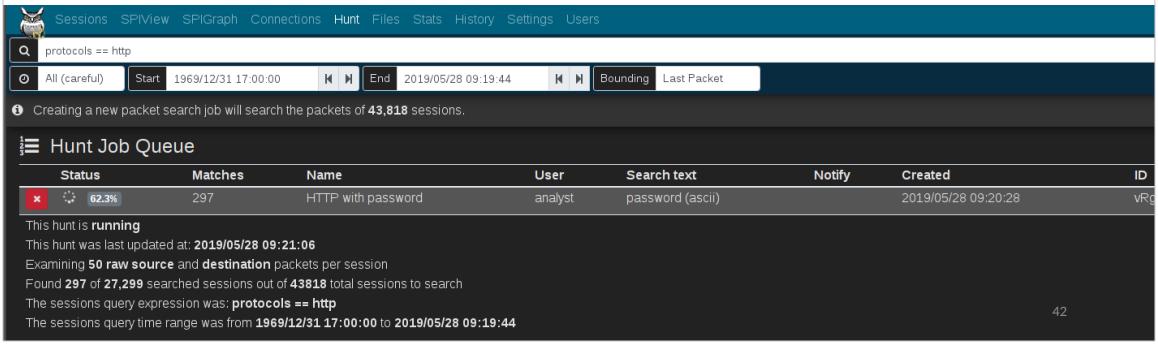
While the default source and destination fields are Src IP and Dst IP:Dst Port, the Connections view is able to use any combination of any of the fields populated by Arkime and Zeek. For example:

- Src OUI and Dst OUI (hardware manufacturers)
- Src IP and Protocols
- Originating Network Segment and Responding Network Segment
- Originating GeoIP City and Responding GeoIP City
- or any other combination of these or other fields.

A recent addition to this feature (and one developed specifically for Malcolm and then contributed back upstream to the Arkime project) is the ability to specify a “baseline” time frame in the Connections view to visualize changes to a network over time (e.g., new hosts or protocols appearing in your network). This feature is mainly useful if you have prior long-term packet captures available in orde to establish baseline against which current traffic may be compared.

Packet Search (“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



Arkime's Hunt feature allows an analyst to search within the packets themselves (including payload data) rather than simply searching the session metadata. The search string may be specified using ASCII (with or without case sensitivity), hex codes, or regular expressions. Once a hunt job is complete, matching sessions can be viewed in the Sessions view.

Clicking *Create a packet search job* on the Hunt page will allow you to specify the following parameters for a new hunt job:

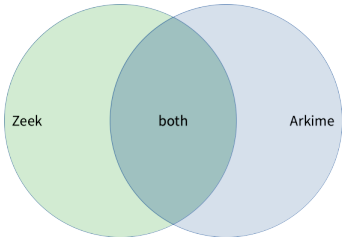
- a packet search job name
- a maximum number of packets to examine per session
- the search string and its format (ascii, ascii (case sensitive), hex, regex, or hex regex)
- whether to search source packets, destination packets, or both
- whether to search raw or reassembled packets

Click the *Create* button to begin the search. Arkime will scan the source PCAP files from which the sessions were created according to the search criteria. Note that whatever filters were specified when the hunt job is executed will apply to the hunt job as well; the number of sessions matching the current filters will be displayed above the hunt job parameters with text like "ⓘ Creating a new packet search job will search the packets of # sessions.”

Once a hunt job is submitted, it will be assigned a unique hunt ID (a long unique string of characters like yuBHAGsBdljYmwGkbEMm) and its progress will be updated periodically in the Hunt Job Queue. More details for the hunt job can be viewed by expanding its row with the plus icon on the left

Data Source Correlation

- Search syntax is different between Arkime and Kibana (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



44

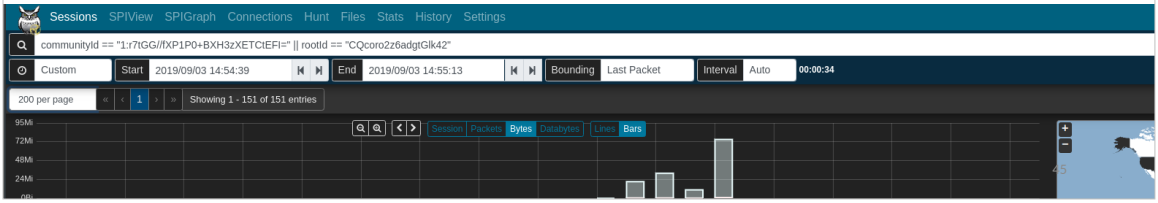
Although Malcolm's Kibana and Arkime interfaces provide two different views into the same network data, there are a few notable differences between the two:

- The Arkime and Kibana search syntax is different.
- Arkime uses its own field names in its user interface: for example, searching protocols == http in Arkime is equivalent to searching protocol:http in Kibana. Enabling *Display Database Fields* in the *Fields* section of Arkime's help can help you map them.
- Despite considerable overlap, there are differences in protocol parser support between Zeek and Arkime. In particular, Malcolm's configuration of Zeek parses many more ICS protocols than Arkime.

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"
```



As previously discussed, Arkime generates session records containing metadata about network connections. Zeek generates similar session metadata, linking network events to sessions via a connection UID. Malcolm aims to facilitate analysis of Zeek logs by mapping values from Zeek logs to the Arkime session database schema for equivalent fields, and by creating new "native" Arkime database fields for all the other Zeek log values for which there is not currently an equivalent in Arkime. The *Fields* section of Arkime's help provides a list of known fields across both the Arkime and Zeek data sources.

In this way, when full packet capture is an option, analysis of PCAP files can be enhanced by the additional information Zeek provides. When full packet capture is not an option, similar analysis can still be performed using the same interfaces and processes using the Zeek logs alone. The values of records created from Zeek logs can be expanded and viewed like any native Arkime session by clicking the plus icon to the left of the record in the Sessions view. However, note that when dealing with these Zeek records the full packet contents are not available, so buttons dealing with viewing and exporting PCAP information will not behave as they would for records from PCAP files. Other than that, Zeek records and their values are usable in Malcolm just like native PCAP session records.

A few fields of particular mention that help limit returned results to those Zeek logs and Arkime session records generated from the same network connection are Community ID and Zeek's connection UID (zeek.uid), which Malcolm maps to Arkime's rootId field.

Community ID is a specification for standard flow hashing published by Corelight with

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!

47

As mentioned earlier, Zeek has the ability to “carve” files from a variety of protocols in observed network traffic. Malcolm leverages this feature to submit these carved files to a number of file scanning tools:

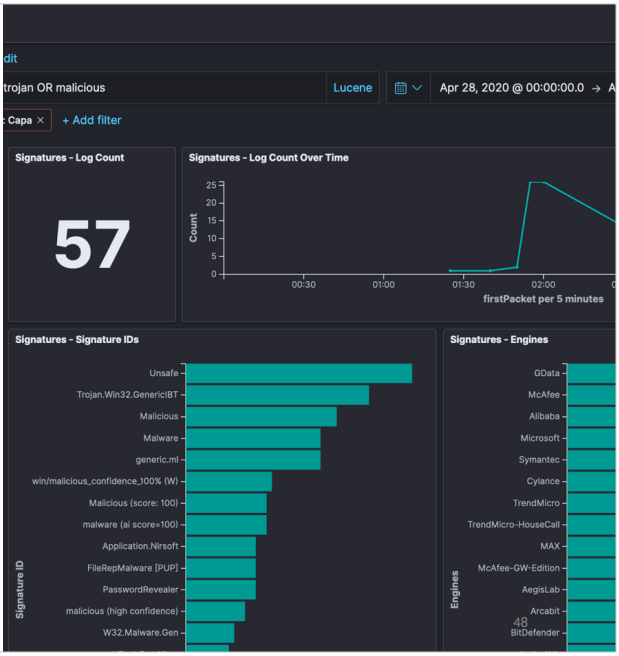
- ClamAV – an open source antivirus engine to scan for known malware signatures
- YARA – pattern matching swiss army knife using a curated list of security-related signatures or your own custom signatures
- Capa – portable executable capabilities analyzer
- VirusTotal – online database of file hashes (requires API token and internet connection)

This can be set globally as the default behavior (by modifying a configuration file in the Malcolm install directory prior to starting up) or on a per-PCAP basis when uploading using the upload web UI interface.

Malcolm can be configured to preserve files carved from network traffic (either all files or just “suspicious” ones that trigger the scanning engines) for further examination.

Signatures

- Signatures dashboard in Kibana 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



If Zeek file carving is enabled, questionable files will be written to the signatures log and reported in the signatures dashboard.

The Zeek connection UID (zeek.uid) and file UID (zeek.fuid) fields in these logs can be used to cross reference to other visualizations to provide the context for how file was transferred.

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 Kibana and `protocols == http` in Arkime)
- Use tags

49

Finally, here are a few tips for effective searching in Kibana and Arkime:

- Always check your search time frame. If you’re not seeing the data you’re expecting to see, often it’s because the data lies outside of the window of time you’re searching.
- An effective technique for investigating is to “zoom in” (for example, narrowing in on a particular file type transferred), pivot to another field (select the source IP address that transferred the file) then “zoom out” (removing the file type filter to see what other activity that source IP was involved in).
- Most elements in both Kibana and Arkime are interactive and can be configured to work with any of the more than 1000+ data fields Malcolm knows about.
- Learn how to filter on common fields like Zeek log type and network application protocol.
- Utilize the tags field in your searches, including prepopulated tags like based on public/private IP address space, tags populated based on the host and segment mapping you’ve configured and tags populated based on PCAP filenames.

Thank you!

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