Syslog Export

27 October 2023

This document describes the capabilities and formatting of the syslog export features built into our Threater Enforce software. This document applies to software **builds 224** and later. If you are on an older build, please update your Threater Enforce software first before continuing.

Earlier builds may not match the descriptions in this document, so it is important to update your software to ensure compatibility. It is possible that later builds will still match the descriptions in this document, but if you have a later build, you should consult the software release notes which can be viewed from the Threater portal.

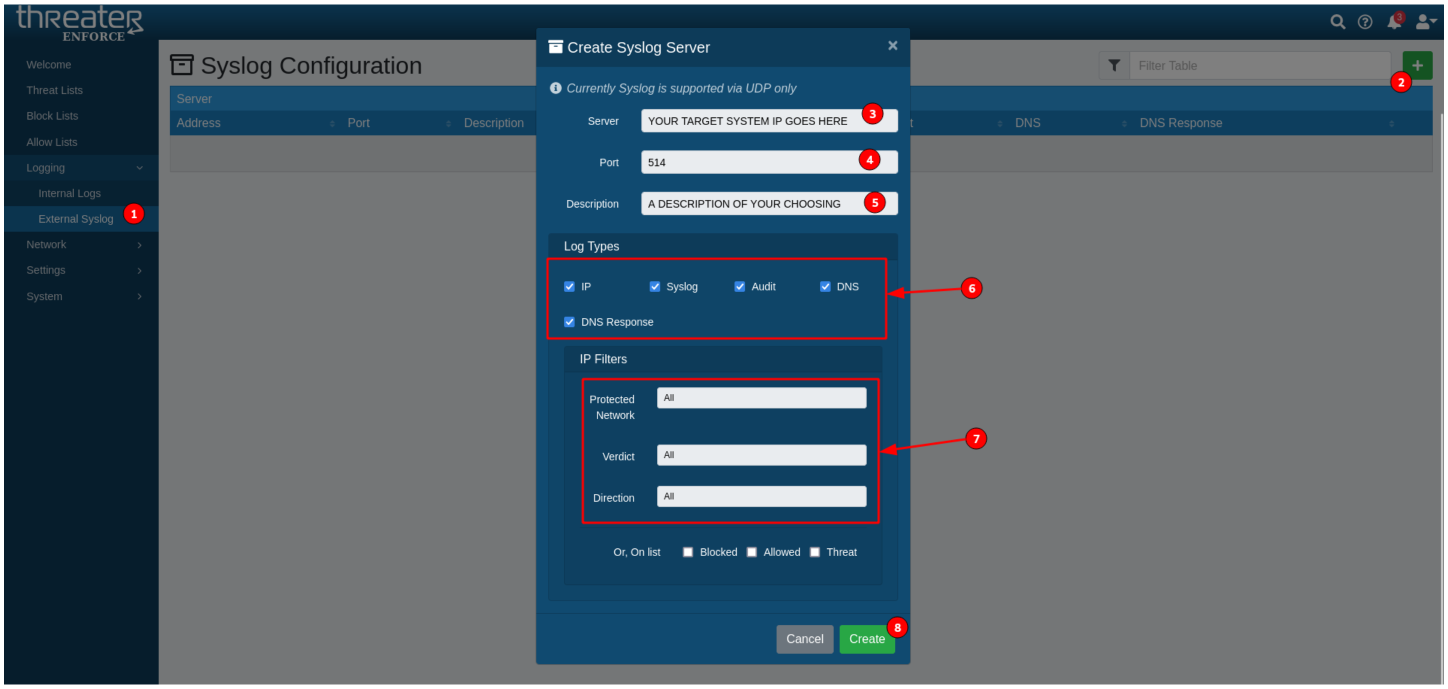
Why Syslog Exports?

Syslog exports are an industry-standard and time-proven way of exporting data in a concise, standards-based manner. **Our syslog export format is compliant to RFC-5424.** This ensures seamless integration alongside any number of external tools, including popular security information and event management (SIEM) tools, such as Splunk and IBM QRadar, as well as popular data analytics tools like Gravwell, and even full open-source tools like syslog-ng.

As such, this document is not written with any particular SIEM tool in mind. Instead, this document is meant to focus on the comprehensive data contained in our syslog exports, enabling our logs to be parsed in an intelligent way by any tool that can ingest RFC-compliant syslog exports.

Configuring the Syslog Export Features

To configure the syslog export features of your Threater Enforce software installation, the general ordered flow is shown in the following image:



Currently Syslog is supported via UDP only

Server	YOUR TARGET SYSTEM IP GOES HERE
Port	514
Description	A DESCRIPTION OF YOUR CHOOSING

Log Types

☑ IP ☑ Syslog ☑ Audit ☑ DNS

☑ DNS Response

IP Filters

Protected Network	All
Verdict	All
Direction	All

Or, On list ☐ Blocked ☐ Allowed ☐ Threat

After logging into your Threater Enforce's secure web interface, select `Logging > External Syslog` (step 1). Click the green/white plus sign in the top right (step 2) to add a new syslog server. You can add as many of these as you want - you aren't limited to just one. This can be useful if you purposely want to send different log information to different business systems.

Choose the IP address (step 3) of your target system, and generally, you'll use 514 for the UDP port (step 4), which is the typical listening port most systems use for syslog ingest. Feel free to change it if you need to. You can optionally add a description for your target system if you'd like (step 5). This can be very useful if you are sending logs to more than one system so that you can more easily know which is which.

You can then select the logs that you want to send - either a subset or all of them (steps 6 and 7). In the example above, we have configured things to send "everything" by enabling the `IP`, `Syslog`, `Audit`, `DNS`, and `DNS Response` checkboxes, and for the `IP Filters`, we've set the `Protected Network`, `Verdict`, and `Direction` to `All`. In a subsequent section, we'll describe the various configuration options in significantly more detail.

When you've configured it to your liking, click create (step 8), and at that point, the software will immediately start sending the logs you've chosen as they are generated from that point forward in real-time, to the system of your choosing.

Multiple Targets

The software supports as many external syslog targets as you'd like. That is, you can selectively filter the logs of your choosing and send them to one particular SIEM or business tool, and send the same or different filtered configuration to a different SIEM or business tool. If you do this, we recommend that you include descriptions in your configuration so that you can more easily identify them downstream.

Using multiple targets can be quite useful in situations where you need to be concerned with data ingest rates or cost implications of data ingest rates on various target systems. This capability is also useful when you have multiple external critical business systems that need live, real-time access to the robust, standards-compliant logs.

RFC-5424 Compliant Headers

All of our export logs are RFC-5424 compliant. We use an intelligent comma-separated key=value scheme after the RFC-compliant header, to simplify ingesting and parsing by target SIEM and business systems.

Our log messages are compliant to RFC-5424 and resemble:

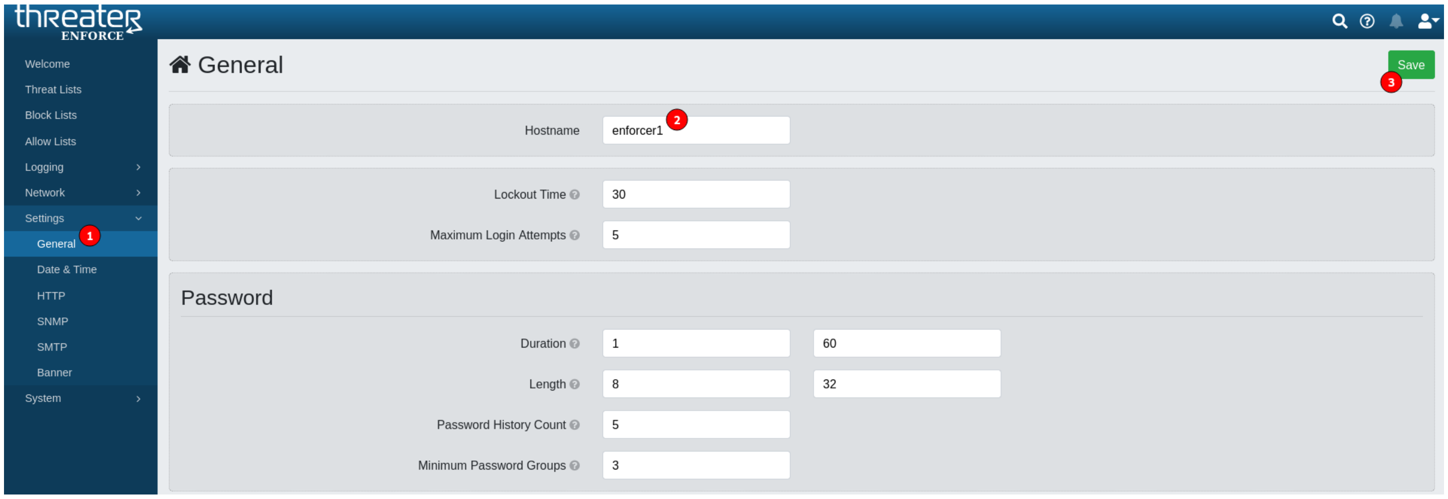
```
<nn>V YYYY-MM-DDTHH:MM:SS.sssZ hostname bctifw - type_log - data
```

The leading `<nn>` is defined in RFC-5424 and represents the priority value. Generally, many syslog parsers will automatically parse and store this field, even though its value is limited in our application given the intelligent way we've done our data modeling, which we'll discuss in more detail soon.

The immediately following `V` is a version number. Currently, all of our logs are built-out as version 1, with our official version 1 being first released with software build 48. Previous releases all used version 1 for purposes of RFC compliance, but note that build 48 is the first officially supported version 1 and may differ from prior releases. Subsequent releases (post build 48) will maintain adherence to version compatibility. This means that the API will remain backwards compatible, however, this does not necessarily mean that a version bump will occur when a new feature is added, instead, only when an existing feature is modified or deprecated.

The UTC timestamp format shown is standardized and well-known; most SIEM and related business tools are able to natively parse it. A populated actual value might be: `2020-08-14T14:37:11.650Z`.

The **`hostname`** quantity will be replaced by the hostname of the system running the Threater Enforce software. Note that setting a unique hostname on each of your Threater Enforce installations will allow for simple filtering and correlation across systems. To change the hostname of a Threat Enforce instance, navigate to `Settings > General`, like this:



General

Hostname: enforcer1

Lockout Time: 30

Maximum Login Attempts: 5

Password

Duration: 1 | 60

Length: 8 | 32

Password History Count: 5

Minimum Password Groups: 3

The value **`bctifw`** is a static value that we use to uniquely identify our logs in the APP-NAME field of the RFC 5424 header. Using that field as part of any specific query might simplify the query and improve overall performance as a top-level filter.

The **`type_log`** quantity provides a simple text filter that can be used in your target system to know that you are operating on a particular log type. Currently, the five possible values are `packet_log`, `dns_log`, `dns_resp_log`, `system_log` and `audit_log`.

And last but not least, the trailing comma-separated key=value data is dependent on the `type_log` quantity. The various formats and log configurations are described in detail in the remainder of this document.

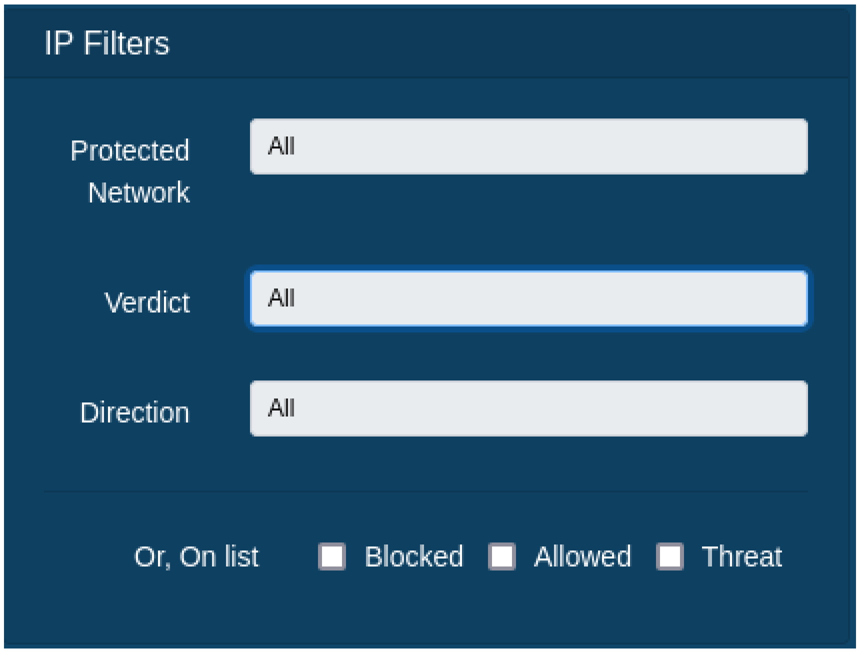
Packet Logs - Configuration

The packet log exports are generally the logs that you will be most interested in keeping track of. They track all connections and pseudo-connection traffic. They provide a simple, centralized way to get ground-truth on whether a connection was allowed or denied, including why those decisions were made, with full attribution to the various supported geo-mapping, ASNs, threat lists, denied lists, allowed lists, and so on, in comprehensive fashion.

Regardless of whether a connection is tagged as allowed or denied, we always perform a full security analysis with the cyber intelligence information made available from the Threater portal. That means that even when a connection might be allowed because its IP address was manually placed on an allowed list, the syslog entry will still contain detailed attribution information about the country, ASN, and its presence on any or all of the configured lists at the time the decision was made.

That capability provides a fantastic way to do security triage in true real time, across any number of Threater Enforce installations, by configuring them to send their syslog export data to suitable external systems of your choosing, whether those are on-premise systems or cloud-based systems.

The IP packet logs can get quite chatty, and are responsible for the vast majority of the generated data. Because of this, we've added powerful filtering features to the packet log export configurator as briefly described in the previous section. We'll talk a bit more about them below, in conjunction with corresponding imagery:


IP Filters

Protected Network: All

Verdict: All

Direction: All

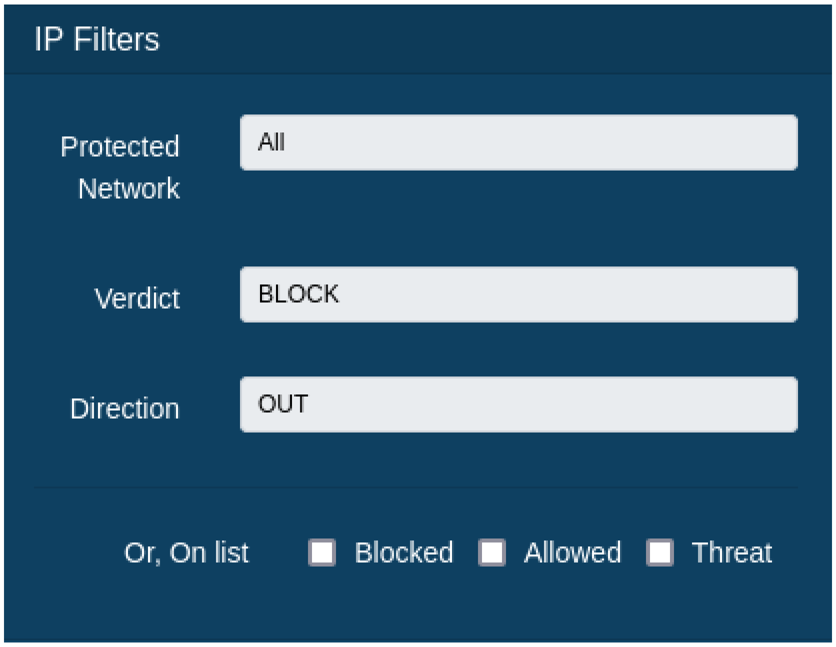
Or, On list ☐ Blocked ☐ Allowed ☐ Threat

`Protected Network` refers to either the collection of `All` of the protected networks on a given Threater Enforce installation, or you can isolate them. For example, if you have an `OUTBOUND` protected network configured and you want to send just the IP connection logs associated to that `OUTBOUND` to a particular SIEM, you can click in the `Protected Network` field and select that entry from a dropdown that will appear.

`Verdict` selects whether you want to export allowed (`ALLOW`) connections, blocked (`BLOCK`) connections, or both (`All`) and is also selectable via a simple dropdown.

Direction selects the connection direction that you wish to export, which is either inbound (`IN`) , outbound (`OUT`), or both (`All`), selectable by its own dropdown.

The collection of those three things - `Protected Network`, `Verdict`, and `Direction` - are combined with boolean AND logic. That is, in the following example, only outbound connections that have been blocked from any protected network policy would be exported:


IP Filters

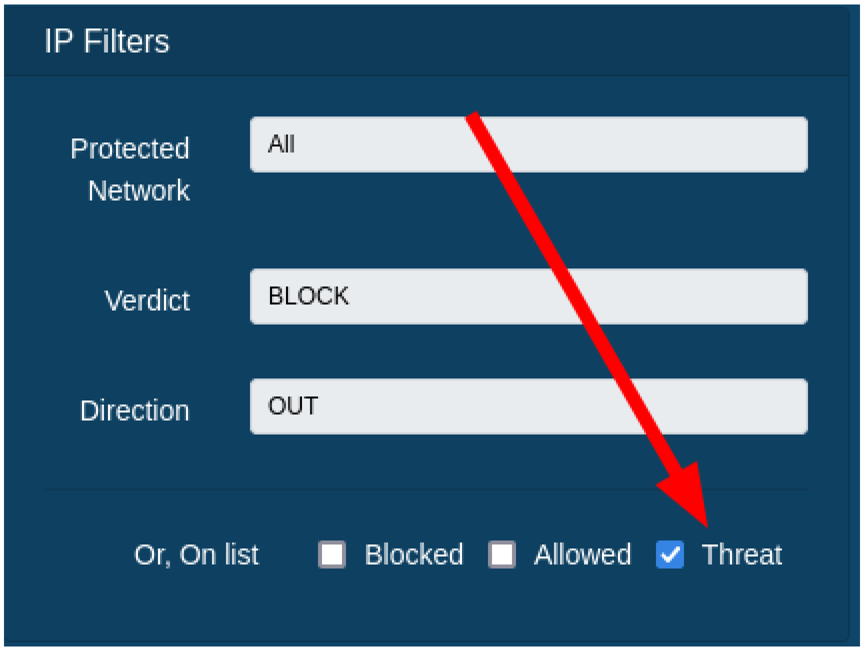
Protected Network: All

Verdict: BLOCK

Direction: OUT

Or, On list ☐ Blocked ☐ Allowed ☐ Threat

But let's say that, in addition to exporting information about your policy-specific outbound blocks, you also wanted to log, to that same target system (for, perhaps, correlation and cybersecurity analysis), any connection that happened to also be registered on a threat list, irrespective of whether or not it was filtered by the AND logic that we just configured? We can achieve that by leveraging the second layer of filters which represent a boolean logic OR section:


IP Filters

Protected Network: All

Verdict: BLOCK

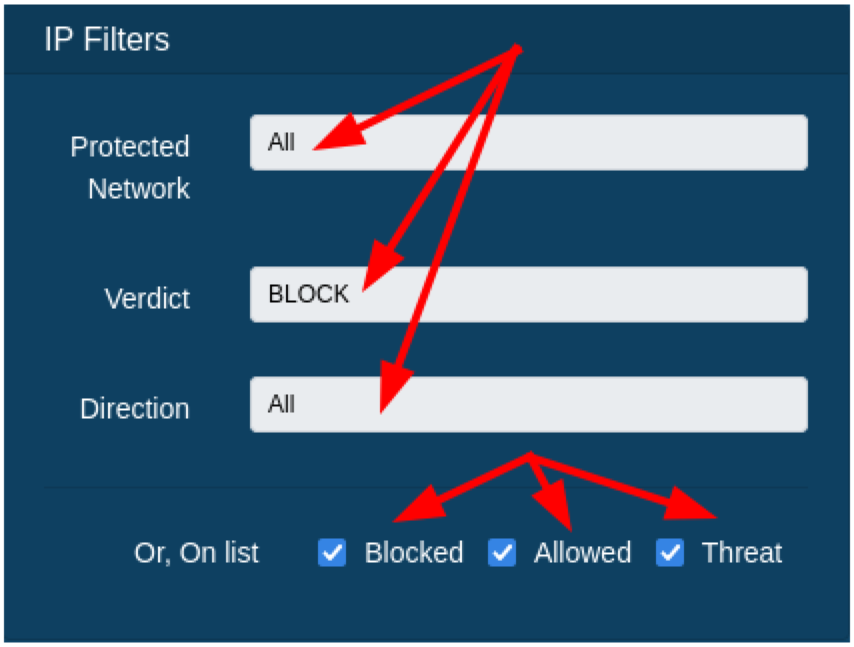
Direction: OUT

Or, On list ☐ Blocked ☐ Allowed ☑ Threat

Checking one or more of the `Blocked`, `Allowed`, and `Threat` list checkboxes in that boolean logic section will result in any IP log that is associated with the selected list types to be exported as well, independent of the boolean AND clause from the first section. Note that a log is exported only once, so if a particular log matches both sections it will still only appear in your target system one time.

For scenarios where it isn't feasible to send everything to a given SIEM, this layered scheme provides a best-practices approach to maximizing the use of a target SIEM system that may be limited by its ingest rate and/or cost considerations.

As a final example, here is one typical configuration we often see, where any blocks (independent of protected network and direction) are conveyed, plus anything that was allowed that happened to be associated to one or more of the list types:


IP Filters

Protected Network: All

Verdict: BLOCK

Direction: All

Or, On list: ✓ Blocked ✓ Allowed ✓ Threat

Effectively, in that example, everything is exported except for standard allowed traffic not appearing on any list.

Packet Logs - Format Descriptions

Following the RFC-5424 compliant header, the data section of our packet logs contain rich information about both allowed and denied connections and pseudo-connections. Before we describe the detailed data formats, we'll look at a few real-life examples.

Here's a log that details that Threater Enforce allowed a UDP request to Google's well known secondary DNS server at 8.8.4.4 on port 53. Note that the list scenario here has been contrived to show all possible allowed and denied list variants for both active and inactive lists (but no threat list variants):

```
<30>1 2020-08-18T00:51:59.294Z myenforcer bctifw - packet_log -
action=allowed, direction=outbound, group="Local Office Outbound",
proto=UDP, country="UNITED STATES", as_num=15169, as_name="Google Inc.",
reason=allowedlist, src=10.0.0.95, dst=8.8.4.4, src_port=39113,
dst_port=53, dl_active="Manual Denied List", dl_inactive="Manual Denied
List Inactive", al_active="Local Office", al_inactive="Local Office
Inactive,Curated DNS"
```

Here's a separate example of a log that was denied because it was on a Proofpoint threat list, categorized as a known-malicious proxy server, and its threat score of 92 was above the associated policy's configured threshold of 90, which resulted in the connection being blocked. It did not show up on any other list:

```
<29>1 2020-08-18T00:57:12.401Z myenforcer bctifw - packet_log -
action=denied, direction=outbound, group="Local Office Outbound",
proto=TCP, country="NETHERLANDS", as_num=14061, as_name="Digital Ocean,
Inc.", reason=threatlist, src=10.0.0.95, dst=198.211.120.59,
src_port=58802, dst_port=443, flags="SYN", tl="Proofpoint",
tl_category="PROXY", tl_threshold="90", tl_score="92"
```

Here's a complex example showing multiple lists detailed in double-quoted comma separated value sets. This one is a really, really bad IP out of China, and it has a further interesting characteristic in that its ASN name contains a misspelled word in their official ASN registry!

```
<29>1 2021-11-11T15:50:02.006Z myenforcer bctifw - packet_log -
action=denied, direction=outbound, group="Outbound", proto=ICMP,
country="CHINA", as_num=56048, as_name="China Mobile Communicaitons
Corporation", reason=deniedlist, src=192.168.0.95, dst=223.71.167.165,
tl="Webroot,Proofpoint", tl_category="SCANNER", tl_threshold="90",
tl_score="92", dl_active="ET Compromised IPs,State of Missouri
SOC,Blocklist.de,CINS Army list"
```

Note that fields that have no bearing on a particular connection are not populated. For example, if no denied lists are associated with an entry, then keys relating to denied lists are not created for that entry. Put another way, parsers should not expect any given key to be populated. Instead, if a key is not present, it simply did not apply for that particular record. This keeps the data transfer sizes minimized, which is important since many SIEMs charge based on the amount of data ingested, or related factors.

It is critical that your parsing engines operate based on the named values and **not** their positions, as key=value position order is not guaranteed from software release to release.

The following table describes the various packet log key value pair constructs.

packet_log Key	**Description**
action	This represents the connection/packet verdict. If this reads `allowed`, the connection was allowed, and if it reads `denied`, it will have been blocked.
direction	This represents the determined direction of the initial connection (or pseudo-connection, in the case of connectionless protocols such as UDP). The value will be either `inbound` or `outbound`. Note that this is the direction of the connection and not necessarily the packet orientation. That is, a connection is outbound if a protected IP initiates the conversation. It is inbound if a protected IP is the target of an external initiator. It's similar to a phone call: although I might call you on the phone, meaning that it's an outbound connection from my perspective. During that connection, there are both inbound and outbound packets, regardless of the established and reported connection direction, which is always set for the duration of that particular conversation.
lsh	This represents whether or not loose state handling was employed in determining the connection direction. Generally, if we see a pure SYN packet, we can definitively determine the connection direction. If, on the other hand, we do not see the originating SYN packet, we use a loose packet handling scheme to predict the packet direction. This can happen when a unit is first installed, or if it is rebooted. In the event that we used such a scheme, this key will appear in the log and will be set to `true`.
group	This is the protected network responsible for the verdict. It is enclosed in double quotes.
proto	This is the identified protocol as determined from the packet's header information. Most often it is `TCP` or `UDP`, but the full gamut of layer 2 and layer 3 protocols are observed by Threater Enforce. Generally, all layer 2 traffic that does not encapsulate an IP packet is always bridged through, whereas the IP traffic is evaluated against our always-on real-time country, ASN, and threat/denied/allowed list intelligence.
country	This is the country attributable to the external endpoint IP address. This information comes from our always-up-to-date geolocation intelligence for all IPs across the globe. It is enclosed in double quotes.
as_num	This is the unique asynchronous subscriber number (ASN) id as determined

	from our always-up-to-date ASN intelligence for all networks across the globe. Normally this would only be used to cross-index into a separate SIEM database for advanced reporting. It is generally ignored by a parser in favor of human-readable `as_name`, as noted below.
`as_name`	This is the ASN's physical name. An example might be "Google Inc." or "CloudFlare, Inc." or "Orange S.A." and so on. It is enclosed in double quotes.
`reason`	The reason key provides the primary reason for the verdict. Values can be `allowedlist`, `asn`, `country`, `deniedlist`, `policy`, and `threatlist`. For details about the decision making process, the reader is referred to the Threater Enforce flowchart in the separate document *Threater - Policy Enforcement* and our user manuals.
`src`	This is the source IP address from the perspective of the connection orientation. For example, if a protected IP initiates an outbound connection to a Google server, then the protected IP is the source IP. If, on the other hand, an external IP initiates a connection to a protected IP, then the connection is inbound, and the source IP is the external IP.
`dst`	This is the destination IP address from the perspective of the connection orientation. For example, if a protected IP initiates an outbound connection to a Google server, then the Google server is the destination. If, on the other hand, an external IP initiates a connection to a protected IP, then the connection is inbound, and the destination IP is the protected IP.
`src_port`	This is the source port associated with the source IP of a TCP connection or UDP pseudo-connection.
`dst_port`	This is the destination port associated with the destination IP of a TCP connection or UDP pseudo-connection.
`flags`	For TCP connections, this key lists the associated TCP flags for the particular connection/packet used for evaluation. It is always enclosed in double quotes (since multiple comma-separated TCP flags can be associated to one log, such as perhaps "RST,ACK").
`tl`	Displays, as a double quoted comma separated value list, any and all threat lists associated with the IP address(es) associated with the connection.
`tl_category`	This lists all associated threat list categories for the packet as a double quoted comma-separated list. It is in lockstep-order to the values in the tl_threshold key.

`tl_score`	This lists all associated actual scores as a double quoted comma-separated list, where the order is identical to the ordering of the tl_category key for simple parsing correlation. Note that it is common for the scores to be the same for multiple category entries.
`tl_threshold`	This lists all associated threshold scores as a double quoted comma-separated list, where the order is identical to the ordering of the tl_category key for simple parsing correlation.
`dl_active`	Displays, as a double quoted comma separated value list, any and all active (enabled) denied lists associated with the protected network and policy responsible for the verdict.
`dl_inactive`	Displays, as a double quoted comma separated value list, any and all inactive (disabled) denied lists associated with the protected network and policy responsible for the verdict.
`al_active`	Displays, as a double quoted comma separated value list, any and all active (enabled) allowed lists associated with the protected network and policy responsible for the verdict.
`al_inactive`	Displays, as a double quoted comma separated value list, any and all inactive (disabled) allowed lists associated with the protected network and policy responsible for the verdict.

DNS Logs - Format Descriptions

Following the RFC-5424 compliant header, the data section of our DNS logs contain rich information about outbound unencrypted DNS requests and whether they were allowed or blocked with respect to UDP port 53 activity.

Here's an example unencrypted DNS request made to an internal gateway, attempting to resolve www.google.com, which was allowed:

```
<30>1 2020-08-18T00:53:17.431Z myenforcer bctifw - dns_log -
action=allowed, proto=UDP, reason=policy, src=10.0.0.95, dst=10.0.0.1,
src_port=43030, dst_port=53, domain=www.google.com
```

Here's another example of an unencrypted DNS request against Google's primary DNS server 8.8.8.8 also attempting to resolve google.com that was also allowed:

```
<30>1 2020-08-18T00:28:53.648Z myenforcer bctifw - dns_log -
action=allowed, proto=UDP, reason=policy, src=10.0.0.95, dst=8.8.8.8,
src_port=43152, dst_port=53, domain=google.com
```

In this next example, an unencrypted DNS request that was attempted against Google's primary DNS server 8.8.8.8, but the request was intercepted and denied by Threater Enforce since the domain requested was on a manually crafted denied list:

```
<29>1 2020-08-18T00:28:53.959Z myenforcer bctifw - dns_log -
action=denied, proto=UDP, reason=deniedlist, src=10.0.0.95, dst=8.8.8.8,
src_port=32975, dst_port=53, domain=amazonfoo.cn, dl_active="CN amazon"
```

Now we'll look at an example of an unencrypted DNS request to resolve threater.com, which was on an allowed domain list that was enabled and also a separate allowed list that had not been enabled (and so it reports them individually as being present on both of the lists):

```
<30>1 2020-08-18T00:28:54.119Z myenforcer bctifw - dns_log -
action=allowed, proto=UDP, reason=allowedlist, src=10.0.0.95, dst=8.8.8.8,
src_port=54256, dst_port=53, domain=threater.com, al_active="Threater
Corporate", al_inactive="Threater Corporate Inactive"
```

Here's one final example of another domain lookup that was denied; this one relates to a fake coronavirus site that has been associated to cybercrime as populated by multiple world-class DomainTools denied lists:

```
<29>1 2020-08-18T00:28:54.284Z myenforcer bctifw - dns_log -
action=denied, proto=UDP, reason=deniedlist, src=10.0.0.95, dst=8.8.8.8,
src_port=37834, dst_port=53,
domain=100bellscoronaviruspandemiclockdown.golf, dl_active="COVID-19-
DomainTools- 70,DomainTools,COVID-19- DomainTools- 99"
```

Note that fields that have no bearing on a particular connection are not populated. For example, if no denied lists are associated with an entry, then keys relating to denied lists are not created for that entry. Put another way, parsers should not expect any given key to be populated. Instead, if a key is not present, it simply did not apply for that particular record. This keeps the data transfer sizes minimized, which is important since many SIEMs charge based on the amount of data ingested, or related factors.

It is critical that your parsing engines operate based on the named values and **not** their positions, as key=value position order is not guaranteed from software release to release.

The following table describes the various packet log key value pair constructs.

`dns_log Key`	**Description**
`action`	This represents the verdict. If this reads `allowed`, the outbound unencrypted DNS request was allowed, and if it is `denied`, it will have been blocked. A blocked DNS request will result in the requester not receiving a mapped numeric IP address for the requested domain.
`proto`	This is the identified protocol as determined from the packet's header information. Currently, this value will always show as `UDP`, since we only analyze DNS traffic riding over UDP on port 53.
`reason`	The reason key provides the primary reason for the verdict. Values can be `allowedlist`, `deniedlist`, or `policy`.
`src`	This is the source (requesting) protected IP address from the perspective of the unencrypted DNS request.
`dst`	This is the external destination IP address from the perspective of the unencrypted DNS request. For example, when a protected IP initiates an unencrypted DNS request to a Google DNS server, then the Google DNS server is the destination.
`src_port`	This is the source port, from the perspective of the local protected entity that generated the request. For most modern equipment, this will generally be a high-numbered port.
`dst_port`	This is the destination port, from the perspective of the local protected entity that generated the request. Thus, it will always show as port 53, since we monitor DNS activity only on outbound UDP port 53.
`domain`	The domain requested for resolution as part of the unencrypted DNS request.
`dl_active`	Displays, as a double quoted comma separated value list, any and all active (enabled) denied domain lists.
`dl_inactive`	Displays, as a double quoted comma separated value list, any and all inactive (disabled) denied domain lists.
`al_active`	Displays, as a double quoted comma separated value list, any and all active (enabled) allowed domain lists.
`al_inactive`	Displays, as a double quoted comma separated value list, any and all inactive (disabled) allowed domain lists.

DNS Response Logs - Format Descriptions

Our DNS Response logs were introduced in Build 75. Unlike our other logs that are viewable directly in our Threater Enforce software and can be exported, our new DNS Response logs are built to be exported only and do not have an associated UI view. That is, to view our DNS Response logs, you **must** use the syslog export features described here.

For these logs, following the RFC-5424 compliant header, the data section of our DNS Response logs contain rich information about unencrypted **allowed** DNS responses that we have identified, with respect to UDP port 53 activity. These response logs can be very useful when using advanced downstream SIEM-like tools for correlation of IP address activity as described in our Packet logs.

Our DNS Response log listens for responses that return `A` or `CNAME` (canonical name) records. Only actual responses that we have received from a far-end DNS responder, resulting from an allowed request, are included.

Here's an example unencrypted DNS response that was received, where the associated allowed request resulted in an `A` record response:

```
<30>1 2021-02-28T20:26:01.057Z myenforcer bctifw - dns_resp_log -
action=allowed, proto=UDP, reason=policy, src=192.168.0.1,
dst=192.168.0.95, src_port=53, dst_port=32975, query_type=A,
query_name=detectportal.firefox.com, answer_type=A,
answer_name=prod.detectportal.prod.cloudops.mozgcp.net,
answer_value=34.107.221.82
```

Here's an associated example where a `CNAME` response was received to an allowed request:

```
<30>1 2021-02-28T20:26:01.057Z myenforcer bctifw - dns_resp_log -
action=allowed, proto=UDP, reason=policy, src=192.168.0.1,
dst=192.168.0.95, src_port=53, dst_port=32975, query_type=A,
query_name=detectportal.firefox.com, answer_type=CNAME,
answer_name=detectportal.prod.mozaws.net,
answer_value=prod.detectportal.prod.cloudops.mozgcp.ne
```

Unlike some of our other logs which have some variability to them, our DNS Response logs are well-defined. The fields noted above, and described in the table below, are always present.

When exporting these logs to various SIEM tools, our DNS Response logs are best analyzed alongside the standard DNS logs and Packet logs described in a previous section. Correlation in modern SIEMs can be trivially accomplished using a combination of the UTC timestamp and the properly reversed `src_port` / `dst_port` mappings for the various log types.

It is critical that your parsing engines operate based on the named values and **not** their positions, as key=value position order is not guaranteed from software release to release.

The following table describes the various packet log key value pair constructs.

`dns_resp_log Key`	**Description**
`action`	This represents the verdict. This will always be reported as `allowed` for our DNS Response logs, since only requests that were allowed will have been passed through in the first place. For blocks, you should refer to our standard DNS Logs exports like you always have as described previously in this document.
`proto`	This is the identified protocol as determined from the packet's header information. Currently, this value will always show as `UDP`, since we only analyze DNS traffic riding over UDP on port 53.
`reason`	Since only allowed traffic shows up in the DNS Response logs, and that decision was made at the time as reported in the associated DNS Log as described earlier in this document, this value will always show as `policy`. For related details (such as perhaps the presence of the domain on lists), see the associated DNS Logs as previously described.
`src`	This is the source of the response, meaning the IP address of the outside server responsible for generating the response. For example, if Google's primary DNS was the responding server, you'd see 8.8.8.8 here.
`dst`	This is the protected IP on the inside that is receiving the DNS response.
`src_port`	This is the source port, from the perspective of the far-end entity responsible for generating the UDP response. Thus, it will always show as port 53.
`dst_port`	This is the destination port, from the perspective of the far-end entity responsible for generating the UDP response. That is, this is the local, protected system's port that was used to originate the request in the first place. For most modern equipment, this will generally be a high-numbered port.
`query_type`	The query type reported in the response. Generally, this will show as `A`.

`query_name`	The domain that was looked up by the originating requester, to which this response was generated. Examples include things like `cnn.com` or `foxnews.com` or `google.com` or `bing.com`, and so on.
`answer_type`	This will be either `CNAME` or `A`, depending on the type of answer that was logged. We currently log only those two types and no other.
`answer_name`	This is the domain name associated with the answer returned. This may or may not mimic the query_name value. A few common scenarios where it might not mimic exactly is in some load balancer configurations or a variety of more complicated ‘any’ requests.
`answer_value`	For an `A` response, this will be the IP address served up by the far-end DNS system in response to the originating protected system’s request. For a `CNAME` response, this will be a new canonical name based on the initial query_name specified. `CNAME` records always point to other domain names, and as such, are often used for aliasing.

System Logs and Audit Logs - Format Descriptions

Following the RFC-5424 compliant header, the data sections of our System logs and Audit logs contain information mostly useful for our Customer Success team when troubleshooting low-level problems or configuration issues. The formatting for the data section is generally free-form, and specific to internal system components. We recommend that if a SIEM is storing our system and audit log types, to store the data sections of those types as free-form text strings.

Generally, you do not need to be concerned with messages that show up as system or audit logs unless our Customer Success team instructs you to investigate them during a troubleshooting session.