Build Your Own SOC

Learn what a SOC can & cannot do

Kristen – kristen@srlabs.hk

Security Research Labs

Agenda

- **Introduction to SOC**
- Architecture overview
- Syslog forwarding
- Parsing
- Normalization
- High-fidelity starter alerts
- Recommendation of log sources

Security Operations Center (SOC) monitors security events to detect attacks and facilitate response

Example SOC room – Analysts looking at SIEM[1] dashboards & alerts

SOC functions

- **Log storage:** Easily query the logs from history
- **Log analysis:** Correlate logs to detect attacks
- **Alert:** Send alerts upon certain detection rules trigger
- **Common SIEM solutions:** Splunk, Sentinel, Elastic, Wazuh, etc

[1] SIEM – Security Information Event Management

Build your own SOC trains both red and blue team instincts

Benefits

Blue teamers

- Improve data quality of logs for SOC
- Experiment query language for detection setup
- Experiment threat hunting procedures
- Create & test honeypot or honey-token

Red teamers

- Understand SOC’s monitoring limitation
- Experiment with evasion techniques
  - Learn how to silent log forwarding
- Improve operational stealth

Self-hosters

- Protect your own infrastructure
- Gain threat intelligence insights
  - what wordlist people use to brute-force your web?
- Keep records of abuse and report the IP addresses

We ensure you can detect prevalent Active Directory and Web attacks with a successful SOC

	Benefits
Blue teamers	▪ Improve data quality of logs for SOC ▪ Experiment query language for detection setup ▪ Experiment threat hunting procedures ▪ Create & test honeypot or honey-token
Red teamers	▪ Understand SOC’s monitoring limitation ▪ Experiment with evasion techniques - Learn how to silent log forwarding ▪ Improve operational stealth
Self-hosters	▪ Protect your own infrastructure ▪ Gain threat intelligence insights - what wordlist people use to brute-force your web? ▪ Keep records of abuse and report the IP addresses
Takeaways	▪ Configure correct data format in SIEM ▪ Rsyslog forwarding cheat sheet ▪ Detections of **Kerberoasting, Bloodhound (AD enumeration), ADCS exploits** - will be covered in the last section “High-fidelity starter alerts” - real-time demo

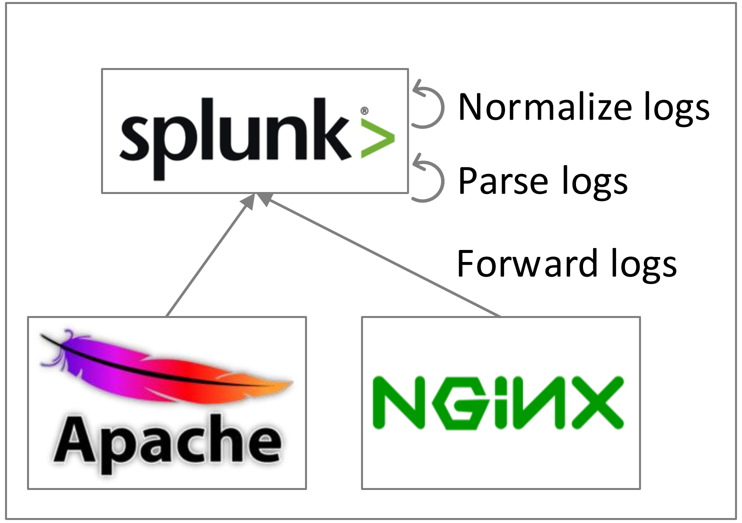
We use Web monitoring to illustrate how data quality (forwarding, parsing, normalization) affects detection

Attack scenario	▪ **Directory brute force:** Attacker throws wordlist onto Web's URL path to identify available resource paths
Detection logic	▪ High-volume of different URL path visited by single source IP in short amount of time
Log sources	▪ **Nginx access logs:** /var/log/nginx/access.log ▪ **Apache access logs:** /var/log/apache2/access.log

Next step

Forward logs to SIEM (Splunk)	▪ Configure Rsyslog to forward logs into SIEM (Splunk)
Parse logs	▪ Ensure fields are extracted (Source IP, URL Path, Timestamp)
Normalize logs	▪ Ensure field names between Nginx & Apache logs are the same, so we can reuse the detection rule on both application

Example environment



Agenda

- Introduction of SOC
- **Architecture overview**
- Syslog forwarding
- Parsing
- Normalization
- High-fidelity starter alerts
- Recommendation of log sources

Component overview and component mapping

	Log Sources	Pre-processor	Data Storage	Analytics	UI/ Monitoring
Tools mapping	Network Devices CI/CD Servers Web Servers Active Directory ...	logstash	elasticsearch	Elastic Security	kibana
		Cribl™	splunk>		
Function	▪ Generate security event logs	▪ Enrich log quality for parsing ▪ Normalize fields ▪ Route logs to SIEM	▪ Store logs for quick searching ▪ Real-time analysis & correlation on logs to detect security alerts		▪ UI for operations - configuration - logs browsing - visualization ▪ Generate alerts

- **ELK stack:** Open-Source version available with limited features (Logstash, Elasticsearch, Kibana)
- **Cribl:** Free version available with up to 1GB daily ingestion limit
- **Splunk:** Free version available up to 500MB daily ingestion limit

Agenda

- Introduction of SOC
- Architecture overview
- **Syslog forwarding**
- Parsing
- Normalization
- High-fidelity starter alerts
- Recommendation of log sources

Understanding log formats is the key to successful parsing, normalization, and detection

Raw logs

- Raw logs generated by applications, e.g.
  - SSH, sudo, Linux auditd, Web access logs, etc
- May not present all key information for analytics, e.g.
  - hostname & software name generated the logs

RFC 3164 & 5424

- Message format that ensures basic structure of meta data, e.g.
  - timestamp, hostname, application name, process ID, etc
- RFC 3164 is obsoleted, but still may be used by devices & widely recognized by SIEM

Common Event Format (CEF)

- Message format based on syslog to standardize additional meta-data structure, e.g.
  - device vendor, device product, device version, destinationHostName, deviceDnsDomain, etc
- Useful in normalizing (unifying) logs from different vendors

Common misconfiguration

- **Forwarding format:** Mismatched format with SIEM parser’s requirement
  - SIEM parsers have specific requirements on log formats
- **Timestamp:** Ambiguity of time zone
  - missing time zone
  - using abbreviation to represent time zone instead of numeric offset

Timestamp needs to be explicit as it is important for correlating events together for effective detections

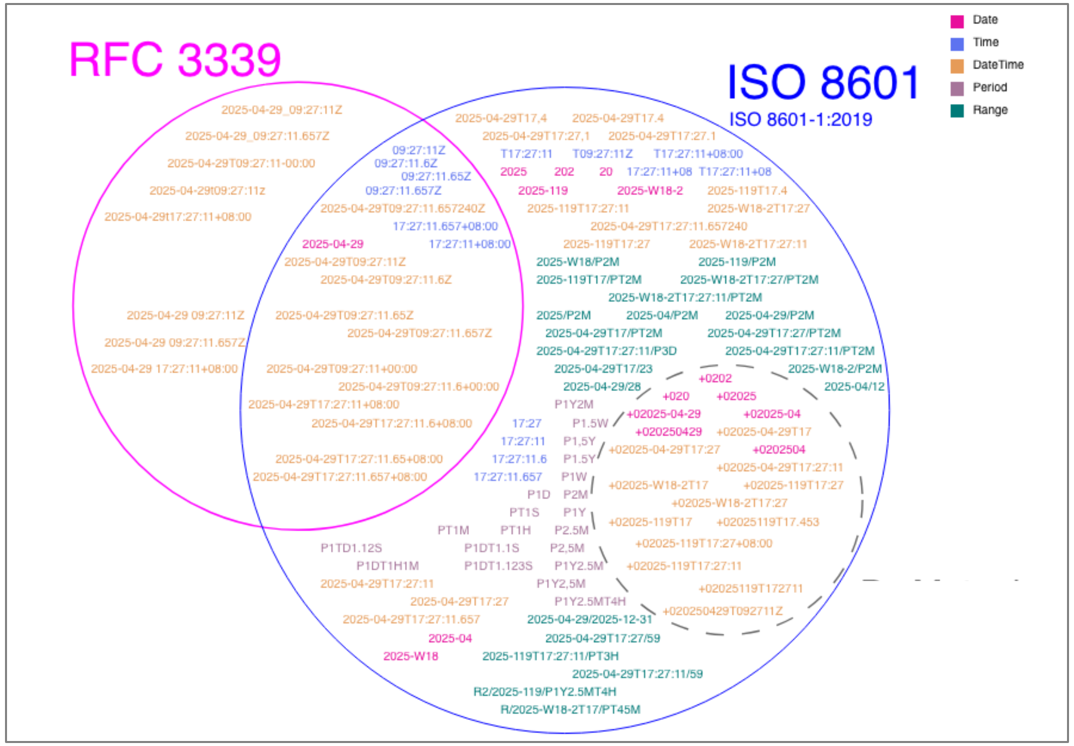
Misconfigurations

- **Missing time zone:** Fri, 02 May, 2025, 12:48:41
- SIEM assumes logs are in its local time zone, but log sources may be from different regions

- **Using abbreviation:** Fri, 02 May, 2025, 12:48:41 IST
- Abbreviation can have duplication, causing wrong time indexed, e.g.
  - IST can stand for Indian Standard Time (UTC+5:30) or Israel Standard Time (UTC+2:00)

Best practice
~~ISO 8601/~~
~~RFC 3389~~

- **Use numeric offset for time zone:**
  - 2025-05-02T12:48:31**+05:30**
  (yyyy-MM-ddThh:mm:ss+TZ)
- **Reference standard:**
  - Overlap[1] between RFC3389 & ISO8601



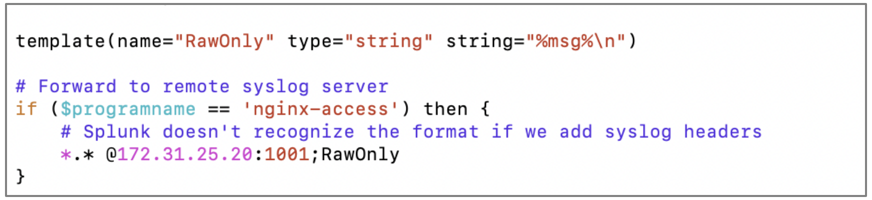
[1] RFC3389 vs ISO8601 – https://ijmacd.github.io/rfc3339-iso8601/

Rsyslog configuration cheat sheet

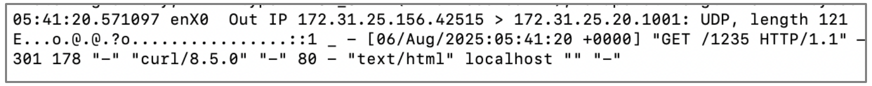
Rsyslog configuration & tcpdump verification

Use case

Raw log forwarding



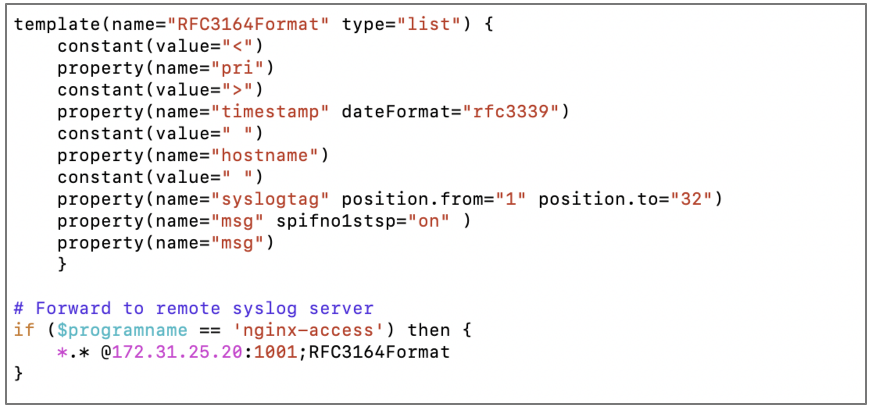
```
template(name="RawOnly" type="string" string="%msg%\n")

# Forward to remote syslog server
if ($programname == 'nginx-access') then {
    # Splunk doesn't recognize the format if we add syslog headers
    *.* @172.31.25.20:1001;RawOnly
}
```



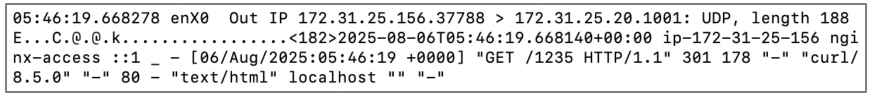
```
05:41:20.571097 enX0  Out IP 172.31.25.156.42515 > 172.31.25.20.1001: UDP, length 121
E...o.@.@.?o.................::1 _ - [06/Aug/2025:05:41:20 +0000] "GET /1235 HTTP/1.1" -
301 178 "-" "curl/8.5.0" "-" 80 - "text/html" localhost "" "-"
```

- The parser only recognize raw format without syslog headers

RFC 3164 forwarding



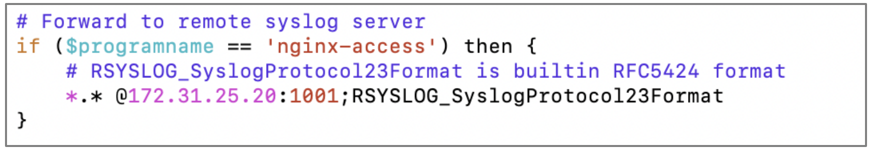
```
template(name="RFC3164Format" type="list") {
    constant(value="<")
    property(name="pri")
    constant(value=">")
    property(name="timestamp" dateFormat="rfc3339")
    constant(value=" ")
    property(name="hostname")
    constant(value=" ")
    property(name="syslogtag" position.from="1" position.to="32")
    property(name="msg" spifno1stsp="on" )
    property(name="msg")
    }

# Forward to remote syslog server
if ($programname == 'nginx-access') then {
    *.* @172.31.25.20:1001;RFC3164Format
}
```



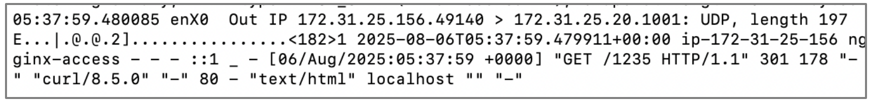
```
05:46:19.668278 enX0  Out IP 172.31.25.156.37788 > 172.31.25.20.1001: UDP, length 188
E...C.@.@.k.................<182>2025-08-06T05:46:19.668140+00:00 ip-172-31-25-156 ngi
nx-access ::1 _ - [06/Aug/2025:05:46:19 +0000] "GET /1235 HTTP/1.1" 301 178 "-" "curl/
8.5.0" "-" 80 - "text/html" localhost "" "-"
```

- The parser may only recognize format when RFC 3164 syslog headers are added

RFC 5424 forwarding



```
# Forward to remote syslog server
if ($programname == 'nginx-access') then {
    # RSYSLOG_SyslogProtocol23Format is builtin RFC5424 format
    *.* @172.31.25.20:1001;RSYSLOG_SyslogProtocol23Format
}
```



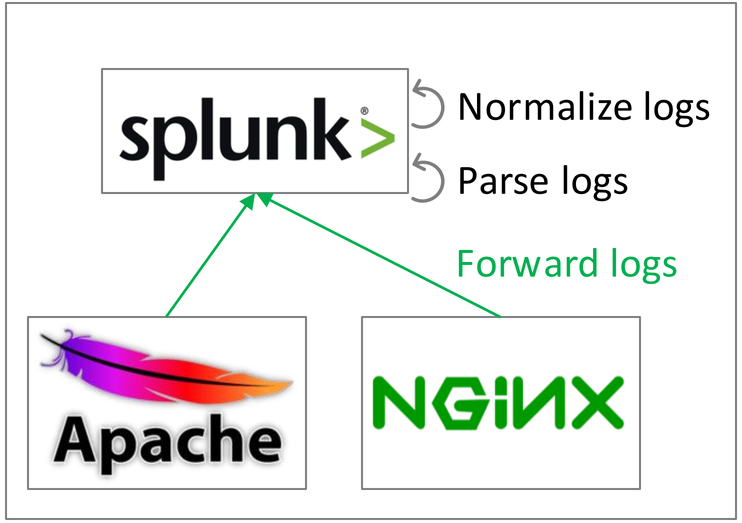
```
05:37:59.480085 enX0  Out IP 172.31.25.156.49140 > 172.31.25.20.1001: UDP, length 197
E...|.@.@.2]................<182>1 2025-08-06T05:37:59.479911+00:00 ip-172-31-25-156 ng
ginx-access - - - ::1 _ - [06/Aug/2025:05:37:59 +0000] "GET /1235 HTTP/1.1" 301 178 "-
" "curl/8.5.0" "-" 80 - "text/html" localhost "" "-"
```

- The parser may only recognize format when RFC 5424 syslog headers are added

After forwarding logs, we need to configure parsing to ensure fields are extracted successfully

	Attack scenario	▪ **Directory brute force:** Attacker throws wordlist onto Web's URL path to identify available resource paths
	Detection logic	▪ High-volume of different URL path visited by single source IP in short amount of time
	Log sources	▪ **Nginx access logs:** /var/log/nginx/access.log ▪ **Apache access logs:** /var/log/apache2/access.log
Done	**Forward logs to SIEM (Splunk)**	▪ Configure Rsyslog to forward logs into SIEM (Splunk)
Next step	**Parse logs**	▪ Ensure fields are extracted (Source IP, URL Path, Timestamp)
	Normalize logs	▪ Ensure field names between Nginx & Apache logs are the same, so we can reuse the detection rule on both application

Example environment

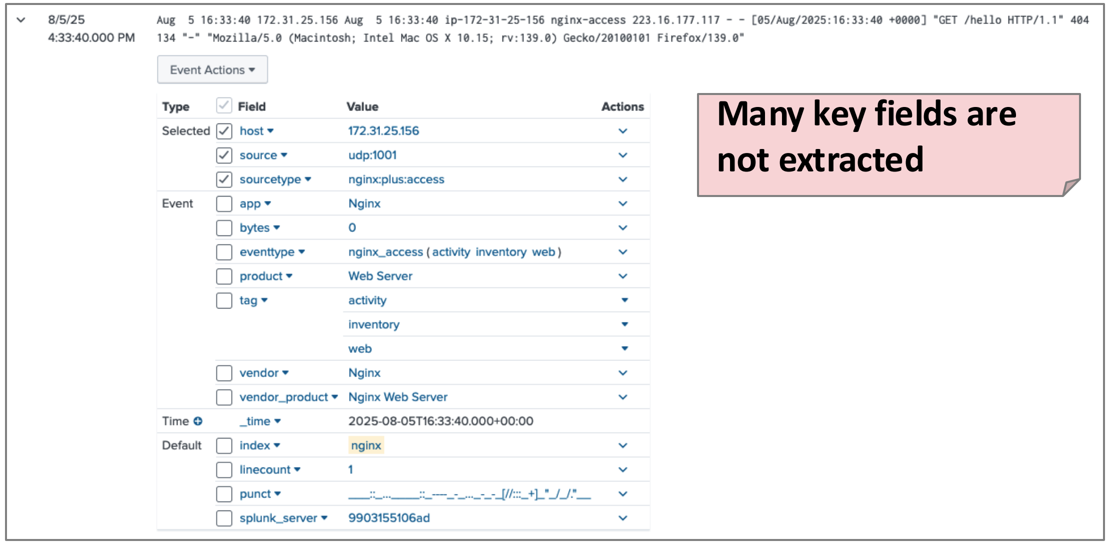


Agenda

- Introduction of SOC
- Architecture overview
- Syslog forwarding
- **Parsing**
- Normalization
- High-fidelity starter alerts
- Recommendation of log sources

Parsing refers to extracting values into keywords during logs processing. It enables keyword search instead of full text search

Unparsed logs – URL, IP address not extracted



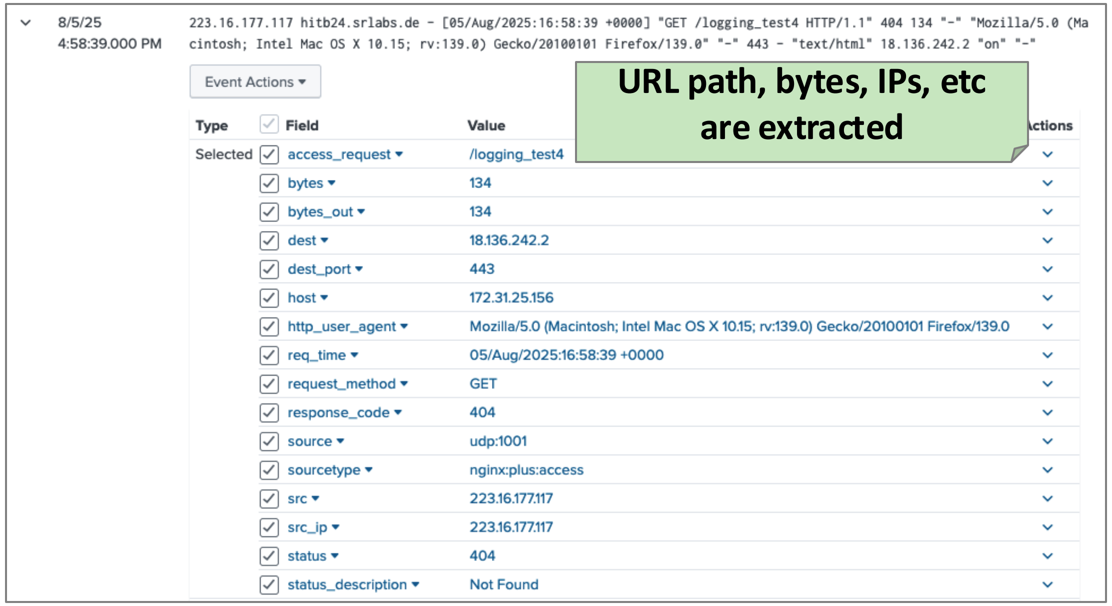
8/5/25
4:33:40.000 PM

```
Aug  5 16:33:40 172.31.25.156 Aug  5 16:33:40 ip-172-31-25-156 nginx-access 223.16.177.117 - - [05/Aug/2025:16:33:40 +0000] "GET /hello HTTP/1.1" 404
134 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:139.0) Gecko/20100101 Firefox/139.0"
```

Type	☑ Field	Value	Actions
Selected	☑ host ▾	172.31.25.156	⌄
	☑ source ▾	udp:1001	⌄
	☑ sourcetype ▾	nginx:plus:access	⌄
Event	□ app ▾	Nginx	⌄
	□ bytes ▾	0	⌄
	□ eventtype ▾	nginx_access (activity inventory web)	⌄
	□ product ▾	Web Server	⌄
	□ tag ▾	activity	▾
		inventory	▾
		web	▾
	□ vendor ▾	Nginx	⌄
	□ vendor_product ▾	Nginx Web Server	⌄
Time ⊕	_time ▾	2025-08-05T16:33:40.000+00:00	
Default	□ index ▾	nginx	⌄
	□ linecount ▾	1	⌄
	□ punct ▾	____::_...______::_----_-_..._-_-_[//:::_+]_"_/_/."___	⌄
	□ splunk_server ▾	9903155106ad	⌄

Many key fields are not extracted

Log parsed – successful URL extraction



8/5/25
4:58:39.000 PM

```
223.16.177.117 hitb24.srlabs.de - [05/Aug/2025:16:58:39 +0000] "GET /logging_test4 HTTP/1.1" 404 134 "-" "Mozilla/5.0 (Ma
cintosh; Intel Mac OS X 10.15; rv:139.0) Gecko/20100101 Firefox/139.0" "-" 443 - "text/html" 18.136.242.2 "on" "-"
```

URL path, bytes, IPs, etc are extracted

Type	☑ Field	Value	Actions
Selected	☑ access_request ▾	/logging_test4	⌄
	☑ bytes ▾	134	⌄
	☑ bytes_out ▾	134	⌄
	☑ dest ▾	18.136.242.2	⌄
	☑ dest_port ▾	443	⌄
	☑ host ▾	172.31.25.156	⌄
	☑ http_user_agent ▾	Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:139.0) Gecko/20100101 Firefox/139.0	⌄
	☑ req_time ▾	05/Aug/2025:16:58:39 +0000	⌄
	☑ request_method ▾	GET	⌄
	☑ response_code ▾	404	⌄
	☑ source ▾	udp:1001	⌄
	☑ sourcetype ▾	nginx:plus:access	⌄
	☑ src ▾	223.16.177.117	⌄
	☑ src_ip ▾	223.16.177.117	⌄
	☑ status ▾	404	⌄
	☑ status_description ▾	Not Found	⌄

- **Parsing improves search speed by enabling keyword search instead of full text search!**

SIEM do not automatically recognize and extract values from logs, configurations are required

	Log Sources	Pre-processor	Data Storage	Analytics	UI/ Monitoring
Tools mapping	Network Devices CI/CD Servers Active Directory …	logstash Cribl™	elasticsearch splunk>	Elastic Security	kibana
Parsing options	**Agent-based parsing:** ▪ Agent software on source devices - Elastic-agent - filebeat - Splunk universal forwarder	**Middle-ware parsing:** ▪ Dedicated intermediate software - Elastic Logstash - Cribl	**Index-time parsing:** ▪ Prebuilt plugin available on the SIEM - Elastic Integrations - Splunk add-ons ▪ Custom Regex expressions - Elastic ingest pipeline - Splunk props.conf & transform.conf		**Search-time parsing:** ▪ Inline regex extraction on raw log at search time

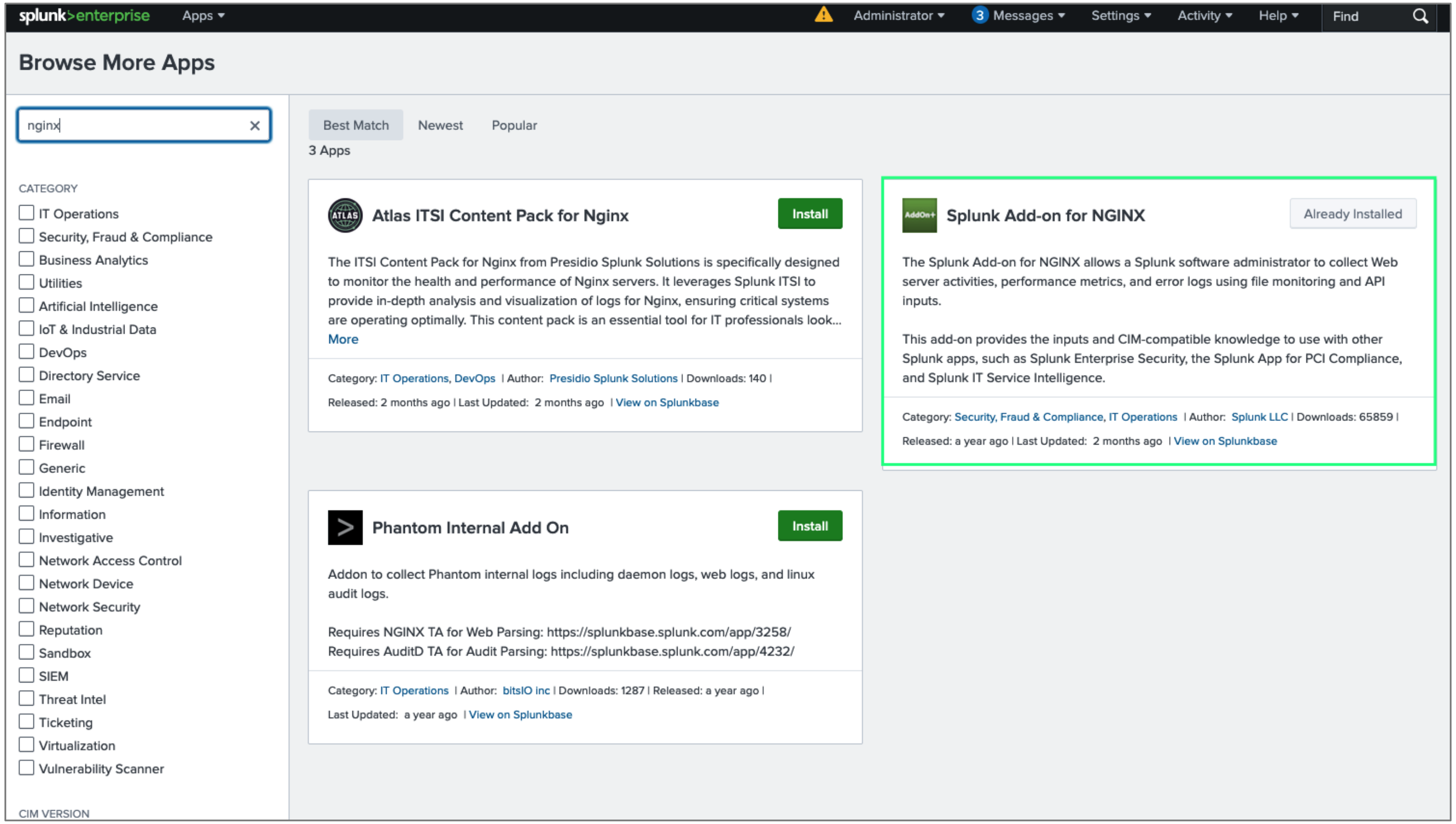
Suggested approach

Flexibly choose parsing methods

1. Check if pre-built plugin/ add-on available for parsing the logs
2. Check if you can parse logs from agent software (Elastic-agent, Splunk universal forwarder)
3. Write your regex extractions at Index-time or search-time as the last resort

Hands-on parsing configuration (1/5) – Nginx Access Logs
Check & install built-in parser add-on in SIEM

Download parsing add-on for nginx



Browse More Apps

nginx

3 Apps

Atlas ITSI Content Pack for Nginx

The ITSI Content Pack for Nginx from Presidio Splunk Solutions is specifically designed to monitor the health and performance of Nginx servers. It leverages Splunk ITSI to provide in-depth analysis and visualization of logs for Nginx, ensuring critical systems are operating optimally. This content pack is an essential tool for IT professionals look... More

Category: IT Operations, DevOps | Author: Presidio Splunk Solutions | Downloads: 140 |
Released: 2 months ago | Last Updated: 2 months ago | View on Splunkbase

Splunk Add-on for NGINX

The Splunk Add-on for NGINX allows a Splunk software administrator to collect Web server activities, performance metrics, and error logs using file monitoring and API inputs.

This add-on provides the inputs and CIM-compatible knowledge to use with other Splunk apps, such as Splunk Enterprise Security, the Splunk App for PCI Compliance, and Splunk IT Service Intelligence.

Category: Security, Fraud & Compliance, IT Operations | Author: Splunk LLC | Downloads: 65859 |
Released: a year ago | Last Updated: 2 months ago | View on Splunkbase

Phantom Internal Add On

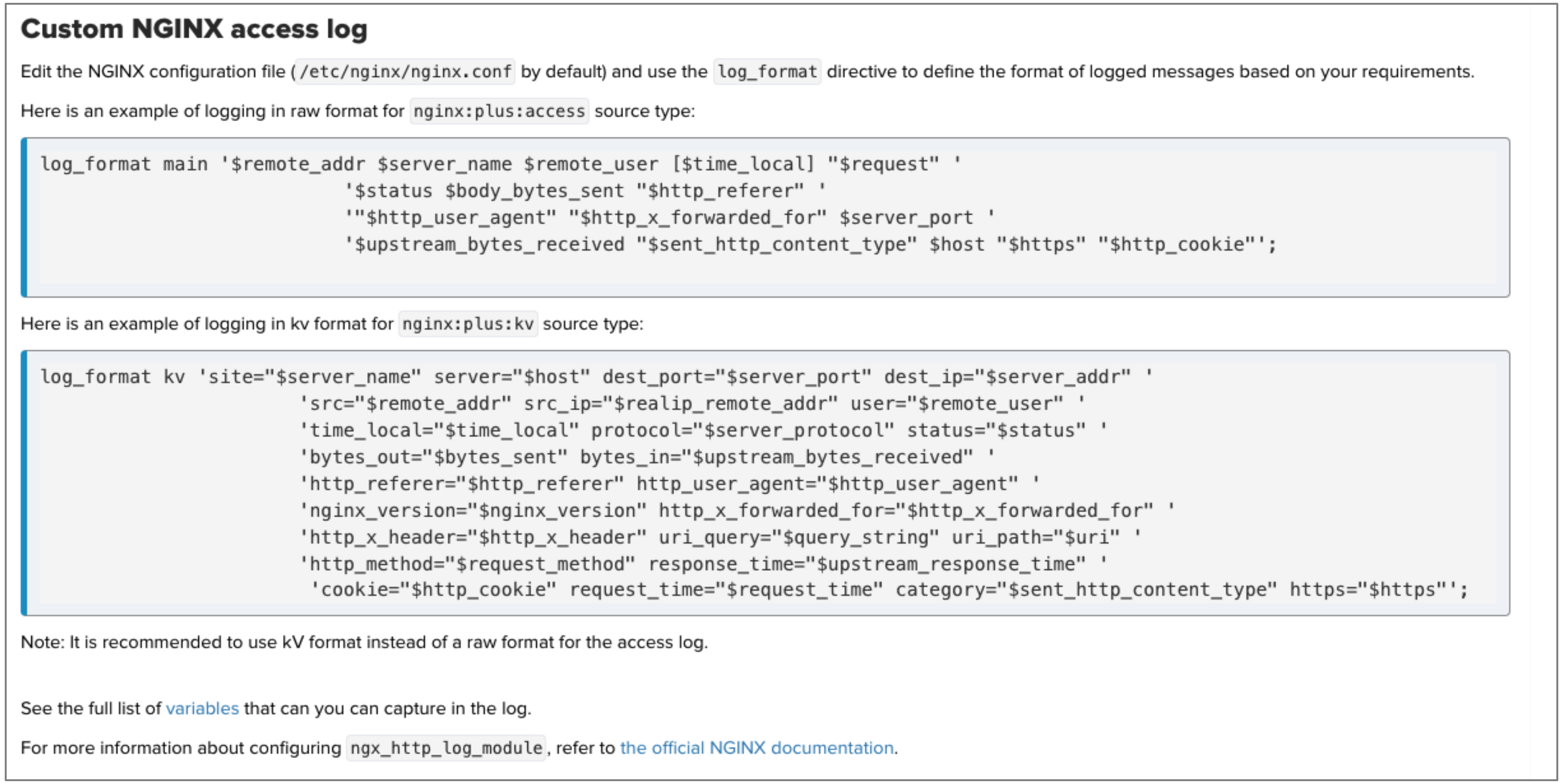
Addon to collect Phantom internal logs including daemon logs, web logs, and linux audit logs.

Requires NGINX TA for Web Parsing: https://splunkbase.splunk.com/app/3258/
Requires AuditD TA for Audit Parsing: https://splunkbase.splunk.com/app/4232/

Category: IT Operations | Author: bitsIO inc | Downloads: 1287 | Released: a year ago |
Last Updated: a year ago | View on Splunkbase

Hands-on parsing configuration (2/5) – Nginx Access Logs
Read the doc on the log format requirements for parsing

Follow instruction to modify /etc/nginx/nginx.conf to ensure log format is parsable



Custom NGINX access log

Edit the NGINX configuration file (`/etc/nginx/nginx.conf` by default) and use the `log_format` directive to define the format of logged messages based on your requirements.

Here is an example of logging in raw format for `nginx:plus:access` source type:

```
log_format main '$remote_addr $server_name $remote_user [$time_local] "$request" '
                           '$status $body_bytes_sent "$http_referer" '
                           '"$http_user_agent" "$http_x_forwarded_for" $server_port '
                           '$upstream_bytes_received "$sent_http_content_type" $host "$https" "$http_cookie"';
```

Here is an example of logging in kv format for `nginx:plus:kv` source type:

```
log_format kv 'site="$server_name" server="$host" dest_port="$server_port" dest_ip="$server_addr" '
                       'src="$remote_addr" src_ip="$realip_remote_addr" user="$remote_user" '
                       'time_local="$time_local" protocol="$server_protocol" status="$status" '
                       'bytes_out="$bytes_sent" bytes_in="$upstream_bytes_received" '
                       'http_referer="$http_referer" http_user_agent="$http_user_agent" '
                       'nginx_version="$nginx_version" http_x_forwarded_for="$http_x_forwarded_for" '
                       'http_x_header="$http_x_header" uri_query="$query_string" uri_path="$uri" '
                       'http_method="$request_method" response_time="$upstream_response_time" '
                        'cookie="$http_cookie" request_time="$request_time" category="$sent_http_content_type" https="$https"';
```

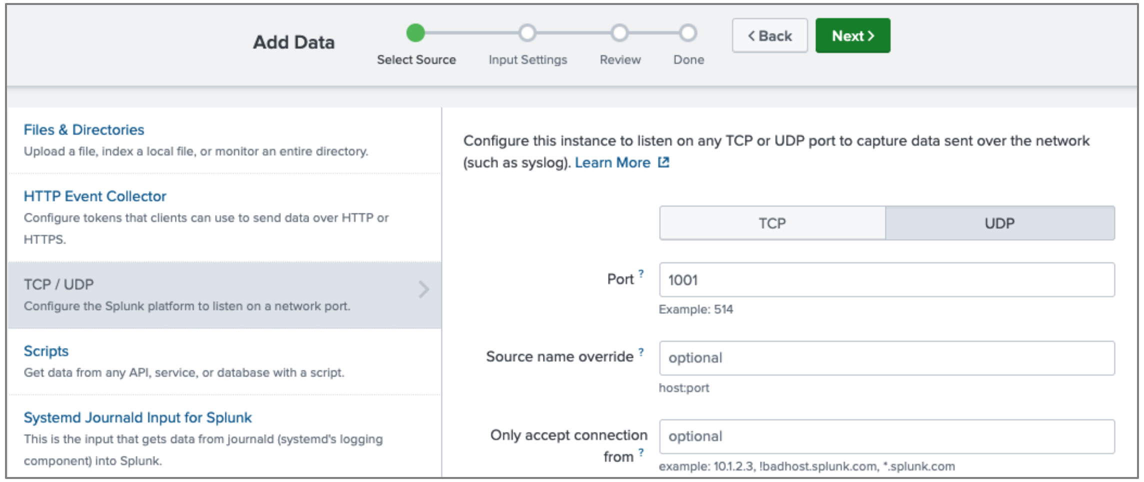
Note: It is recommended to use kV format instead of a raw format for the access log.

See the full list of variables that can you can capture in the log.

For more information about configuring `ngx_http_log_module`, refer to the official NGINX documentation.

Hands-on parsing configuration (3/5) – Nginx Access Logs
Configure Splunk input receiver for Nginx Access Logs

Open UDP listener on UDP/1001



Files & Directories
Upload a file, index a local file, or monitor an entire directory.

HTTP Event Collector
Configure tokens that clients can use to send data over HTTP or HTTPS.

TCP / UDP
Configure the Splunk platform to listen on a network port.

Scripts
Get data from any API, service, or database with a script.

Systemd Journald Input for Splunk
This is the input that gets data from journald (systemd's logging component) into Splunk.

Configure this instance to listen on any TCP or UDP port to capture data sent over the network (such as syslog). Learn More

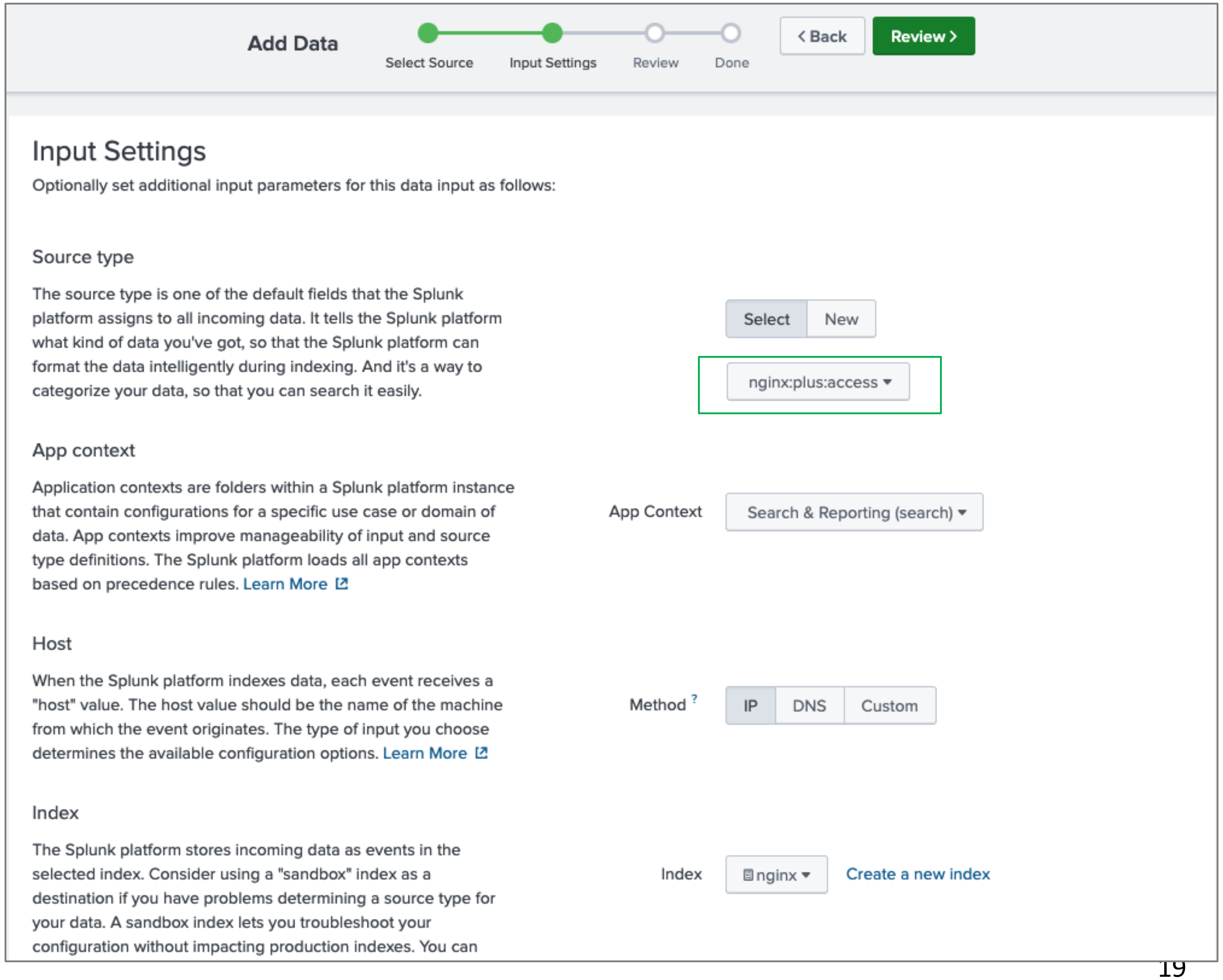
TCP | UDP

Port ?: 1001
Example: 514

Source name override ?: optional
host:port

Only accept connection from ?: optional
example: 10.1.2.3, !badhost.splunk.com, *.splunk.com

Configure Source type as “nginx:plus:access” to enable parsing



Input Settings

Optionally set additional input parameters for this data input as follows:

Source type

The source type is one of the default fields that the Splunk platform assigns to all incoming data. It tells the Splunk platform what kind of data you've got, so that the Splunk platform can format the data intelligently during indexing. And it's a way to categorize your data, so that you can search it easily.

Select | New

nginx:plus:access ▾

App context

Application contexts are folders within a Splunk platform instance that contain configurations for a specific use case or domain of data. App contexts improve manageability of input and source type definitions. The Splunk platform loads all app contexts based on precedence rules. Learn More

App Context: Search & Reporting (search) ▾

Host

When the Splunk platform indexes data, each event receives a "host" value. The host value should be the name of the machine from which the event originates. The type of input you choose determines the available configuration options. Learn More

Method ?: IP | DNS | Custom

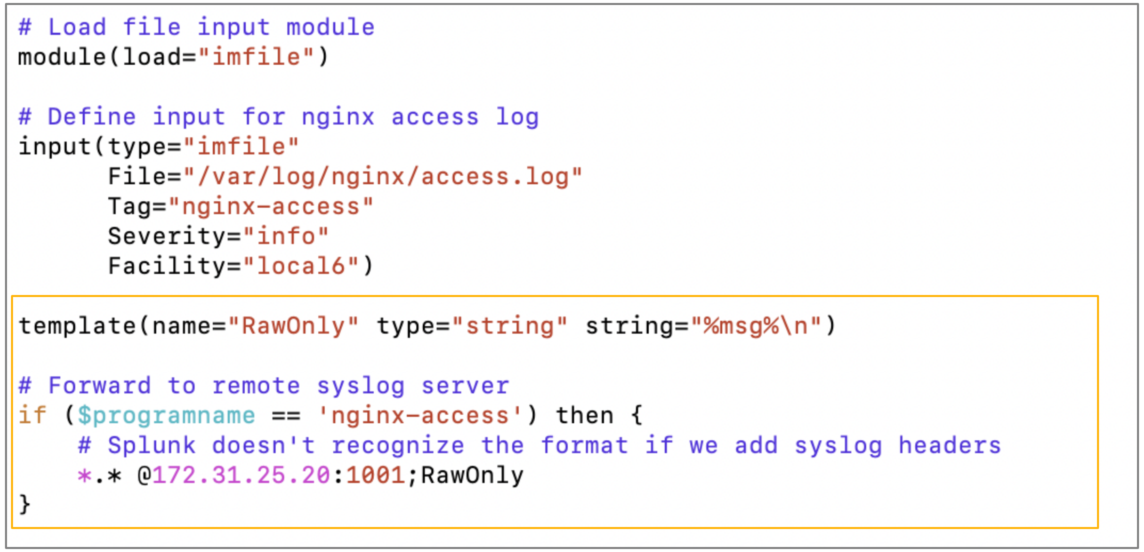
Index

The Splunk platform stores incoming data as events in the selected index. Consider using a "sandbox" index as a destination if you have problems determining a source type for your data. A sandbox index lets you troubleshoot your configuration without impacting production indexes. You can

Index: nginx ▾ Create a new index

Hands-on parsing configuration (4/5) – Nginx Access Logs
Configure Rsyslog to forward nginx access logs without syslog header

Configure Rsyslog to forward nginx logs in Raw format

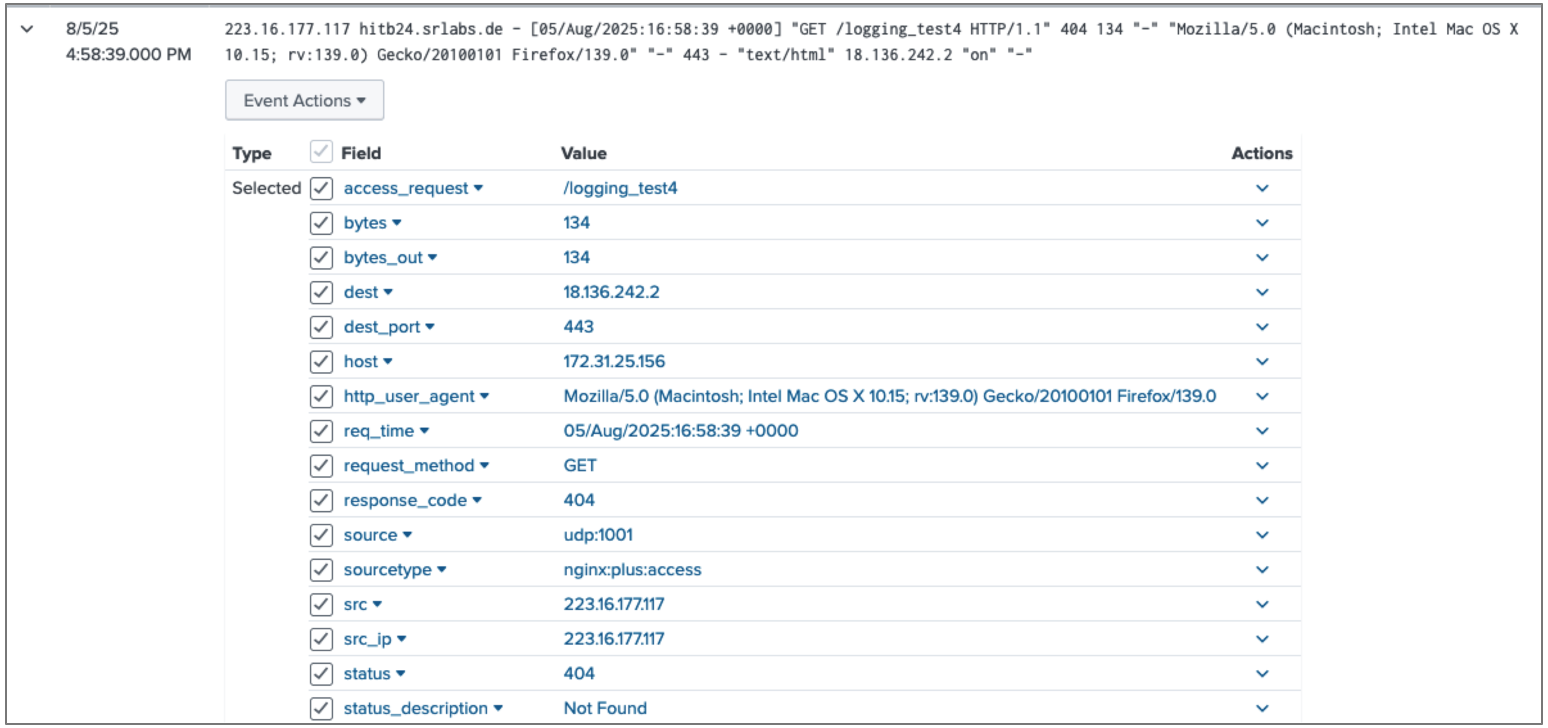


```
# Load file input module
module(load="imfile")

# Define input for nginx access log
input(type="imfile"
      File="/var/log/nginx/access.log"
      Tag="nginx-access"
      Severity="info"
      Facility="local6")

template(name="RawOnly" type="string" string="%msg%\n")

# Forward to remote syslog server
if ($programname == 'nginx-access') then {
    # Splunk doesn't recognize the format if we add syslog headers
    *.* @172.31.25.20:1001;RawOnly
}
```

We need to forward the raw logs in this case because Splunk doesn't recognize the format if syslog headers are added

Hands-on parsing configuration (5/5) – Nginx Access Logs
Verify log quality in Splunk

Verify important fields are extracted successfully



8/5/25
4:58:39.000 PM

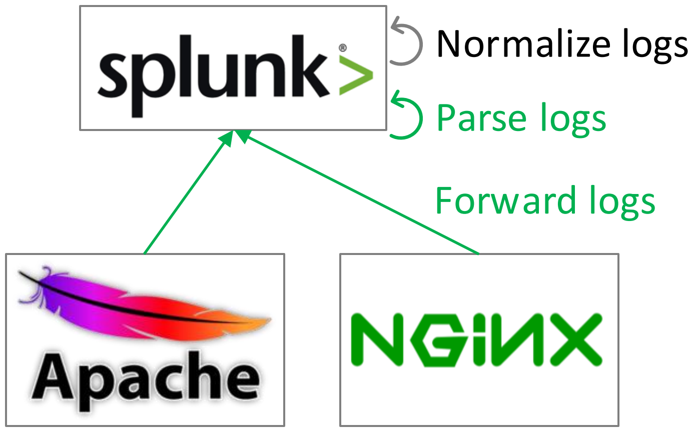
```
223.16.177.117 hitb24.srlabs.de - [05/Aug/2025:16:58:39 +0000] "GET /logging_test4 HTTP/1.1" 404 134 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X
10.15; rv:139.0) Gecko/20100101 Firefox/139.0" "-" 443 - "text/html" 18.136.242.2 "on" "-"
```

Type	☑	Field	Value	Actions
Selected	☑	access_request ▾	/logging_test4	⌄
	☑	bytes ▾	134	⌄
	☑	bytes_out ▾	134	⌄
	☑	dest ▾	18.136.242.2	⌄
	☑	dest_port ▾	443	⌄
	☑	host ▾	172.31.25.156	⌄
	☑	http_user_agent ▾	Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:139.0) Gecko/20100101 Firefox/139.0	⌄
	☑	req_time ▾	05/Aug/2025:16:58:39 +0000	⌄
	☑	request_method ▾	GET	⌄
	☑	response_code ▾	404	⌄
	☑	source ▾	udp:1001	⌄
	☑	sourcetype ▾	nginx:plus:access	⌄
	☑	src ▾	223.16.177.117	⌄
	☑	src_ip ▾	223.16.177.117	⌄
	☑	status ▾	404	⌄
	☑	status_description ▾	Not Found	⌄

With parsing completed, normalizing fields between Nginx and Apache logs is the last step before detection engineering

	Attack scenario	▪ **Directory brute force:** Attacker throws wordlist onto Web's URL path to identify available resource paths
	Detection logic	▪ High-volume of different URL path visited by single source IP in short amount of time
	Log sources	▪ **Nginx access logs:** /var/log/nginx/access.log ▪ **Apache access logs:** /var/log/apache2/access.log
Done	**Forward logs to SIEM (Splunk)**	▪ Configure Rsyslog to forward logs into SIEM (Splunk)
Done	**Parse logs**	▪ Ensure fields are extracted (Source IP, URL Path, Timestamp)
Next step	**Normalize logs**	▪ Ensure field names between Nginx & Apache logs are the same, so we can reuse the detection rule on both application

Example environment

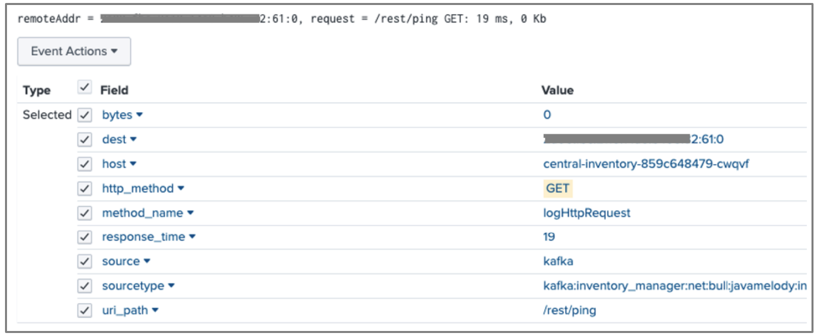


Agenda

- Introduction of SOC
- Architecture overview
- Syslog forwarding
- Parsing
- **Normalization**
- High-fidelity starter alerts
- Recommendation of log sources

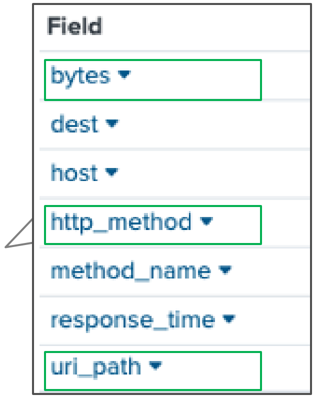
Normalization refers to unifying the data naming and types in SIEM for fully-covered search

Random Web1 log parsing result



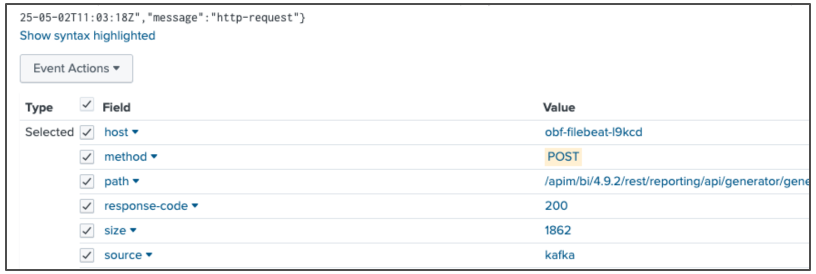
```
remoteAddr = [redacted]2:61:0, request = /rest/ping GET: 19 ms, 0 Kb
```

Type	☑ Field	Value
Selected	☑ bytes ▾	0
	☑ dest ▾	[redacted]2:61:0
	☑ host ▾	central-inventory-859c648479-cwqvf
	☑ http_method ▾	GET
	☑ method_name ▾	logHttpRequest
	☑ response_time ▾	19
	☑ source ▾	kafka
	☑ sourcetype ▾	kafka:inventory_manager:net:bull:javamelody:in
	☑ uri_path ▾	/rest/ping



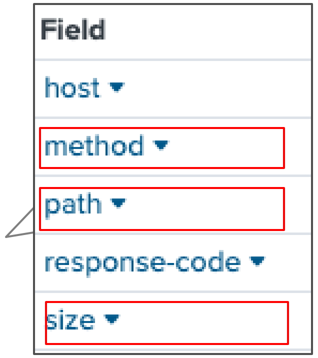
Field
bytes ▾
dest ▾
host ▾
http_method ▾
method_name ▾
response_time ▾
uri_path ▾

Random Web2 log parsing result



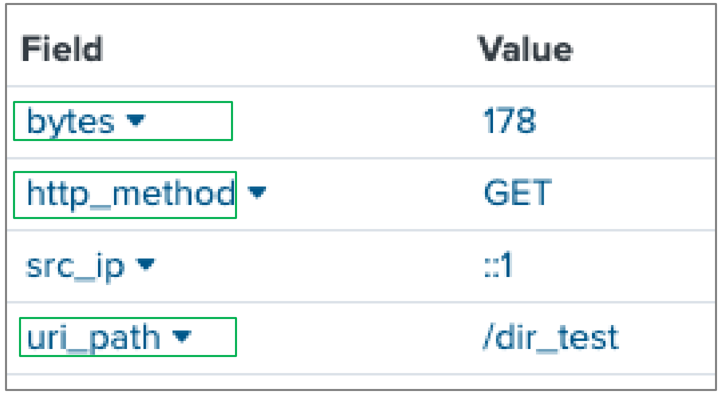
```
25-05-02T11:03:18Z","message":"http-request"}
```

Type	☑ Field	Value
Selected	☑ host ▾	obf-filebeat-l9kcd
	☑ method ▾	POST
	☑ path ▾	/apim/bi/4.9.2/rest/reporting/api/generator/gene
	☑ response-code ▾	200
	☑ size ▾	1862
	☑ source ▾	kafka



Field
host ▾
method ▾
path ▾
response-code ▾
size ▾

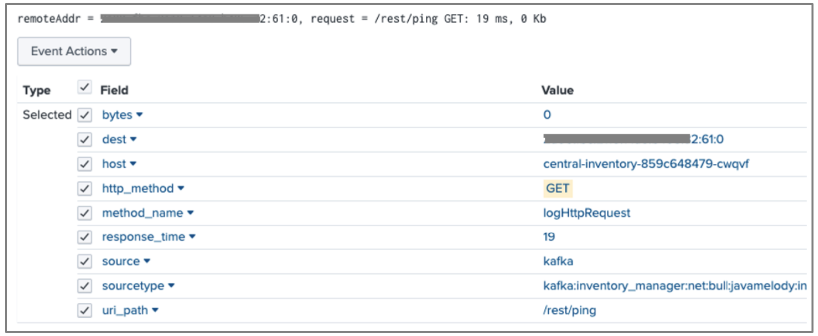
Unify names



Field	Value
bytes ▾	178
http_method ▾	GET
src_ip ▾	::1
uri_path ▾	/dir_test

Normalization refers to unifying the data naming and types in SIEM for fully-covered search

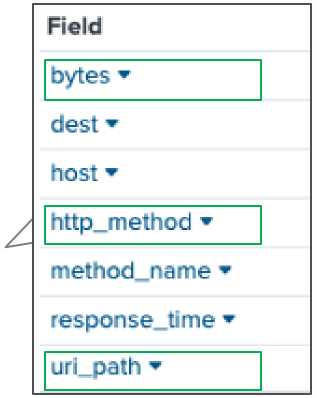
Random Web1 log parsing result



```
remoteAddr = [redacted]2:61:0, request = /rest/ping GET: 19 ms, 0 Kb
```

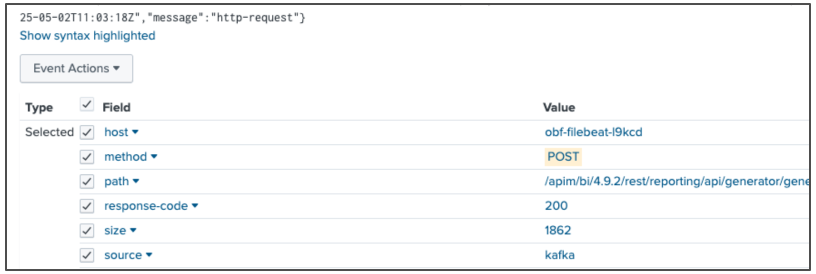
Event Actions ▾

Type	☑	Field	Value
Selected	☑	bytes ▾	0
	☑	dest ▾	[redacted]2:61:0
	☑	host ▾	central-inventory-859c648479-cwqvf
	☑	http_method ▾	GET
	☑	method_name ▾	logHttpRequest
	☑	response_time ▾	19
	☑	source ▾	kafka
	☑	sourcetype ▾	kafka:inventory_manager:net:bull:javamelody:in
	☑	uri_path ▾	/rest/ping



Field
bytes ▾
dest ▾
host ▾
http_method ▾
method_name ▾
response_time ▾
uri_path ▾

Random Web2 log parsing result

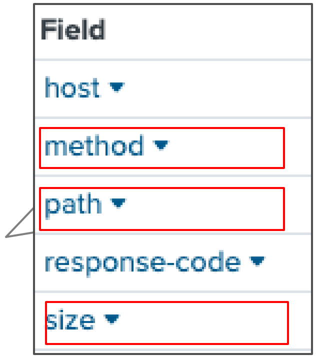


```
25-05-02T11:03:18Z","message":"http-request"}
```

Show syntax highlighted

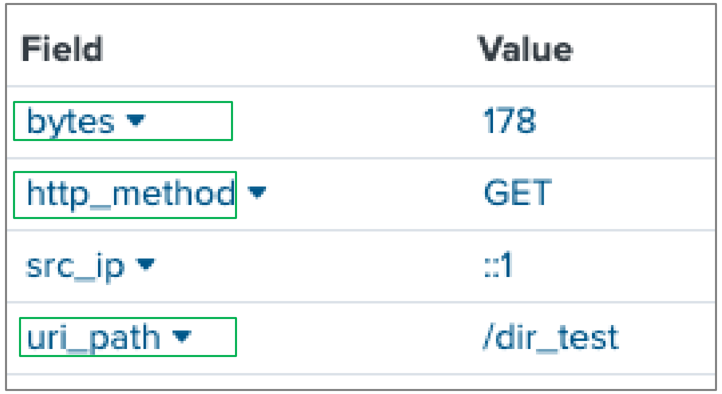
Event Actions ▾

Type	☑	Field	Value
Selected	☑	host ▾	obf-filebeat-l9kcd
	☑	method ▾	POST
	☑	path ▾	/apim/bi/4.9.2/rest/reporting/api/generator/gene
	☑	response-code ▾	200
	☑	size ▾	1862
	☑	source ▾	kafka



Field
host ▾
method ▾
path ▾
response-code ▾
size ▾

Unify names



Field	Value
bytes ▾	178
http_method ▾	GET
src_ip ▾	::1
uri_path ▾	/dir_test

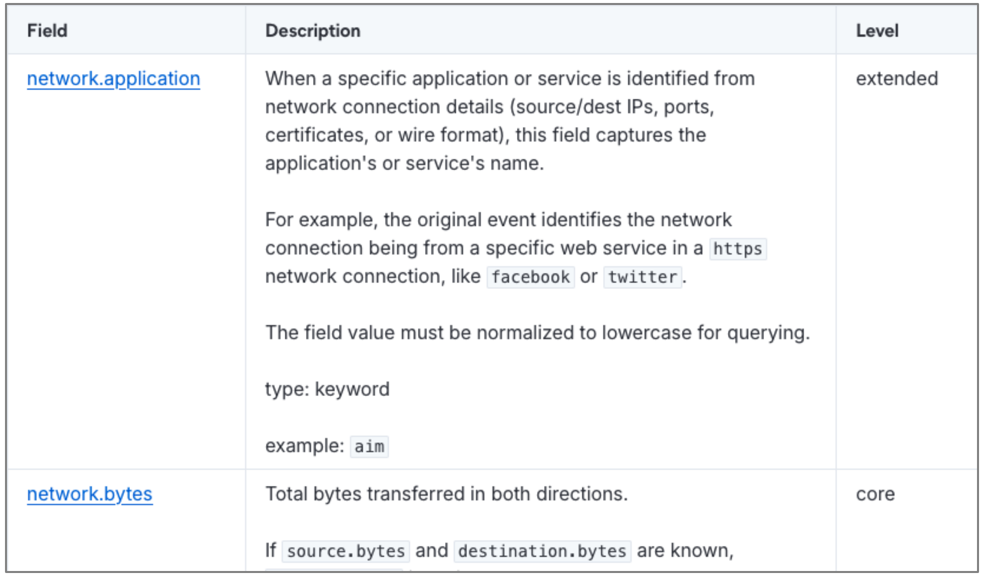
What naming scheme should we use?

Each SIEM platform offers its own guideline for naming scheme & data types, but no universal golden standard exists

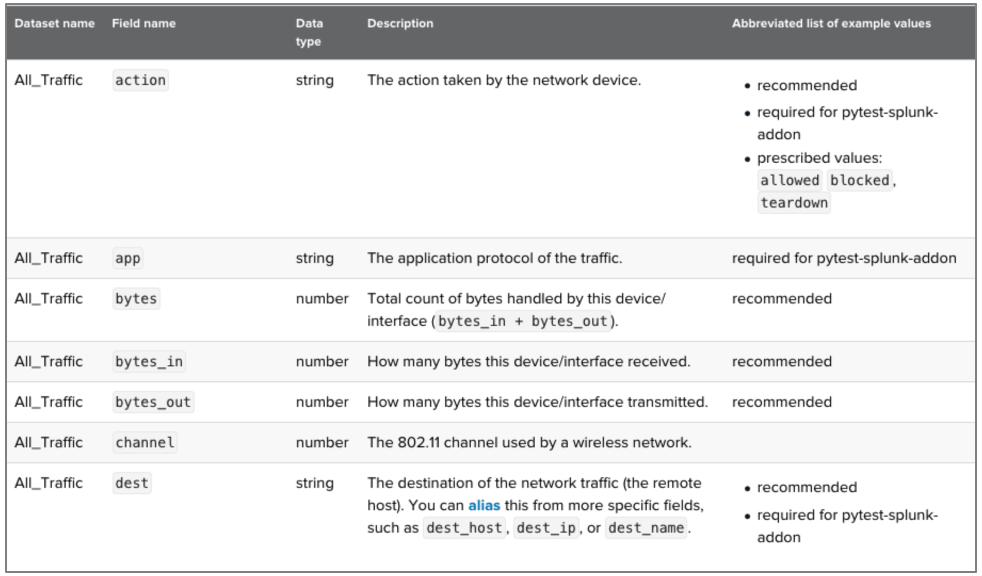
Naming schemes	
	- **Elastic:** ECS (Elastic Common Schema) - **Splunk:** CIM (Common Information Model)

Example (Network logs)

- **Elastic ECS 8.17** – source.ip, source.port, destination.ip, destination.port
- **Splunk CIM 6.0.3** – src_ip, src_port, dest_ip, dest_port



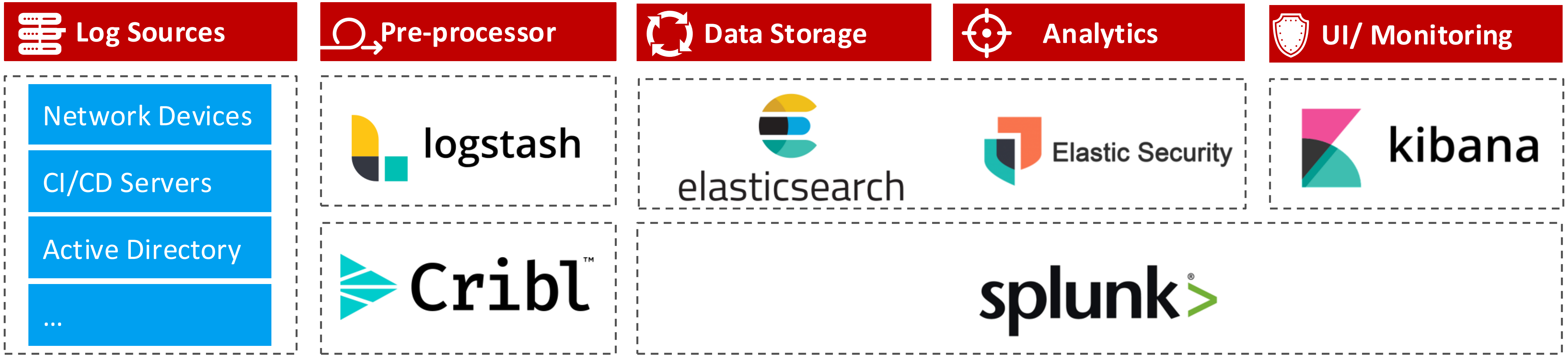
Field	Description	Level
network.application	When a specific application or service is identified from network connection details (source/dest IPs, ports, certificates, or wire format), this field captures the application's or service's name. For example, the original event identifies the network connection being from a specific web service in a `https` network connection, like `facebook` or `twitter`. The field value must be normalized to lowercase for querying. type: keyword example: `aim`	extended
network.bytes	Total bytes transferred in both directions. If `source.bytes` and `destination.bytes` are known,	core



Dataset name	Field name	Data type	Description	Abbreviated list of example values
All_Traffic	`action`	string	The action taken by the network device.	• recommended • required for pytest-splunk-addon • prescribed values: `allowed` `blocked`, `teardown`
All_Traffic	`app`	string	The application protocol of the traffic.	required for pytest-splunk-addon
All_Traffic	`bytes`	number	Total count of bytes handled by this device/ interface (`bytes_in + bytes_out`).	recommended
All_Traffic	`bytes_in`	number	How many bytes this device/interface received.	recommended
All_Traffic	`bytes_out`	number	How many bytes this device/interface transmitted.	recommended
All_Traffic	`channel`	number	The 802.11 channel used by a wireless network.	
All_Traffic	`dest`	string	The destination of the network traffic (the remote host). You can alias this from more specific fields, such as `dest_host`, `dest_ip`, or `dest_name`.	• recommended • required for pytest-splunk-addon

Ensuring normalization at parser setting is ideal but impractical to achieve, leaving search-time normalization the only viable option

Tools mapping



Log Sources

Pre-processor

Data Storage

Analytics

UI/ Monitoring

Network Devices

CI/CD Servers

Active Directory

…

Normalization configuration

Modify parser's logic to name fields consistently

- ✓ Fixing problem from the root
- ✗ Might break the parser or prevent future update

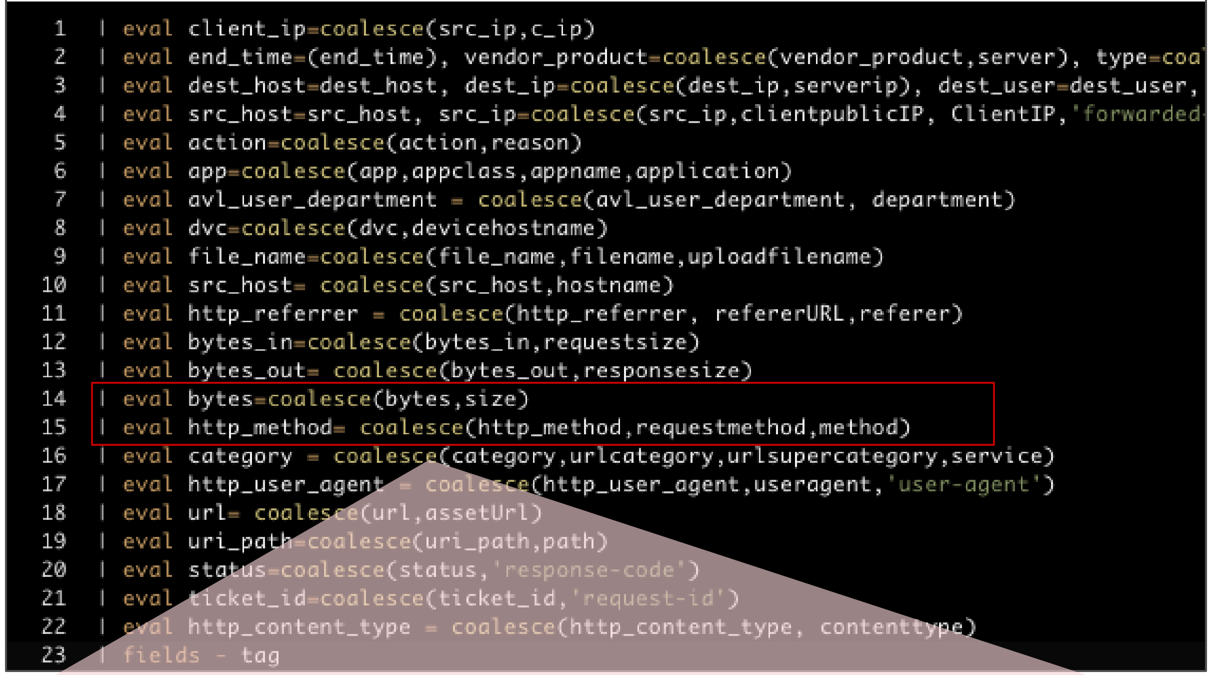
Include all name variations in search-time

- ✓ Avoid breaking parser functions
- ✗ Increase search time overhead

Search-time normalization is not the most efficient method but practical

Search-time normalization example

- Always prepend normalization queries at **search time**, using coalesce() function, to unify names



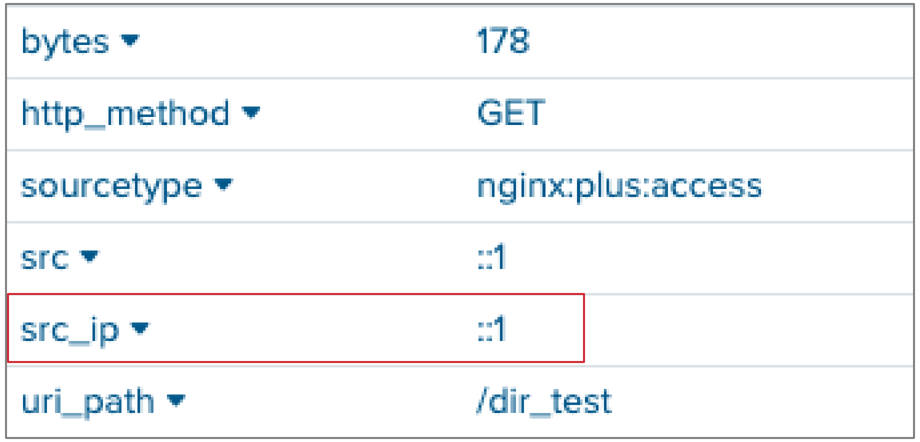
```
| eval client_ip=coalesce(src_ip,c_ip)
| eval end_time=(end_time), vendor_product=coalesce(vendor_product,server), type=coa
| eval dest_host=dest_host, dest_ip=coalesce(dest_ip,serverip), dest_user=dest_user,
| eval src_host=src_host, src_ip=coalesce(src_ip,clientpublicIP, ClientIP,'forwarded
| eval action=coalesce(action,reason)
| eval app=coalesce(app,appclass,appname,application)
| eval avl_user_department = coalesce(avl_user_department, department)
| eval dvc=coalesce(dvc,devicehostname)
| eval file_name=coalesce(file_name,filename,uploadfilename)
| eval src_host= coalesce(src_host,hostname)
| eval http_referrer = coalesce(http_referrer, refererURL,referer)
| eval bytes_in=coalesce(bytes_in,requestsize)
| eval bytes_out= coalesce(bytes_out,responsesize)
| eval bytes=coalesce(bytes,size)
| eval http_method= coalesce(http_method,requestmethod,method)
| eval category = coalesce(category,urlcategory,urlsupercategory,service)
| eval http_user_agent = coalesce(http_user_agent,useragent,'user-agent')
| eval url= coalesce(url,assetUrl)
| eval uri_path=coalesce(uri_path,path)
| eval status=coalesce(status,'response-code')
| eval ticket_id=coalesce(ticket_id,'request-id')
| eval http_content_type = coalesce(http_content_type, contenttype)
| fields - tag
```

```
| eval bytes=coalesce(bytes,size)
| eval http_method= coalesce(http_method,requestmethod,method)
```

Although both Nginx & Apache logs are using latest parsing add-on from Splunk, their naming schemes still differ slightly on source IP address

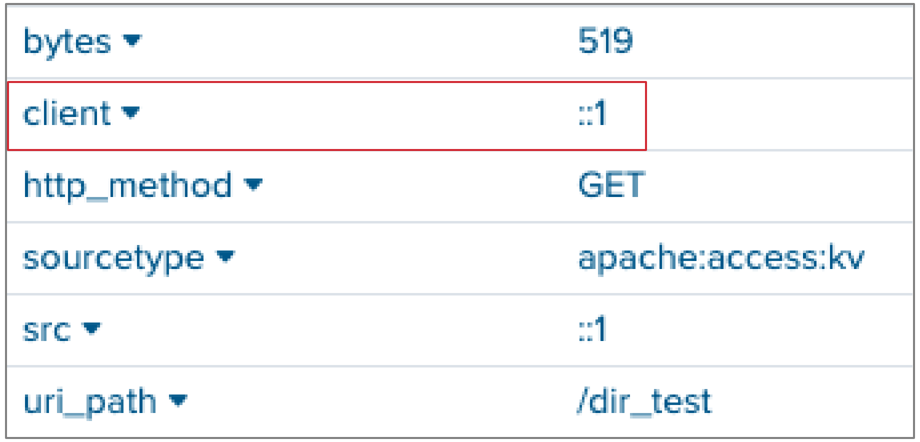
	Bytes transferred	HTTP method	Source IP	Source	URI Path
Nginx naming scheme	bytes	http_method	src_ip	src	uri_path
Apache naming scheme	bytes	http_method	client	src	uri_path

Nginx access log parsing result



bytes ▾	178
http_method ▾	GET
sourcetype ▾	nginx:plus:access
src ▾	::1
src_ip ▾	::1
uri_path ▾	/dir_test

Apache access log parsing result



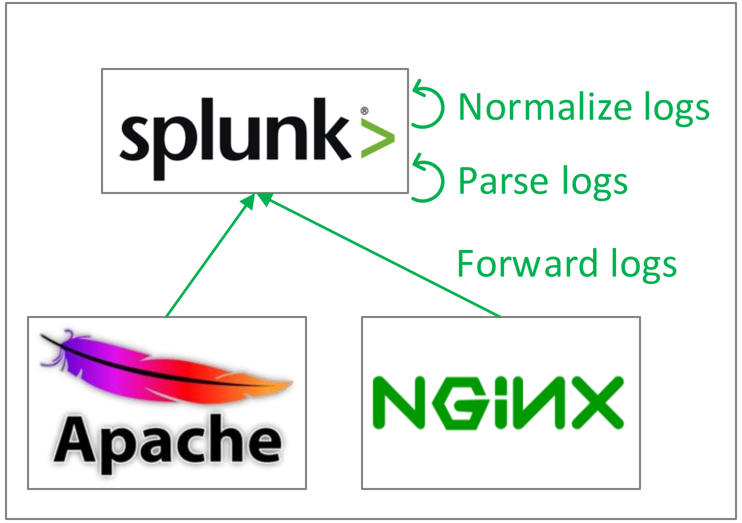
bytes ▾	519
client ▾	::1
http_method ▾	GET
sourcetype ▾	apache:access:kv
src ▾	::1
uri_path ▾	/dir_test

Luckily, there is common field ‘src’ represents source IP. We don’t need to use coalesce() to normalize src_ip & client. We can consider normalization done in our Nginx & Apache logs

After logs forwarding, parsing and normalization are completed. We can try out detection on directory brute-force attacks

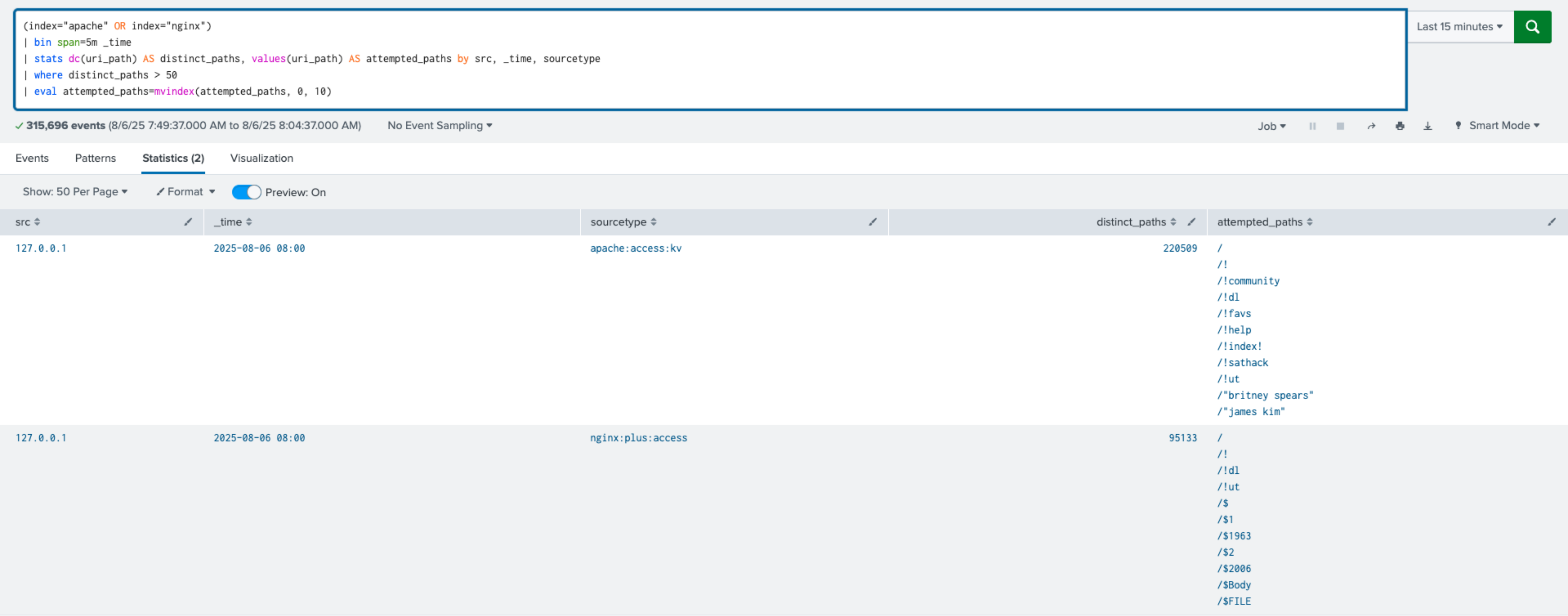
	Attack scenario	▪ **Directory brute force:** Attacker throws wordlist onto Web's URL path to identify available resource paths
	Detection logic	▪ High-volume of different URL path visited by single source IP in short amount of time
	Log sources	▪ **Nginx access logs:** /var/log/nginx/access.log ▪ **Apache access logs:** /var/log/apache2/access.log
Done	**Forward logs to SIEM (Splunk)**	▪ Configure Rsyslog to forward logs into SIEM (Splunk)
Done	**Parse logs**	▪ Ensure fields are extracted (Source IP, URL Path, Timestamp)
Done	**Normalize logs**	▪ Ensure field names between Nginx & Apache logs are the same, so we can reuse the detection rule on both application

Example environment



Security Research Labs

Detection rule to catch directory brute force, and record the paths (wordlist) attacker used



```
(index="apache" OR index="nginx")
| bin span=5m _time
| stats dc(uri_path) AS distinct_paths, values(uri_path) AS attempted_paths by src, _time, sourcetype
| where distinct_paths > 50
| eval attempted_paths=mvindex(attempted_paths, 0, 10)
```

✓ 315,696 events (8/6/25 7:49:37.000 AM to 8/6/25 8:04:37.000 AM)

src	_time	sourcetype	distinct_paths	attempted_paths
127.0.0.1	2025-08-06 08:00	apache:access:kv	220509	/ /! /!community /!dl /!favs /!help /!index! /!sathack /!ut /"britney spears" /"james kim"
127.0.0.1	2025-08-06 08:00	nginx:plus:access	95133	/ /! /!dl /!ut /$ /$1 /$1963 /$2 /$2006 /$Body /$FILE

Agenda

- Introduction of SOC
- Architecture overview
- Syslog forwarding
- Parsing
- Normalization
- **High-fidelity starter alerts**
- Recommendation of log sources

Honeypot/-token mimics vulnerability to deceive attackers and raise early alerts when triggered

Honeypot/-token	System	Alert conditions	Related-attacks
Fake database table	▪ Databases	▪ Someone accessed the fake table	▪ SQL injection ▪ Stolen DB credentials
Fake API keys	▪ AWS, GCP, Azure, Kubernetes	▪ The API key is found used	▪ Stolen credentials
Fake document	▪ Any file system ▪ Email, message records	▪ A DNS callback triggered by opening the document	▪ File system enumeration ▪ Data leakage
Dummy server	▪ Network	▪ Someone scan its ports ▪ Someone visited a URL of it	▪ Port scanning/ sweeping
Kerberoastable account	▪ Active Directory	▪ Someone request a Kerberos ticket of the account	▪ Kerberoasting for offline password cracking
Tripwire AD account	▪ Active Directory	▪ Someone query details of the account through LDAP	▪ Bloodhound ▪ AD enumeration
Decoy ADCS template (ESC1[1])	▪ Active Directory Certificate Service	▪ Someone request a certificate with the decoy template	▪ ADCS attacks, ESCs

[1]: ESC1 – https://specterops.io/wp-content/uploads/sites/3/2022/06/Certified_Pre-Owned.pdf

Deep dive: Intended Kerberoastable account detects Kerberoasting attempts

What is Kerberoasting?

- **Attack pre-requisite:** Any valid AD account; Target account has SPN[1] attribute set
- **Approach:** Request service ticket of target account via Kerberos protocol
- **Goal:** Crack service ticket’s encryption key offline, as it is encrypted with target user’s password

Detection challenge

- **Noise:** Difficult to differentiate malicious intent in service ticket requests in the network
- **Easy to evade:** Attackers can request service tickets slowly to avoid huge volume of logs

Kerberoastable account setup

1. Create dummy user account with SPN set
2. Configure account’s Kerberos encryption algorithm as RC4 to raise attacker’s interest
3. Monitor event ID 4769[2] & 4770[3] on the dummy user
   - Any service ticket request on the account is high-confidence alert of Kerberoasting attempt
4. Check account name & source IP address in log to identify the compromised account & host

[1]: SPN – Service Principal Name; [2]: Event ID 4769 – A Kerberos service ticket was requested; [3]: Event ID 4770 – A Kerberos service ticket was renewed;

Deep dive: Tripwire AD account detects bloodhound or AD enumeration

What is bloodhound/ AD enumeration?

- **Attack pre-requisite:** Any valid AD account
- **Approach:** Make LDAP queries to learn permissions, attributes, relationship of AD objects
- **Goal:** Identify misconfigured permissions in AD for lateral movement

Detection challenge

- **Noise:** Difficult to differentiate which LDAP requests were with enumeration intent
- **Easy to evade:** Attackers can reduce LDAP requests volume by separating enumerations on users, groups, ACL, etc, and even apply jitter

Tripwire account setup

1. Create dummy user account
2. Enable auditing on “read all properties” actions on the dummy user account
3. Monitor event ID 4662[1] on the dummy user (use account’s GUID instead of username)
   - Any read action on the dummy user account properties is high-confidence alert
4. Check account name in log to identify the compromised account
   - Source IP is not directly available, but we can identify it by reviewing login history

[1] Event ID 4662 – An operation was performed on an object

Deep dive: Decoy ESC1 ADCS template can detects ADCS hacking attempts

What is ADCS ESC1[1] abuse

- **Attack pre-requisite:** Any valid AD account; ADCS template that allows user supply SAN[2]
- **Approach:** Supply high-privileged username in SAN during certificate request
- **Goal:** Login as arbitrary user leveraging the certificate's SAN value

Detection challenge

- **Noise:** Difficult to differentiate between legitimate & malicious certificate requests

Fake ESC1 vulnerable template

1. Configure an ADCS template with ESC1 vulnerability
2. Install and configure TameMyCerts[3] plugin to prevent issuance if CSR contains SAN
3. Enable extended audit logs in ADCS & TameMyCerts
4. Monitor event ID 4886[4] and TameMyCerts event ID 6[5]
   - CSR denied triggered is a high-confidence alert of ESC1 exploitation attempt
5. Check account name & source IP address in log to identify the compromised account & host

Check our Certiception[6] for more details

1: ESC1 – https://specterops.io/wp-content/uploads/sites/3/2022/06/Certified_Pre-Owned.pdf; 2: SAN – Subject Alternative Name;
3: TameMyCerts – https://github.com/Sleepw4lker/TameMyCerts; 4: Event ID 4886 – Certificate enrollment requested;
5: TameMyCerts event ID 6 – CSR denied due to policy violation; 6: Certiception – https://github.com/srlabs/Certiception/tree/main

Agenda

- Introduction of SOC
- Architecture overview
- Syslog forwarding
- Parsing
- Normalization
- High-fidelity starter alerts
- **Recommendation of log sources**

Overview of recommended log sources

System	Log name	Common log path(s)	Logging scope
Linux OS	Authentication logs	▪ /var/log/secure ▪ /var/log/auth.log	Authentication usage, including sudo, su, ssh
Linux OS	Auditd logs	▪ /var/log/audit/audit.log	Command execution, SYSCALL usage, file's read & write & execute history
Windows OS	Security logs	▪ Event Viewer > Windows Logs > Security	Local authentication history
Windows OS	PowerShell logs	▪ Event Viewer > Applications and Services Logs > Microsoft > Windows > PowerShell > Operational	PowerShell execution. Verbosity depends if Script Block Logging and Module Logging are enabled
Active Directory	Security logs	▪ Event Viewer > Windows Logs > Security (on Domain Controller)	Domain-wide authentication and directory service access history. Logging verbosity depends on GPO settings
Web application	Access logs	▪ /var/log/nginx/access.log ▪ /var/log/squid/access.log ▪ etc	Source IP, user agent accessed what URI of application

References

- **Elastic Common Schema**
  https://www.elastic.co/docs/reference/ecs
- **Splunk Common Information Model**
  https://help.splunk.com/en/splunk-enterprise/common-information-model/6.0/introduction/overview-of-the-splunk-common-information-model
- **RFC 3389 vs ISO 8601**
  https://ijmacd.github.io/rfc3339-iso8601/
- **How to hear the Bloodhound barking**
  https://medium.com/mercadonait/how-to-hear-the-bloodhound-barking-5ac290427b17
- **TameMyCerts**
  https://github.com/Sleepw4lker/TameMyCerts
- **Certiception**
  https://github.com/srlabs/Certiception/blob/main/documentation/The_Red_Teamers_Guide_To_Deception.pdf

Thank you!
Any questions?

Please feel free to reach out offline for more in-depth discussion for both Red and Blue Team operations!

Kristen Huang
Security Consultant, SRLabs
kristen@srlabs.hk

LinkedIn: