On The Hunt: The Retroactive and Proactive Hunt for CTI Indicators

Author: Dennis Basilio, dennis.basilio@student.sans.edu
Advisor: *David Fletcher*

Accepted: *11/1/2023*

Abstract

Cyber threat intelligence (CTI) has many standards and models to define it. However, very few of these standards reach a technical level of implementation and instead leave it to the interpretation of organizations. Hunting for CTI indicators itself needs to be a well-defined process. Most organizations receive a list of indicators of compromise (IOCs) and conduct ad-hoc retroactive hunts for them within their environment for any historical hits. IOC hits are then proactively searched for in newly ingested data. The results should be saved to a dataset separate from the original data to avoid rerunning these hunts, allowing for a quicker review and longer retention, whereas the original data could age out. A standard hunt framework that accounts for retroactive and proactive hunts while integrating a separate dataset for IOC hits addresses the ad-hoc nature of these hunts.

1. Introduction

With the rise of cybercrime services becoming more easily accessible, cyber-attacks ranging from phishing campaigns to ransomware attacks have become more prevalent impacting nearly one-third of companies (Huang, 2018). To combat these cyber-attacks, cyber threat intelligence (CTI) enables security teams and practitioners to identify indicators of potentially malicious activity occurring on their systems and networks. Some organizations may have a dedicated threat intelligence team. In contrast, others may rely on open or closed-source CTI feeds to obtain these indicators of compromise (IOCs), but what do security teams do when they receive a list of new IOCs?

Conducting a retroactive hunt for these IOCs can be tedious and time-consuming, especially when searching large sets of heterogeneous data. Upon completion, the results of any IOC hits may be exported (such as to a CSV) and shared with stakeholders. In addition, going forward, newly ingested data must be proactively searched for IOC hits to respond quickly to potential threat actor activity. If the IOC hits need to be revisited or searched for again, the original data may have aged out, depending on the retention policy. The lack of a standard for conducting retroactive and proactive hunts, in conjunction with different operational environments and logging capabilities, results in various implementations and methodologies to address this hunting problem.

This research proposes a standard hunting framework. By adopting this framework, an organization will benefit from the retroactive and proactive hunt capabilities, which creates a smaller dataset that is quickly reviewable, updateable with new indicator hits, and maintained beyond the retention of the original data. CTI encompasses more than hunting for IOCs, as the input and output of this framework account for a larger lifecycle.

1.1. Cyber Threat Intelligence (CTI) Overview

Cyber threat intelligence (CTI) helps security teams stay informed about adversaries targeting their organization. Simply put, it is “data that is collected, processed, and analyzed to understand a threat actor’s motives, targets, and attack behaviors” (Baker, 2023). The CTI lifecycle takes this data and turns it into actionable intelligence. It involves these six steps: requirements, collection, processing, analysis, dissemination, and feedback (Baker, 2023).

Dennis Basilio, dennis.basilio@student.sans.edu

The requirements dictate the goal of integrating CTI into current processes and influence subsequent steps in achieving it. The collection step varies for organizations depending on their resources, such as open or closed-source CTI feeds or a dedicated threat intelligence team to collect intelligence. This intelligence then needs to be processed so that it is in a more readily consumable format either by another human or a tool such as a Security Information and Event Management (SIEM) system. The analysis step takes the processed data and produces a product that meets the requirements from the initial step. The product is then disseminated to the relevant stakeholders, such as the system owners or business leadership, where feedback is gathered and fed back into the CTI lifecycle. The hunt framework described in this research sits between the processing and analysis steps, taking in a collection of IOCs as an input and producing a product as an output that can then be disseminated.

This feedback cycle enables organizations to follow a process to start integrating CTI into their workflows. The vagueness of this loopback cycle results in various interpretations and implementations, which is part of a more significant problem with CTI, as identified in the following research.

1.2. Related Work

A review of prior research identified a lack of focus on the hunting process. However, some articles identified different standards, methodologies, and frameworks with various levels of interpretations and implementations across organizations, leading to inconsistencies in handling CTI data. As an example, “Little emphasis has been given to developing a comprehensive cyber threat intelligence ontology with existing efforts not being thoroughly designed, non-interoperable and ambiguous, and lacking semantic reasoning capability” (Mavroeidis, 2021). An example is the format in which IOCs can be shared between organizations. Mavroeidis identifies various sharing standards, such as the Structured Threat Information eXpression (STIX), Malware Attribute Enumeration and Characterization (MAEC) from MITRE, and OpenIOC from Mandiant (Mavroeidis, 2021). STIX is a language used to exchange CTI between organizations easily, and it is the most popular of the three. MAEC is another language intended for expressive malware sharing and has integrations with STIX. OpenIOC is like both STIX and MAEC

but focused on lower-level IOCs. Mapping to one of these standards can make the hunt framework more robust and capable of sharing any IOC hits. The following research presents an entirely new model to address their identified shortcomings.

Amaro proposed a methodological framework with an eight-step CTI model to collect, organize, filter, share, and visualize CTI data (Amaro, 2022). The eight steps are management, indexer, collect, generate, search, share, visualization, and analysis. Amaro's research also included building a tool focused on the last two steps to aid analysts in visualizing indicators and encompassing the entirety of the CTI lifecycle. Because their proposed model covers multiple steps, their technology stack is more complicated than the one proposed in this research. Regarding the search step, minimal information was given on how to search. When detailing the search step, Amaro says, "It uses all technology embedded under it in order to allow fast and efficient search from the user's perspective" (Amaro, 2022). The requirements for the search step were that the data must be easily explorable and manipulated. This research addresses this search step by creating a hunt framework that provides the mechanisms to achieve those requirements and identify IOC hits.

2. Research Method

The hunt framework was developed with retroactive and proactive hunt capabilities and an IOC hit dataset that is quickly reviewable, updateable with new indicator hits, and maintained beyond the retention of the original data. This hunt framework lives in the processing and analysis steps of the CTI lifecycle and builds upon the search step from Amaro's CTI model. It assumes that an organization already has a list of IOCs that acts as an input for the hunt framework, which can then be used in a query to build an IOC hit dataset. This dataset is the framework's output, which can then be used for any further analysis or investigation and shared with organizations. The following is a breakdown of the hunt framework methodology:

1. A new IOC is added to the list of IOCs to be searched and marked as new.
2. A search is run conducting a lookback search for the indicator over a specified timeframe (i.e., last 30 days).

3. Any IOC hits are saved to an IOC hit dataset, and the IOC is no longer marked as new.
4. Another search is then run hourly to search for any IOC hits in the last hour.
5. Any new IOC hits are then saved to the IOC hit dataset until they are removed from the IOC list.

Step two addresses the retroactive hunt for an indicator, whereas step four addresses the proactive hunt for an indicator. The timeframes and frequency of these searches should be adjusted to enable a quicker response to any hits but are dependent on the organization's preferences and capabilities (Oasis, 2022). Steps three and five address the generation of a smaller dataset that can be used to track IOC hits and retained longer than the original data. The dataset can then be used to validate and tune the IOCs that are being hunted. To test this framework, Splunk was used as the platform to conduct both the retroactive and proactive hunts while also providing a place to log any IOC hits. To simplify this research, the types of IOCs were limited to IPv4 addresses.

2.1. Environment Overview

The test environment consisted of a Splunk Docker container that acted as an all-in-one instance rather than a distributed environment seen in larger organizations. A distributed environment would allow for better scaling and resource allocation to support a larger dataset, but it is independent of this experiment due to many factors related to its implementation. The environment is fully replicated and maintained within a GitHub repository, and more details can be found in Appendix A.

The Docker container acted as both the search head and indexer, as well as other tertiary roles that are usually hosted in separate Splunk instances. A search head is where a user interacts with the data via search queries, reports, or dashboards. An indexer is where data is indexed and logs are forwarded. Three critical features of Splunk that will be heavily utilized are the summary index, key-value (KV) store, and automatic lookups.

2.1.1. Summary Index

A summary index is intended for quicker searching across a large dataset. As a result, "Searches you run against the completed summary index should complete much faster than similar searches run against the source dataset" (Splunk, n.d.). A summary

index is built via a scheduled search, and it contains the statistical results rather than the raw event where the indicator hit was observed. This makes it an ideal place to log any IOC hits seen across the larger dataset and will be used for steps three and five of the hunt framework.

2.1.2. KV Store

A KV store is a collection of key-value pairs where individual records can have Create-Read-Update-Delete (CRUD) operations performed against them. These actions can be accomplished via the Splunk REST API or lookups within the Splunk search language. The flexibility to interact with and manipulate a KV store makes it ideal for maintaining a list of IOCs addressing step one of the hunt framework.

2.1.3. Automatic Lookups

Automatic lookups enrich raw data with additional fields if a match between a field value and a lookup value exists. For example, a lookup may contain a list of IP addresses and fields detailing geolocation information. Suppose an IP address from that lookup matches one in a destination IP address field. In that case, that destination IP address field will have additional fields enriching the data with the associated geolocation data. This capability will make it easier to identify events that contain an IOC, as additional fields will be available for those events.

2.2. Data Sources

The data sources used for this experiment were generated using the Splunk Eventgen app. The app simulated various data sources to create an environment where searching would be done across heterogeneous data. The Eventgen app can randomly generate various fields depending on the data type. Events with IOCs were also generated to guarantee that the hunt framework would identify the IOC hits. The formats of the data used were CSV or JSON but can be expandable to free text, XML, and other common log formats.

2.2.1. Firewall Logs

The firewall data was in a CSV format with the following fields: a timestamp, source IP address, destination IP address, source port, destination port, application,

protocol, action, and bytes. None of the logs represented actual traffic, as all the fields were randomly generated. Any event with an IOC would have an IPv4 IOC within the destination IP address field. These logs were stored in the firewall index.

2.2.2. DNS Logs

The DNS logs were in a JSON format with the following fields: a timestamp, destination IP address, destination MAC address, destination port, message type, name, protocol stack, query, query type, reply code, reply_code_id, response time, source IP address, source MAC address, source port, time taken, transaction ID, transport, and TTL. This was based on the output of a Splunk Stream DNS log. Most fields were randomly generated, and any IOC hits occurred within the answer field. These logs were stored in the DNS index.

2.2.3. Intel logs

The intel logs were not generated from the Eventgen app but from the hunt framework, identified while searching across the above datasets for any IOC hits. These logs were stored in the intel summary index.

2.3. Variables

The creation of a new dataset warrants a comparison with the original data to evaluate the efficacy of this new framework. The search query used, and the time range set were independent variables, whereas the search completion time and event volume were dependent variables. The dependent variables acted as metrics that would be used for the comparison.

2.3.1. Search Query

The search query used to search across the original data for indicator hits would also be used to populate the summary index. This query was built to account for various log formats and was used in steps two and four of the hunt framework. It must ensure that any logs where an IOC may be seen are searched. The results should also include relevant data that provides context on what occurred with the IOC hit, not just that a hit was observed. A separate query was built to search the summary index. This query would be

more straightforward as the data in the summary index comprises the statistical results from the original search query.

2.3.2. Time Range

Time range measures the period over which indicators are searched. Searching over a more extended period means a larger volume of events included from the search source, thus impacting search completion time and event volume. The retroactive hunt query in step two and the proactive hunt query in step four have different time ranges. The retroactive hunt query has a longer time range, given that it looks at historical IOC hits, but this would need to be adjusted to ensure the search is still performant. The proactive hunt query ran over a specified frequency, such as every hour looking at the last hour of logs every hour. The frequency for both the retroactive and proactive hunts can be increased or decreased to ensure IOC hits are tracked promptly and conform with Service Level Agreements (SLAs) established by the organization. An increased frequency may result in a quicker response but may increase the amount of noise, whereas a decreased frequency may provide a better historical context for an IOC but may go beyond its lifespan and usefulness (Oasis, 2022).

2.3.3. Search Completion Time

Measured in seconds, search completion time measures how long a search for an IOC takes. It is dependent on an optimized search query and a reasonable time range. Both must be performant to ensure the search is completed within a reasonable time while taking up a manageable amount of system resources.

2.3.4. Event Volume

Event volume measures how many events are scanned for an IOC hit. Like the search time completion, this is dependent on an optimized search query and a reasonable time range. Both ensure that only the relevant events are searched.

3. Findings and Discussion

Implementing the hunt framework within Splunk takes advantage of Splunk features such as the summary index, KV store, and automatic lookups to implement the various steps. The metrics used to compare raw data and IOC hit dataset searches not only revealed some of the benefits of the framework but also highlighted some of the shortcomings of the test environment.

3.1. Implementation of the Hunt Framework

The hunt framework is meant to be vendor agnostic, where this experiment demonstrates an implementation within a Splunk environment. The framework, when applied to Splunk, follows these steps:

1. A new IOC is added to the KV store that contains a list of IOCs to be searched for and is marked as new.
2. A Splunk query is run hourly, conducting a lookback search for any new indicators that were added in the last hour over a specified timeframe (i.e., last seven days).
3. Any IOC hits are then saved to the intel summary index, and the IOC is no longer marked as new.
4. Another query is then run hourly to search for any IOCs within the KV store searched for hits within the last hour.
5. Any new IOC hits are then saved to the intel summary index. This IOC will continue to be searched for until it is removed from the KV store. IOC validation, tuning, and retiring are out-of-scope for this experiment.

The following diagram details the framework.

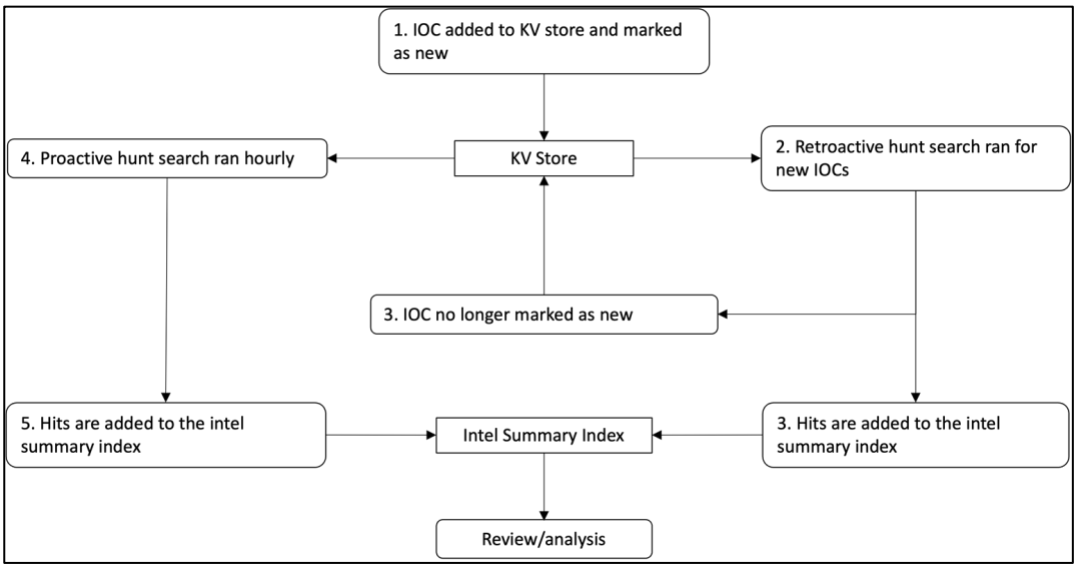


Figure 1. Hunt framework methodology diagram

A KV store is used to house the IOCs in step one and contains the following about each IOC: the source of the IOC, an arbitrary confidence level, and whether it has been searched for within the environment already. No real IOCs were used for this research, and a Splunk query generated the fake IOCs within the KV store. The placeholder value "indicator_placeholder" ensures the successful run of both the retroactive and proactive hunt queries. Without it, the macro used within the query would return an empty string, resulting in every event being scanned.

indicator	searched	confidence	source
123.123.123.123	0	low	external
124.124.124.124	0	medium	internal
125.125.125.125	0	high	internal
126.126.126.126	0	medium	external
indicator_placeholder	0		

Figure 2. KV store contents of fake IOCs

An automatic lookup is then used to enrich the raw data with the source, confidence, and the IOC as new fields. This lookup is needed to quickly identify events where an IOC was seen, as these fields will be null for unrelated events. The automatic lookup is created only for fields where an indicator may be found, such as a destination or source IP address or a DNS query answer.

dest_ip	>100	100%	String
dest_ip_cti_confidence	3	0.01%	String
dest_ip_cti_indicator	3	0.01%	String
dest_ip_cti_key	3	0.01%	String
dest_ip_cti_source	2	0.01%	String

Figure 3. Automatic enrichment of dest_ip field

Step two used an hourly search to conduct the retroactive hunt for any new IOCs that were added in the last seven days. Once the new IOC was searched for, it was no longer marked as new, preventing it from being searched retroactively. This period was arbitrarily chosen for this experiment but will likely be much longer in production environments (such as looking back 90 days for any IOC hits). Figure 4 shows the query that is used for the retroactive hunt. Line 1 lists all the indexes that will be searched, while line 2 is a macro that only lists new IOCs. Lines 3-15 generate a table containing any IOC hits and identify a risk item that communicated or interacted with the IOC while including any relevant fields to provide context around the event. While the firewall and DNS logs may not share the same fields, only the relevant fields will be saved for each event when written to the intel summary index, which is where step four is accomplished. Lines 16-17 output the results to the intel summary index as JSON, thus separately storing the IOC hits from the original data. Lines 18-27 update the KV store, marking IOCs as searched. This prevents these IOCs from being searched for again in the retroactive hunt query.

```
1  index IN (firewall, dns)
2     [| `indicators(0)`]
3  | eval threat_indicator=coalesce(answer_cti_indicator, src_ip_cti_indicator, dest_ip_cti_indicator)
4  | eval threat_field=case(threat_indicator==answer_cti_indicator, "answer", threat_indicator==src_ip_cti_indicator,
5     "src_ip", threat_indicator==dest_ip_cti_indicator, "dest_ip")
6  | eval threat_key=coalesce(answer_cti_key, src_ip_cti_key, dest_ip_cti_key)
7  | eval threat_confidence=coalesce(answer_cti_confidence, src_ip_cti_confidence, dest_ip_cti_confidence)
8  | eval threat_source=coalesce(answer_cti_source, src_ip_cti_source, dest_ip_cti_source)
9  | eval risk_field=case(threat_field IN ("answer", "dest_ip"), "src_ip", threat_field IN ("src_ip"), "dest_ip")
10 | eval risk_item=case(risk_field=="src_ip", src_ip, risk_field=="dest_ip", dest_ip)
11 | stats values(src_ip) as src_ip values(dest_ip) as dest_ip values(src_port) as src_port values(dest_port) as dest_port
12    values(app) as app values(protocol) as protocol values(action) as action values(bytes) as bytes
13    values(dest_mac) as dest_mac values(src_mac) as src_mac values(answer) as answer values(query{}) as query
14    values(query_type{}) as query_type values(reply_code) as reply_code values(transaction_id) as transaction_id
15    count by _time index sourcetype source host threat_indicator threat_field threat_key threat_confidence threat_source risk_item risk_field
16 | collect index=intel output_format=hec testmode=true
17 | collect index=intel output_format=hec file=cti_hits_summary
18 | rename threat_indicator as indicator threat_key as key threat_confidence as confidence threat_source as source
19 | table indicator searched key confidence source
20 | dedup indicator
21 | append
22    [| inputlookup cti_indicators
23    | eval key=_key
24    | fields indicator searched key confidence source]
25 | eval searched=if(indicator=="indicator_placeholder",0,1)
26 | dedup indicator
27 | outputlookup cti_indicators append=true key_field=key
```

Figure 4. Retroactive hunt query

Step five used a similar search from step two for the proactive hunt that searches for IOCs no longer marked as new. It only uses lines 1-17, and line 2's macro has a value of "1" to search only for indicators not marked as new. This search also ran hourly, looking at the last hour of recently ingested data. Because both the retroactive and proactive hunt queries ran hourly, they ensure complete coverage of all data that is ingested.

3.2. Output Comparison

Searching across heterogeneous datasets such as firewall and DNS logs meant building a query that accounted for the variation in the log format, such as the one shown in the previous section. For this research, the firewall and DNS logs were in CSV and JSON format, respectively. The retroactive and proactive hunt queries take these different datasets and standardize them to the JSON format. The resulting dataset is more easily searchable. The following is a sample of the data from each data source and their respective (and heterogeneous) formats.

index	_raw
firewall	2023/12/17 01:02:09,160.28.93.23,159.107.54.163,59631,445,smb,tcp,allowed,7913
dns	{"timestamp":"2023/12/17 01:02:05","dest_ip":"167.176.233.243","dest_mac":"6a:84:d0:ab:15:36","dest_port":"53","message_type":["QUERY","RESPONSE"],"name":["10.155.48.177"],"protocol_stack":"ip:udp:dns","query":["ngvpngydz.taf"],"query_type":["A"],"reply_code":"NoError","reply_code_id":0,"response_time":262,"src_ip":"28.148.97.225","src_mac":"3b:c1:76:20:1a:bc","src_port":"49177","time_taken":262,"transaction_id":46674,"transport":"udp","ttl":[229,49,49]}
intel	{"time":1702742209,"host":"dns-svr1","source":"dns.sample","sourcetype":"dns","index":"intel","event":"{}","fields":{"count":"1","query":["rkltvdypic.pmo"],"answer":["123.123.123.123"],"src_ip":["107.178.183.221"],"dest_ip":["131.246.62.185"],"src_mac":["16:f9:6b:68:e8:d9"],"dest_mac":["2f:94:d1:2d:0b:33"],"src_port":["51154"],"dest_port":["53"],"risk_item":"107.178.183.221","threat_source":"external","threat_indicator":"123.123.123.123","query_type":["A"],"reply_code":["NoError"],"risk_field":"src_ip","threat_key":"657e344dc6e79ace9c0b8ea0","threat_confidence":"low","threat_field":"answer","transaction_id":["46674"]}}

Figure 5. Sample raw data comparison

Searching the intel index used a more straightforward query, shown in Figure 6, than the previous one used to search across the different datasets, as the format of the logs was consistent regardless of the source. The benefits become prevalent when organizations must hunt through a broader variety of data. This index can also have a more extended retention period than the firewall and DNS indexes, thus outliving the original data in which the IOC hits were originally found.

```
index=intel
| stats values(src_ip) as src_ip values(dest_ip) as dest_ip values(src_port) as src_port values(dest_port) as dest_port
    values(app) as app values(protocol) as protocol values(action) as action values(bytes) as bytes values(dest_mac) as dest_mac
    values(src_mac) as src_mac values(answer) as answer values(query{}) as query values(query_type{}) as query_type
    values(reply_code) as reply_code values(transaction_id) as transaction_id
    count by _time original_index original_sourcetype original_source host threat_indicator
    threat_field threat_key threat_confidence threat_source risk_item risk_field
```

Figure 6. Query used to view IOC hits within the intel index

3.3. Search Completion Time Comparison

To test the performance of the searches, a new Splunk container was generated for each search. Each search had a job inspector that provided the search's metrics. The metrics included the number of results returned, the number of events scanned, and the completion time. The retroactive hunt queries were done at a seven-day lookback, whereas the proactive hunt queries were a 60-minute lookback. For each time frame, three searches were run three times, and an average of the times was then taken. The first query acted as the baseline and searched for IOC hits within the specified timeframe. The second query was used for the retroactive and proactive hunts, which encompassed writing the results back to the intel index. The final query was a search against the intel index, where only the IOC hits were stored.

Over a seven-day timeframe, the retroactive hunt query took slightly longer than the baseline query, but the intel query saw a significant improvement in search time. While the retroactive hunt query was like the baseline search, the added actions of writing to the intel summary index and updating the IOC KV store may lend to the additional search time.

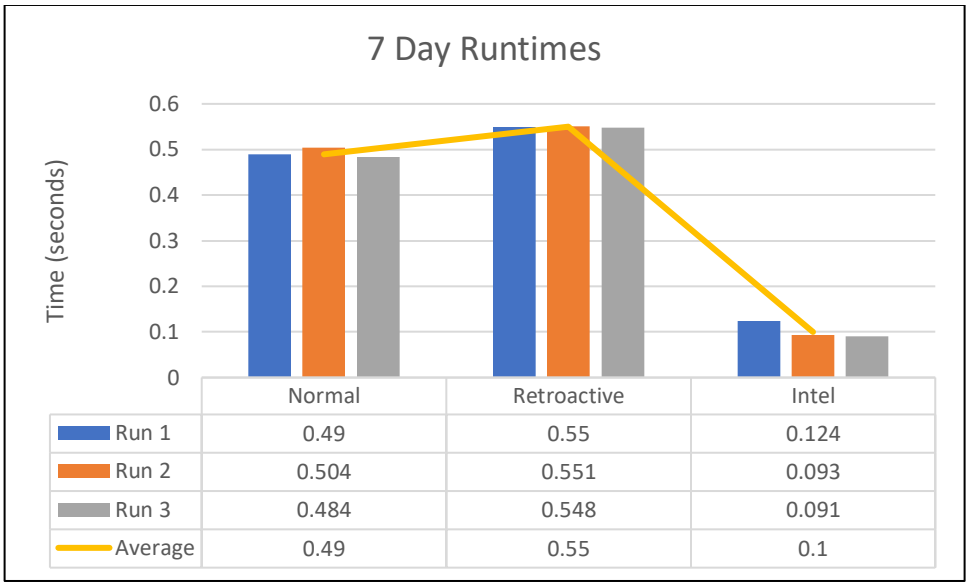
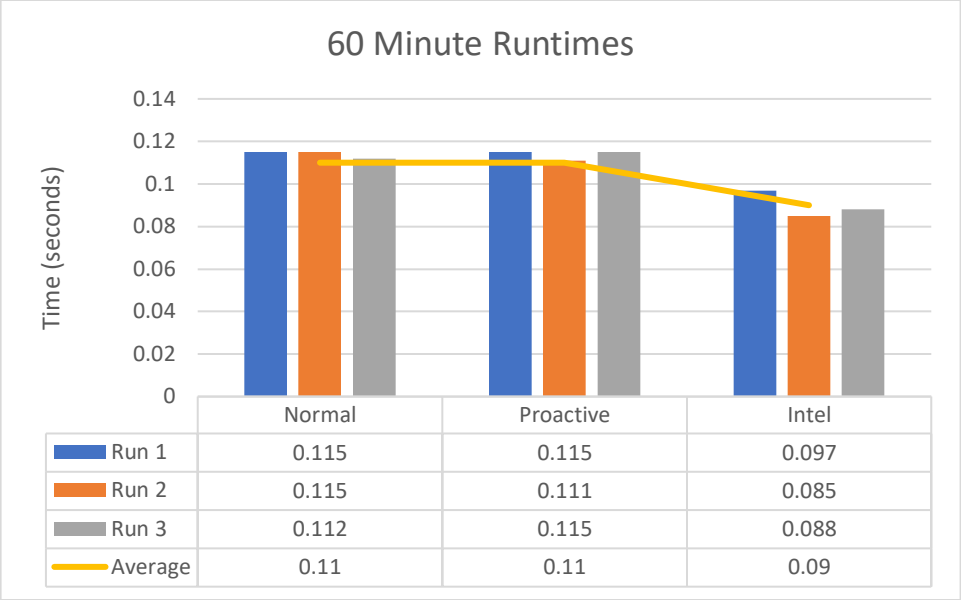


Figure 7. Seven-day run metrics

The 60-minute timeframe yielded similar results to the Seven-day timeframe but with fewer discrepancies. The proactive hunt query performed similarly to the baseline query, while the intel query only saw slight improvements. This is likely due to the volume of events searched, given that a 60-minute timeframe is much shorter than a Seven-day timeframe.



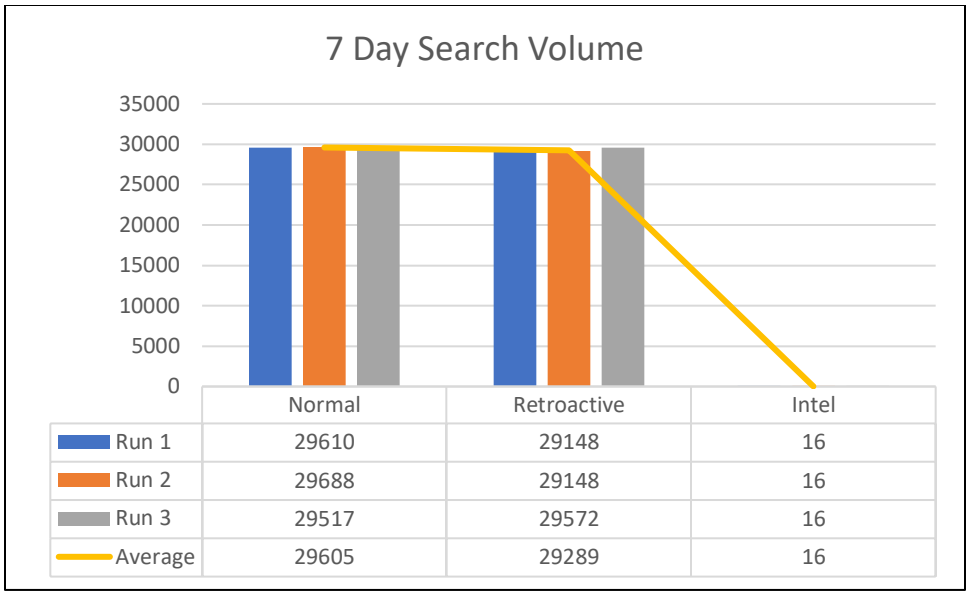
	Normal	Proactive	Intel
Run 1	0.115	0.115	0.097
Run 2	0.115	0.111	0.085
Run 3	0.112	0.115	0.088
Average	0.11	0.11	0.09

Figure 8. 60-minute run metrics

Utilizing this framework highlighted a sizable drop in search time performance when searching against the smaller dataset of IOC hits. While the search times dropped less than a second, searching the intel index showed more significant improvements as the search timeframe was extended.

3.4. Event Volume Comparison

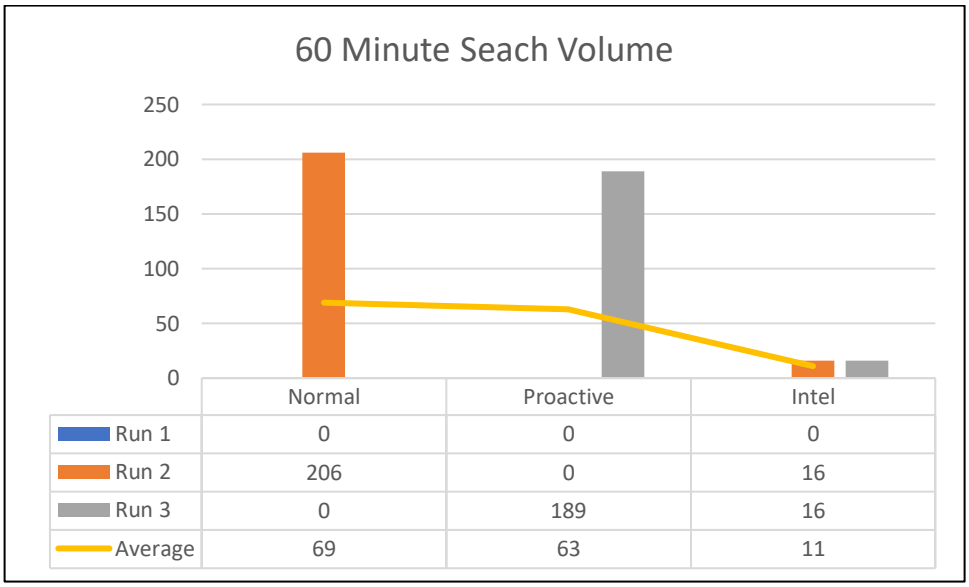
Event volume showed significant improvements when comparing searches conducted against the intel index and raw data. Search event volume varied from run to run due to the data being randomly generated each time the container was regenerated. The metrics were pulled from the exact searches run for the comparison of the search completion time. Over a seven-day timeframe, a significant drop in event volume was observed since searching the intel index returns IOC hits saved to it and does not include any other raw data.



	Normal	Retroactive	Intel
Run 1	29610	29148	16
Run 2	29688	29148	16
Run 3	29517	29572	16
Average	29605	29289	16

Figure 9. Number of events scanned for each search over seven days

The 60-minute timeframe saw more variety due to the randomness of the generated data, but whenever there were hits, the intel query was more performant.



	Normal	Proactive	Intel
Run 1	0	0	0
Run 2	206	0	16
Run 3	0	189	16
Average	69	63	11

Figure 10. Number of events scanned for each search over 60 minutes

3.5. Shortcomings

Testing the hunt framework within a Splunk environment relied on taking advantage of the native features of the platform. Implementation of the framework could vary wildly depending on the organization and the capabilities of the selected platform. Due to the constraints of the test environment, a more extensive test dataset could not be generated.

Dennis Basilio, dennis.basilio@student.sans.edu

The smaller dataset had more to do with the limitation of the Eventgen app used to generate the random data, as it would not generate the expected amount of data or would occasionally need to be restarted to start generating more events. Enterprise environments usually have a more robust platform for ingesting data.

The retroactive and proactive hunt queries are automated for this framework to work, so an analyst doesn't have to run them. They would run a search against the intel index when analyzing IOC hits. This means analysts no longer have to run these longer lookbacks manually. The impact of these searches is acceptable if the Splunk license that was used was based on data ingest. However, for organizations that rely on Splunk's infrastructure licensing, which is based on vCPU utilization, the performance of the searches will have a more significant impact. This may impact both the frequency and timeframe in which the searches are run. This is more of a shortcoming to Splunk rather than the hunt framework itself.

Another potential shortcoming with the automated hunt queries is that because these queries run hourly, there can be up to an hour delay before an IOC hit is written to the intel summary index. This could be remediated by increasing the frequency of the queries to every 15 or 30 minutes to reduce the delay, but this could add more strain to the environment with the increased number of queries being run. For time-sensitive IOCs, ad-hoc searches can still be run.

The IOC type was limited to IPv4 addresses. Searching for different types of IOCs, such as IPv6 addresses, hashes, and domains, could result in less performant searches. The search would have to account for the various IOC types, or separate searches would have to be run for each type. The hunt framework calls for a method to search IOCs, so the capability would be dependent upon the platform on which it was implemented.

4. Recommendations and Implications

When implementing this framework in an enterprise environment, four design principles should be considered for easier sharing: CTI must be consolidated and consistent, centralized and unified, responsive, timely, targeted, and accessible and pre-analyzed (Kotsias, 2022). The hunt framework accounts for these principles while

addressing a small portion of the CTI lifecycle. The input to the framework is a database containing any IOCs that an organization would want to hunt for, which assumes that the data is already centralized and unified. The framework's output is any hits identified during the hunt, which can be used for further analysis or dissemination to any relevant stakeholders. The automated nature of the framework provides timely and targeted data, as each hit also shows a risk item that was seen communicating with an IOC. To implement this framework, there are other considerations, especially regarding a larger, more varied dataset, to ensure that these design principles are met. Additional research needs to be done to expand and define this hunt framework further to make it more easily accessible to different organizations.

4.1. Recommendations for Practice

To handle searching across large datasets, the query used needs to be optimized to be more performant and have little impact on the environment. The query used in this research searched for the raw IP address. When Splunk searches for the IP address, the actual IP address isn't searched for, rather, "If you search for the IP address 127.0.0.1, Splunk software searches for '127 AND 0 AND 1' and returns events that contain those numbers anywhere in the event" (Splunk, n.d.). To search for the full IP address, Splunk has a built-in search function called TERM that can be used to match full phrases. Using TERM(127.0.0.1) within the query searches the IP address 127.0.0.1. This results in fewer events scanned, thus improving performance.

The IOC hit dataset likely contains a much smaller dataset than any other data source an organization may collect. This means that the retention period for this data can be set where it would outlive the original data where it was found. In this research, the intel summary index would have a more extended retention period than the firewall and DNS indexes. This allows an organization to identify any CTI trends without having to maintain multiple years of all the original data. It would also be inefficient to run a query going back over a year.

While Splunk was used as the SIEM of choice for this experiment, any solution could have been used if it met the criteria for the framework. The criteria include the capability to read an IOC list, search across a large dataset, and write the results back to

separate storage. The intent of using Splunk was to have a simpler technology stack that relied on a single tool rather than using multiple to address the different criteria. Organizations already using Splunk would not have to procure a separate tool to meet their CTI needs.

4.2. Implications for Future Research

One of the significant drawbacks of this experiment was the limitation of the searchable IOC types. The scope of this research was limited to just IPv4 addresses, which could be expanded to cover additional IOC types such as IPv6 addresses, domains, URLs, hashes, and more. This would require a complicated query or a breakout into multiple queries depending on IOC types. In addition to these IOCs, expanding to a threat actor's tactics, techniques, and procedures (TTPs) is an additional expansion of the framework. Searching for TTPs is more difficult since there is no simple indicator that can represent a TTP. This aligns with the pyramid of pain, where simpler IOCs such as hashes, IP addresses, and domains are easier to detect than the TTPs used by threat actors (Bianco, 2014). IOCs should follow a standard format that accounts for the various IOC types to detect this better.

The hunt framework needs to be flexible to take in IOCs of various formats and produce a product that can be easily shared. STIX was identified as a format that allows organizations to exchange CTI easily. STIX has the concept of STIX domain objects (SDOs) and STIX cyber-observable objects (SCOs) that are used to describe concepts within CTI. SDOs represent behaviors and constructs such as indicators or attack patterns, whereas SCOs represent observed facts that provide more context around an SDO (Oasis, 2021). SDOs are the IOCs fed into the hunt framework and encompass the various IOC types.

SDOs also contain various properties that can be implemented into the hunt framework. The retroactive hunt would look back at an arbitrary timeframe, seven days in this experiment, and the proactive hunt would continue to search for IOC hits until the indicator was removed. Two properties of an SDO are the "valid from" and "valid until" properties, which indicate when an SDO is considered valid. This can limit the timeframe for which an indicator is searched, and the IOC no longer needs to be deleted to prevent it

from being searched for. Each of the IOC types can also be represented by the "pattern" and "pattern type" properties, including a regular expression (PCRE), SIGMA, Snort, Suricata, or YARA (Oasis, 2021). This versatility allows IOCs to be represented in various ways as SDOs.

STIX also has the concept of a sighting STIX relationship object (SRO), which indicates that something in CTI was observed (Oasis, 2021). This aligns with the output of the hunt framework, which produces any IOC hits that were sighted within the environment. The SRO only requires what was sighted to be specified, but it does not require who sighted it or where. The hunt framework not only answers what was sighted but also identifies who sighted it (the organization itself) and where (the original data source and the risk item that interacted with the IOC).

Another benefit of using STIX over other standards is the accompanying protocol for sharing STIX data. While STIX is used for representational use, the trusted automated exchange of indicator information (TAXII) format is for operational use and supports CTI sharing using a client-server model (Schlette, 2021). Utilizing both STIX and TAXII, the hunt framework can be further improved to use a common standard and allow for easier dissemination via TAXII.

Lastly, creating a vendor-agnostic solution to implement this framework makes it viable for more organizations. The solution should integrate with the existing technology stack enhancing their capabilities rather than having them build around the solution.

5. Conclusion

Hunting for IOC hits needs to be better documented in CTI literature. High-level processes and lifecycles do not get into the technical details that organizations need to implement them. The proposed hunt framework aims to aid organizations by providing a standard process to identify any IOC hits within their environment. When an organization receives a new IOC, it is added to the database of IOCs. Every hour, any new IOC is searched for in historical data, and subsequently, any identified hits are written to the IOC hit dataset. Any existing IOC is also searched hourly so that IOC hits are identified in recently ingested data, ensuring complete coverage of their data. An analyst can then

review the IOC hit dataset for anything notable without running any ad-hoc queries against the original data, thus reducing the time the analyst spends gathering any of the IOC hits.

References

Amaro, L. C., Azevedo, B. W. P., De Mendonça, F. L. L., Giozza, W. F., De Oliveira Albuquerque, R., & Villalba, L. J. G. (2022). Methodological Framework to Collect, Process, Analyze and Visualize Cyber Threat Intelligence Data. *Applied Sciences*, *12*(3), 1205. https://doi.org/10.3390/app12031205

Baker, K. (2023). What is Cyber Threat Intelligence? *Crowdstrike*. https://www.crowdstrike.com/cybersecurity-101/threat-intelligence/

Bianco, D. J. (2014). The Pyramid of Pain. *Enterprise Detection & Response.* http://detect-respond.blogspot.com/2013/03/the-pyramid-of-pain.html

Huang, K., Siegel, M., Madnick, S.(2018). Systematically Understanding the Cyber Attack Business: A Survey. *ACM Computing Surveys.* https://dl.acm.org/doi/10.1145/3199674

Kotsias, J., Ahmad, A., & Scheepers, R. (2022). Adopting and integrating cyber-threat intelligence in a commercial organisation. *European Journal of Information Systems*, *32*(1), 35–51. https://doi.org/10.1080/0960085x.2022.2088414

Oasis Open. (2021, June 10). STIX Version 2.1. https://docs.oasis-open.org/cti/stix/v2.1/stix-v2.1.html

Oasis Open. (2022, September 15). STIX Best Practices Guide Version 1.0.0. https://docs.oasis-open.org/cti/stix-bp/v1.0.0/stix-bp-v1.0.0.html

Schlette, D., Caselli, M., & Pernul, G. (2021). A Comparative Study on Cyber Threat Intelligence: The Security Incident Response Perspective. *IEEE Communications Surveys and Tutorials*, *23*(4), 2525–2556. https://doi.org/10.1109/comst.2021.3117338

Splunk (n.d.). Use CASE() and TERM() to match phrases. https://docs.splunk.com/Documentation/Splunk/latest/Search/UseCASEandTERMtomatchphrases

Splunk (n.d.). Use summary indexing for increased search efficiency. https://docs.splunk.com/Documentation/Splunk/latest/Knowledge/Usesummaryindexing

Appendix A
Replicating this Research

The materials needed to replicate this research are provided in a GitHub repository. The Splunk container is launched via the docker-compose file, which specifies the necessary apps. The apps that are included in the repository are the Eventgen app and the custom sans-research-app app, which contains all the configuration files and searches needed to replicate this research. Once the container is launched, the following steps will generate the KV store and intel summary index:

1. Use the username "admin" and the password "sans-research2023" to log in. This will then take you to the first page titled "Fill CTI Indicator KV Store" where the KV store is first populated with the test IOCs. Click on the red box on the top-right corner of the box and select "Run Query" to bypass the warning.
2. Once it is populated, navigate to the second page titled "Retroactive Hunt" where the retroactive hunt query is run. This manual run prevents having to wait for the query to run at the hour mark. This won't show any of the IOC hits, but rather, it will show the updated IOC table that is outputted to the KV store.
3. Once completed, navigate to the third page titled "Intel Search" to search the intel summary index and see all the identified IOC hits.
4. Under the search tab, ad-hoc searches can be run against the firewall, DNS, and intel index to compare the different datasets both in format and quantity.

To view the metrics of any of the underlying searches from the dashboards above, hover over the table and click the magnifying glass on the bottom right. If a warning appears, select "Run Query Anyway." Once the search completes, select "Inspect Job" in the job dropdown underneath the search window. This will return the metrics that were used for this research such as how many hits were seen and how long the search took. The container can be taken down and brought back up, resulting in a new dataset.

Link: https://github.com/dennybdointhngs/sans-research

Dennis Basilio, dennis.basilio@student.sans.edu