Automated Cyber Threat Intelligence Generation on Multi-Host Network Incidents

1st Cristoffer Leite
Eindhoven University of Technology
Eindhoven, Netherlands
c.l.da.silva@tue.nl

2nd Jerry den Hartog
Eindhoven University of Technology
Eindhoven, Netherlands
j.d.hartog@tue.nl

3rd Daniel R. dos Santos
Forescout Technologies
Eindhoven, Netherlands

4th Elisa Costante
Forescout Technologies
Eindhoven, Netherlands

***Abstract*—The lack of automation is one of the main issues hindering the broad usage of high-level Cyber Threat Intelligence (CTI). Creating and using such information by capturing Tactics, Techniques and Procedures (TTPs) is currently an arduous manual task for Cyber Security Incident Response Teams (CSIRT). For CSIRTs, a Network Intrusion Detection System (NIDS) automates the detection of cyber threats. It provides relevant information about alerts to the analysts. This information could generate CTI reports to help others better protect themselves from similar attacks. Due to the demanding work involved in manually creating high-level CTI reports for multi-host incidents, automating this process has become increasingly important.**

In this paper, a solution is presented to automate the creation of verifiable high-level cyber threat intelligence reports by mapping chains of alerts to TTPs. The solution enables visualisation of attack chains and tactics used, but also manual analysis and validation of the reports created. The proposed approach is evaluated by comparing generating reports with existing CTI, validating any additional TTPs found. The evaluation shows that, not only it was able to match existing reports, but it was also able to improve the knowledge about these threats.

***Index Terms*—Cyber Threat Intelligence (CTI), Automation, Tactics, Techniques and Procedures.**

I. Introduction

The threat of cyber attacks is an ever-present concern in modern networked systems. Security analysts continuously monitor their networks for threats and evidence of cyber attacks. A Network Intrusion Detection System (NIDS) can provide alerts regarding potentially malicious events to analysts, but these need to be interpreted and analysed. If possible, analysts collect and share information about incidents to help others. This type of shared information is called Cyber Threat Intelligence (CTI). It includes analysed knowledge about capabilities, infrastructure, methods and victims of cyber threat actors. CTI can be organised based on its level of maturity: low-level for Technical CTI as Indicators of Compromise (IoCs), and higher-levels for Tactical, Operational CTI and Strategic CTI as Tactics, Techniques and Procedures (TTPs), Attacker Identity and Goals [1, 2].

After analysing and documenting threats in their network, analysts can publish CTI reports so others may benefit from what they learned. This information is easy to generate (and use) for low-level CTI with automated structures. Lists with allowed and blocked indicators are an example of a commonly used method for IoCs. By contrast, due to the lack of automation, high-level CTI requires a lot of manual work, both during creation and use [3, 4, 5]. As a result of the currently high effort needed to generate it, only a limited amount of reports provide higher-level CTI such as TTPs.

979-8-3503-2445-7/23/$31.00 ©2023 IEEE

The growing number of programs sponsored by national agencies, non-profit institutions and commercial organizations to share intelligence among community members show the importance of sharing CTI for defence. Examples include several country-specific or worldwide information sharing and analysis centres (ISACs) [6, 7], the Cyber Threat Alliance [8] and government initiatives such as CISA's AIS [9] and the NCSC-UK's CISP [10]. Usually, sharing intelligence is a prerequisite or allows members to access more intelligence. High-level CTI tends to be more valued than low-level [8], as the first is more difficult for the attackers to change.

In a previous work, we automated the use of high-level CTI [11], further increasing the value of sharing it. But there is still the need for an automated way of generating this information to reduce its scarcity [12]. To address this latter issue, we investigate the following research question: **RQ**: How can the creation of high-level CTI reports about network incidents be automated? This question leads us to the following sub-questions: **SQ1**: How can the output from a NIDS monitoring multiple hosts be translated into CTI automatically? **SQ2**: How to enable the validation of automatically generated CTI by an analyst when necessary before sharing it?

Our approach takes a set of correlated events classified as alerts as a starting point. NIDS and supporting tools typically provide some basic methods to correlate network events. We define an underlying structure based on alert graphs due to its compatibility with most correlation techniques [13]. We extract chains of alerts out of this graph, building simplified Knowledge Graphs (KG) called Alert Chains. These chains depict the attack propagation between hosts in different attack phases. We apply a map between alert types from the NIDS into TTPs, translating the chains into structured high-level CTI and generating machine-readable reports. As a type of KG, chains are capable of automatically generating and presenting CTI in both machine- and human-readable forms [14, 15, 16]. We also add the possibility of ranking to prioritise or filter the most important chains, providing additional structure to the analyst and helping visualise the CTI and related alerts.

We evaluate our approach on publicly available datasets containing malicious network traffic related to malware families and botnets. First, for the few samples with already available high-level CTI, we compare our automatically generated reports with the available CTI and show that our approach finds the TTPs they report, as well as additional ones. For the new TTPs our solution finds, we validate their correctness by looking at the network flow. Next, we apply our approach to a more extensive set of samples, including new families to provide a broad qualitative analysis of the generated CTI.

Our main contributions are: (1) Providing a method to automatically generate high-level CTI from network incidents, with sound reports that are even more complete than manually-created ones. (2) Enabling verifiability of the generated CTI, allowing analysts to publish reports with less effort. (3) Confirming that it works with multiple NIDS.

Below, after discussing related work on automated CTI creation in Section II, we describe our general approach in Section III before giving a concrete implementation in Section IV. We evaluate the approach in Section V, then discuss how we fill the identified gap in Section VI. Finally, we provide conclusions, limitations and future work in Section VII

II. Related Work

We outline recent works on creating high-level CTI automatically. First, we describe those creating CTI out of existing information by converting or aggregating it. Then, we focus on works that build CTI out of raw host data. Finally, we discuss how none of them covers network data with multiple hosts.

Most works on automatically creating high-level CTI focus on generating unstructured (human-readable) reports from structured (machine-readable) CTI [14, 17]. Others go the opposite direction, extracting structured CTI from existing unstructured reports, usually by applying Natural Language Processing (NLP) [18, 15, 19], and a few use the MITRE ATT&CK Framework to correlate it to TTPs [20]. Notably, Noor et al. [21] extract TTPs from unstructured CTI reports with NLP and create attack patterns for different actors based on attributed reports. They test patterns with different classifiers to improve the accuracy of their attribution.

Some works use CTI aggregation to reduce the known problem of low intersection between CTI sources. Modi et al. [22] aggregate CTI from several sources, correlate reports and create links between attack events previously considered unrelated. Kim et al. [23] also aggregate CTI data from different sources, then matches it with behaviour detected by malware analysis tools to produce new network security rules.

A significant number of recent works mention the need of automating the process of consuming [3, 4, 5] and producing CTI from raw data [24, 25]. Specifically, Haque and Krishnan [12] design a framework to allow controlled CTI sharing between multiple entities. They propose a conceptual module, referred to as a Threat Detection System, which would have the capabilities of generating the required information for automated CTI. They emphasize that there is still a need of creating such a module somehow.

In that direction, Scarabeo et al. [26] use text mining to map raw log events to CAPEC attack patterns for local analysis. Navarro et al. [27] create attack models from raw log events and match them with CTI sources. Similarly, on a previous work [11], we automated the use of available high-level CTI by linking MITRE ATT&CK TTPs and alerts to create and match detectable patterns with existing reports. All of these works extract and match patterns for local analysis without creating shareable reports. One approach that comes close to our methodology is the framework created by Landauer et al. [28], which combines IoCs and TTPs from raw host-based log data into meta-alerts and generates CTI from it.

The main focus of these works is either structuring and merging low-level CTI out of existing information or generating data on host-based scenarios. None of these works automatically generates high-level CTI reports on networks or propose a method of doing it. Therefore, despite the promising work and the identified need, the gap for for further automation in generating high-level CTI about attacks involving multiple hosts in a network remains unaddressed [24, 12, 25].

To address this gap, we build on top of our previous work [11], linking alerts and TTPs in the opposite direction and greatly expanding the mapping to other NIDS to automatically generate CTI reports about multi-host incidents in networks.

III. Approach

Our goal is to automate the creation of shareable high-level CTI reports using alerts generated by a NIDS. Given this goal and our scope, we determined the following requirements for the approach: (1) The shared (high-level) information must be interpretable and verifiable by an analyst and other parties. Thus, we use TTPs from MITRE ATT&CK framework as the standard way to express the threats [29]. (2) It needs to be aware of multiple hosts in a network, thus we apply chains. (3) It must work with new threats. In that sense, we do not rely on signature-based or tag-based approaches like clustering.

Figure 1 shows our approach. It starts with the Discovery of Structures, organising previously correlated alerts from a network into chains representing the propagation of an attack and its phases. Next, Ranking and Mapping, with an optional chain rank to help experts prioritise or filter information when they need to analyse a lot of CTI. It then automatically maps alerts to TTPs and extracts IoCs to include in a CTI report. The chains allow the creation of both machine-readable reports for automated sharing and, as a type of Knowledge Graph (KG), human-readable ones for easier validation and sharing.

A. Discover Structure

We start from a NIDS monitoring the network traffic and correlating alerts when detecting suspicious communication. The collection of correlated alerts received from the NIDS as an input can be captured and depicted as a graph.

Definition 1 (Alert Graph)**.** An Alert Graph is a Multidigraph $G := (H, A)$ comprising a set H of hosts in the network involved in alerts and a set A of NIDS alerts expressed as directed labelled edges.

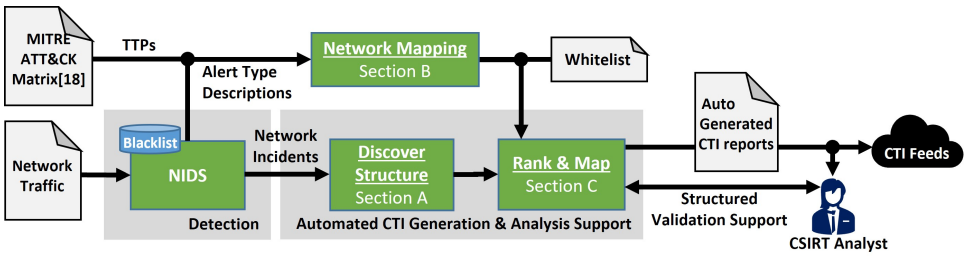


Fig. 1: Approach Overview

We give some assumptions and notation. Each alert $a \in A$ is such that $a = (src, dst, (ty, t, n, ob))$. It includes involved source src and destination dst entities, the event type ty, the time t of occurrence, the amount n of times this combination repeats within a predefined time window, and a set ob of related observables. We assume a function to extract each element written in post-fix notation. Thus, $G.H$ for the hosts in graph G, $a.src$ for the source of alert a, etc. Where applicable, we lift functions to sets e.g., $A.src = \{a.src \mid a \in A\}$. For a set of sets, we use $\bigcup$ to denote the union of the contained sets e.g., $\bigcup A.ob = \{o \mid \exists a \in A : o \in a.ob\}$.

Each event type ty has a severity sev ranging from Nothing (0), Informational (1), Low (2), Medium (3), High (4) to Critical (5). To simplify notation, we use $a.sev$ as a shorthand for $a.ty.sev$. The list of observables in ob includes the involved low-level indicators such as IPs, URLs, domains and file hashes, each marked with their type.

For a host $h \in H$, we look at the alerts leading to h with $h.aIn = \{a \in A \mid a.dst = h\}$, and alerts originating in h with $h.aOut = \{a \in A \mid a.src = h\}$. We consider hosts involved in those alerts as the the *inward neighbourhood* $h.In = h.aIn.src = \{h' \in H \mid \exists a \in A : a.src = h' \wedge a.dst = h\}$ and *outward neighbourhood* $h.Out = h.aOut.dst = \{h' \in H \mid \exists a \in A : a.src = h \wedge a.dst = h'\}$ of host h.

Alert Graphs are in tune with the output of most correlation techniques, including attack graphs and attack trees [13].

The graphs allow generating at least some machine-readable CTI already, but CTI reports can be human- or machine-readable, respectively called unstructured and structured information [21]. We want the possibility of creating human-readable reports if needed, both for sharing and for validation. Thus, we extract KGs out of it, as they also allow automated creation of human-readable CTI reports [14, 15].

We introduce the following two terms: A sequence of alerts $a_1, \ldots, a_n$ is *time consistent* when $i \leq j \implies a_i.t \leq a_j.t$, i.e. its timestamps are non-decreasing. A graph C is *rooted* if there is a node h, called root, such that all nodes and edges in C are part of some time consistent path starting in h.

From the graphs, we build time-consistent, possibly branching, connected sequences of alerts called Alert Chains. These sequences represent attack phases and the propagation between hosts [30]. In building chains, we ignore minor severity alerts (nothing and informational) as we do not consider them sufficient evidence of an attack reaching the destination host.

Definition 2 (Alert Chain)**.** An Alert Chain is an a-cyclic rooted subgraph of $G' = (G.H, \{a \in G.A \mid a.sev > 1\})$. We use C_h to denote an Alert Chain with h as a root, $C_h.H$ and $C_h.A$ for respectively the hosts and alerts in this chain and $G.AC$ for a set of Alert Chains covering all alerts in G'.

Figure 2 shows an example of multiple Alert Chains in an Alert Graph, with alerts numbered in the order they occur. In this case, there are four chains in $G.AC$. There is no need for chains C_{h2} and C_{h6}, as any sub-graphs rooted in $H2$ or $H6$ are already covered. At the same time, C_{h7} is not included in C_{h1}, lest creating a loop. Thus, it generates a separate chain and guarantees any chain as a KG: a mostly-acyclic directed graph or a Directed Acyclic Graph (DAG) in some cases. KGs can have loops if every instance is in a subset of a separate unique classification, a property called subsumption.

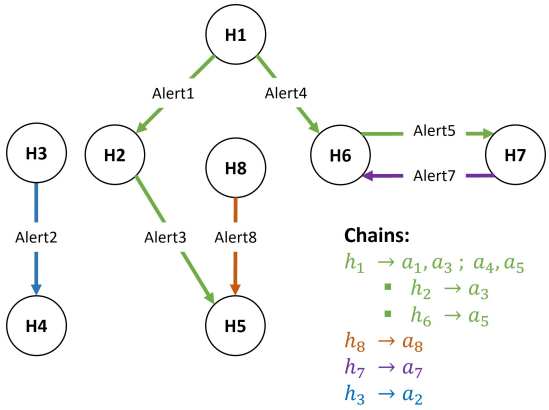


Fig. 2: A simple Alert Graph with multiple Alert Chains

As in Figure 2, there can be multiple chains in a single Alert Graph. In building the chains, only information relevant to finding the path of the attack propagation is included rather than all potential information about this path. But all auxiliary information is helpful later when analysing the incident and validating the CTI for it. To this end, we introduce the notion of enriching a chain, which gathers all alerts relevant to the chain. Below, we define these relevant alerts per host before collecting all of them.

Definition 3 (Relevant Alerts)**.** An alert is relevant to a chain if it involves the root node, or if it involves a host after that host is first seen in the chain:

$$first(h_i, C_h) = \begin{cases} 0 & \text{if } h_i = h \\ \text{Min}((C_h.A \cap h_i.aIn).t) & \text{otherwise} \end{cases}$$

$$Rel(h_i, C_h) = \{a \in h_i.aIn \cup h_i.aOut \mid a.t \geq first(h_i, C)\}$$

The function *first* captures the first moment in the chain with an indication that a host is known, and maybe compromised, by an attacker. The root of the chain, as the starting point, is considered known from the beginning (time=0). Any other host is considered known as soon as an alert of the chain reaches this host. By Definition 2, the set of incoming alerts in h_i from the chain is guaranteed to be non-empty. Thus, the relevant environment of h_i in relation to a chain C_h, given by $Rel(h_i, C_h)$, comprises those alerts involving h_i and which occur after h_i is known in that chain. The alerts in this set are subsumptive by nature. With that, we can obtain all relevant alerts connected to the hosts of a chain to enrich that chain.

Definition 4 (Relevant Alerts Set)**.** The enriched chain, defined by the relevant alerts set, contains all alerts that are relevant to any host in the chain:

$$Rel(C_h) = \bigcup_{h_i \in C_h.H} Rel(h_i, C_h)$$

Alert Chains do not replace current alert correlation techniques employed by the NIDS, but rather use their output to generate CTI. It employs a widely-used, KG structure [16, 17, 14]. The chains provide structured information about the observed events in the network. An event, as a consequence of a step taken by an attacker, can be linked to their behaviour throughout the execution of the attack [31]. The chains can complement any correlation technique that generates graph-like structures or compatible outputs, as a considerable amount do [32], and the result reflects different phases of the attacker's behaviour. This behaviour can be described by the Tactics, Techniques and Procedures deployed by them.

In the next section, we define how we output these TTPs from alerts to generate shareable reports.

B. Alert Map and Report Generation

Having found which alerts are most relevant to describe an incident using the enriched alert chains, the next step translates it into useful CTI. Bromander et al. [1] categorize CTI by assigning a Detection Maturity Level (DML). At low levels (DML 1 to 2), we have IoCs like those our alerts already provide as observables. High-level CTI includes TTPs (DML-3 to 6). To be able to provide high-level CTI, we create a map between every alert in the NIDS and related TTPs.

We refer to the MITRE ATT&CK framework as the most widely adopted way of expressing TTPs [20, 18, 15]. For each event type used by the NIDS for alerts, we map it to one or more TTPs in the MITRE ATT&CK framework [11]. Only a subset of the TTPs relate to behaviour that can be observed in the network. Thus we only need to consider a subset NM of so-called network-mappable TTPs from those in the MITRE ATT&CK matrix [33].

Definition 5 (Alert to TTP mapping)**.** A mapping function $a2n$ returns TTPs mapped to an event type as $a2n : AlertTypes \rightarrow \mathcal{P}(NM)$, with $a.a2n$ as a shorthand for $a.ty.a2n$.

The mapping function $a2n$ gives the network-mappable TTPs, including sub-techniques, for each event type and thus for each alert. With this function, we can generate high-level CTI from the alerts in a chain. Figure 3 shows an example with the highlighted alert *Remote Blacklisted Operation on file* (recommended from the NIST National Vulnerability Database (NVD)) mapped to the Techniques T1570 (Enterprise) and T0867 (ICS), both as a Lateral Tool Transfer.

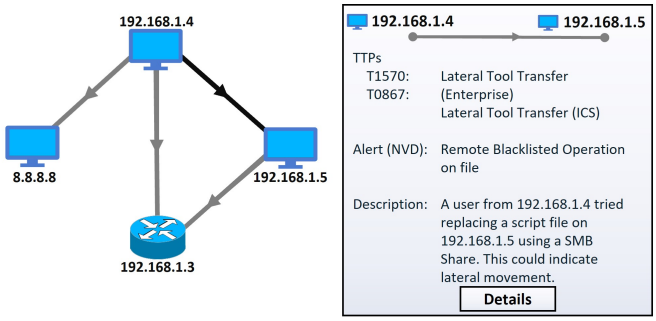


Fig. 3: Analysts can Validate the Reports with the Chains

The IoCs are the observables provided by the alerts. Some observables, while possibly meaningful for the local security analyst, are unsuited for inclusion in the report. For example, well known and trusted IPs, including some local ones, should not go in the CTI report. A set of allowed indicators WL excludes such observables from IoCs in CTI reports if necessary.

A report is a collection of CTI related to a specific incident or threat. For shareability and automated use, it is convenient to have data in structured machine-readable formats. A structured CTI report can be automatically produced by extracting related IoCs and TTPs from the relevant alerts of a chain C_h.

Definition 6 (Report generation)**.** Assuming a given set WL of allowed observables, a structured report $r = (iocs, ttps)$ containing both IoCs and network mappable TTPs can be generated out of an alert chain C_h by the function $report$:

$$\begin{aligned} report(C_h) &= (\bigcup Rel(C_h).ob \setminus WL,\ \bigcup Rel(C_h).a2n) \\ &= (\{o \mid o \notin WL \wedge \exists a \in Rel(C_h) : o \in a.ob\}, \\ &\quad \{t \mid \exists a \in Rel(C_h) :\ t \in a.a2n\}) \end{aligned}$$

For a set of chains we obtain a report by collecting all the IoCs and TTPs for each chain. While the generated report r is, by definition, structured (machine-readable) metadata, the KG construction also allows transforming this into a human-readable report if needed. Additional information such as Cyber Kill Chain phases, Common Vulnerabilities and Exposures (CVE)s, and Courses of Action are easily obtainable from the TTPs. We implement an example on Subsection IV-C.

In the next section, we show how our approach also allows the analyst to interpret and validate the reports before sharing.

C. Interpretation and Validation

We rank the chains to allow an analyst to focus on the most crucial chains detected in the network. It prioritises the most vital information to analyse more critical cases first if necessary. To this end, we first define the severity of a chain.

By Definition 1, any alert has a severity associated with its type. The severity of the alerts in a chain provides the severity of a chain. If a chain only contains low-severity alerts ($a.sev \leq 3$), it does not matter how many there are. For chains without high-severity alerts, the maximum severity that does occur applies. But the severity scales with the number of high-severity alerts ($a.sev \geq 4$). For those, we sum their severity. Thus, both the level of severity and the number of such alerts have an impact on the chain severity.

Definition 7 (Chain Severity)**.** The Severity of a chain C_h is given by a Sev function based on the severity of its alerts:

$$Sev(C_h) = \begin{cases} \text{Max}(C_h.A.sev) & \text{if Max}(C_h.A.sev) \leq 3 \\ \sum\{a.sev \mid a \in C_h.A & \text{otherwise} \\ \quad \wedge a.sev \geq 4\} & \end{cases}$$

Their severity is a crucial factor in the importance of a chains. Depending on the case, there are other aspects to consider, such as prioritising high-volume nodes and reducing clutter. In that sense, alert chains are scored and ranked based on the use case scenario. It is a way of showing the most vital information as a priority. This depends on the environment, the type of devices, and many other factors. So our approach includes an optional ranking system that could be implemented based on each case. The rank adds actionability to the report generation. An analyst can use the ranked chains to examine the automatically generated CTI. In Section IV, we give a concrete example of a ranking function $rank : G.AC \rightarrow \mathbb{N}$ for a NIDS.

By definition, any Alert Chain C_h is a KG, i.e., a directed mostly-acyclic graph. With that, automatically generating human-readable reports from the output r is also possible by using its (KG-equivalent) metadata. Figure 3 depicts a simple example where the description was automatically generated based on the graph. That description is used for human-readable versions of the report but also shown in the chains, allowing the verifiability of the reports created.

Putting all of the above definitions together, a shareable and verifiable report r can be automatically generated by following a simple process: (1) Build an Alert Graph from correlated alerts in the network, (2) Create Alert Chains that represent the behaviour of an attacker acting in the network in different phases, (3) Map the relevant alerts to TTPs, (4) Aggregate those TTPs and the IoCs involved in them, and then show this to the analyst to validate the report if needed.

After showing our general approach for structuring relevant CTI and automatically generating machine-readable reports from network data, the following section presents a concrete implementation of this approach.

IV. Implementation

Here we detail our implementation of the approach described in the previous section. In particular, we look at: (A) The mapping for the state-of-the-art NIDS used for the experiments and how to apply this to other NIDS. (B) Options for the chain ranking function which remained abstract in the previous section. (C) A concrete choice for the observables, with its lists and the format used in the report generation.

A. Network CTI Mapping

After collecting chains of alerts detected in the network, we use a map between alert types and TTPs defined in Subsection III-B to generate a list of techniques used in those chains.

In our implementation, we use TTPs from MITRE ATT&CK framework [29] due to its overall acceptance in literature [34, 20, 18, 15] and its widespread use by CTI platforms [8]. Using different frameworks is also possible, such as including Lockheed Martin's Kill-Chain and Mandiant's Attack Life Cycle [34] to represent the attack phases.

The list of alerts used in our implementation includes the event types suggested by both the NVD and the Industrial Control Systems Cyber Emergency Response Team (ICS-CERT). We map alerts in the chains to TTPs in the selected NIDS based on their type. We define the alert mapping function between the event types in the selected NIDS and TTPs.

We produced an inversed version of the mapping from our previous work [11]. We also create additional maps for two other NIDS: Zeek and Suricata. Using different NIDS changes how TTPs are mapped. On Zeek, type names and purposes are more general, making it difficult to map to the latest MITRE TTPs and its sub-techniques. On Suricata, rules receive tagged TTPs based on their type, as shown in Example 1 for Figure 3.

Example 1: Summary of a Mapped Suricata Rule Triggered

alert smb $HOME_NET any → $HOME_NET any (
msg:"Remote Blacklisted Operation on file"; (...)
threshold:type both, track by_src, count 1, seconds 60;
metadata: mitre_tactic_id TA0040, mitre_technique_id T1570, ics_tactic_id TA0109, ics_technique_id T0867;
)

Mapping alerts in a chain results in TTPs for inclusion in a report. Next section discusses our implementation of the ranking to help analysts prioritise chains.

B. Chain Building and Ranking

We use alerts correlated by adapted Alarm Graphs [35] as input in our implementation. To create the chain set ($G.AC$ in Definition 2) to be ranked, we generate rooted sub-graphs by, starting from the source host h of the alert with the lowest timestamp, iteratively traversing all outgoing edges in order of time. After the root, we consider only those alerts with a timestamp greater or equal to the edge we used to reach the node, ensuring time consistency. We omit 'back edges' which would create a loop. Once there are no more alerts to add to the current chain, we build the next uncovered chain in the graph. Non-overlapping chains avoid redundant information.

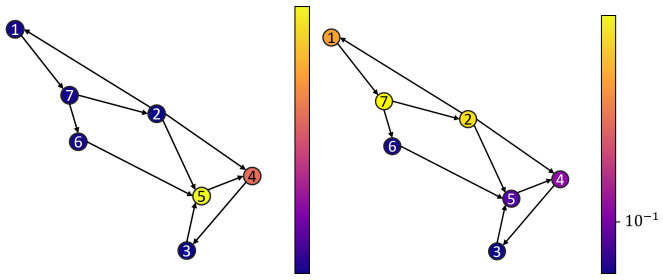
Note that the chain-building order impacts the resulting $G.AC$. With a different order, alert distribution over the chains changes. Here, time-consistency gives the attack propagation.

To rank the chains, we provide several features that help identify the most crucial chains in different settings. Adjusting the ranking to the use case is possible by choosing specific features. We use took inspiration from alert correlation techniques and graph-based CTI aggregation to choose suitable ranking parameters [15, 35]. The selected features are Severity (Definition 7), Depth, Popular Nodes, and Influential Nodes. Below, we discuss why we consider each of them relevant.

Ranking Formula: Our implementation of the ranking uses the combination of the features (F_i) mentioned above and an associated weight factor (x_i) giving the score as: $\sum(x_i * F_i)$.

1) Severity: High-severity alerts are analysed first. Chains with many high-severity alert types, given by the chain Severity score Sev of Definition 7, are more important.

2) Depth: Chains with more hops, i.e., intermediate steps in the attack, may give a more complete view of the propagation. Thus, the chain depth, given by the length of its longest path, is also considered an important factor when ranking chains.



(a) Popularity Degree Centrality (b) Influence Eigenvector Centrality

Fig. 4: Popularity and Influence in a Graph of Network Alerts

3) Popular Nodes: Some chains contain the hosts most targeted by alerts. We call them popular nodes. Giving chains with these nodes a certain degree of importance during analysis can help highlight focal points of attacks. The popularity of a node is the inward degree centrality in relation to all chains: its 'in-degree' in the union of all chains in G. So $h.Pop = \#(h.aIn \cap \bigcup G.AC.A)$. Figure 4a shows one example. The chain popularity ($Pop_{\#}$) is given by the number of devices with the maximum popularity in the chain.

4) Influential Nodes: In contrast to popular nodes, some can be the source of a high quantity of alerts. We refer to them as influential nodes. Considering $O(h) = \#(h.aOut \cap \bigcup G.AC.A)$ as the outward degree centrality of a node h, we define the influence of a node as the inverse of its normalised eigenvector centrality in relation to the whole, given by $h.Infl = \sum_{u \in h.Out}(O(h)^{-1} \times u.Infl)$. Influence is equivalent to the PageRank algorithm [35] applied to the reverse of the graph. Figure 4b shows an example.

We selected the features above for the setting used in the evaluation in the next section. In tuning the ranking for more specific environments, additional features may apply, e.g. Criticality of devices (Purdue Levels) and Frequency of Types.

The ranking score provides analysis priorities for an analyst, but if there is an overwhelming number of chains it may be necessary to limit how many to consider. Using the ranking score by setting a minimum threshold or taking only top-scoring chains allows this filtering. Our implementation supports setting a maximum number of top chains through a configuration file.

C. Generate CTI Report

Observables add possible IoCs to the generated reports. As observables, we consider IPs, URLs and domain names of the involved hosts or otherwise mentioned in the alerts. The list of allowed observables prevents including legitimate sites. For the list of allowed ones, we use all observables internal to the monitoring organization and 'well-known' ones, including the top-known domains and their IPs.

We generate reports using the Structured Threat Information Expression (STIX) 2.1 format due to its wide acceptance and versatility [2, 36]. In this format, CTI data are structured objects called STIX Domain Object (SDO)s, and reports are Bundles of CTI. It is important to note that when multiple alerts in a chain are related to the same TTP, it is possible to represent that to an analyst locally. But current structured CTI formats do not support this representation. Thus, the shareable exported metadata only contains each observed TTP once.

To complement this information, we obtain additional data from MITRE about each of the TTPs, including respective Cyber Kill Chain phases in some cases. If mitigation techniques or efforts to stop an attack exist for individual TTPs, they are also added as a Course Of Action.

Applying a similar mapping to host data logs locally [28] or while analysing available executables [37] would allow adding host-based information to the reports, including CVE, Common Vulnerability Scoring System (CVSS) and host-related TTPs. But because those steps are out of scope for this work, the report generation only takes the CVE numbers the MITRE mapping step provides to include them as *Vulnerability* objects in the final report.

The next section evaluates our solution by applying it to different sandboxed scenarios available and analysing the reports created.

V. Evaluation

A. Experimental Setup

1) Datasets: In these experiments, we employ a dataset including different attack scenarios in a network. The goal is to evaluate if the TTPs in the machine-readable CTI reports automatically generated by our solution out of attacks observed in network traffic match or even improve over existing manually created reports. Also, it tests how a Cyber Security Incident Response Team (CSIRT) can, if needed, validate the automatically generated CTI reports.

To select samples for tests, we focused on common threats such as Malwares and Botnets. We also looked for datasets with previously verified analyses and with, at least in some cases, CTI available for validation. The datasets used are:

- **Ransomware PCAPs:** Sand-boxed ransomware samples from different malware families [38], with more than 200GB in PCAPs. Each PCAP is an instance related to one of these families. The number of samples varies by family. PCAPs contain network traffic and I/O operations.
- **Botnet PCAPs:** The IoT-BDA Botnet Analysis Dataset has 4077 unique IoT botnet samples. It includes ELF files for the botnet samples, captured behaviour on systemcalls and the PCAPs for the network traffic.

Berrueta et al. [38] created the malware samples in a scenario where it encrypts a shared directory (server and client), with additional traffic in some cases, such as DNS requests and communication with Command and Control (C&C) servers. Thus, intersections on TTPs is an expected consequence. Also, as a note, we show the validation without sub-techniques because available CTI for these samples do not include them, and we need to compare the generated ones with that.

2) Validation: To validate our approach, we compare two types of reports related to the same **samples** regarding the high-level CTI information they provide: **already available reports** extracted from CTI sources, and **reports our solution automatically generates**. It is important to note that the low availability of higher-level CTI (TTPs and above) on current reports is a known problem [11]. Consequently, we execute the validation with samples related to Cerber and REvil/Sodinokibi malware families, as they have at least some network-related high-level CTI available.

With our validation metrics, we evaluate two factors in the generated CTI in regards to the TTPs it contains: *Correctness* and (relative) *Completeness*. Relative Completeness shows that the automated CTI report contains at least those TTPs included in the already available (manually created) CTI. It means that the automated analysis is sufficiently comprehensive to provide generated reports with at least as much information as existing ones. Correctness means that provided TTPs are sound, which means they are present in the attack. We assume TTPs in existing CTI reports are sound ones. Any additional TTPs we find provide an improvement compared to the existing reports. Provided, of course, that they are sound ones. We validate those cases using the mapped alert and the mapped CTI.

3) New CTI Reports: After validating with pre-existing CTI, we run our approach on the remaining datasets, creating high-level CTI for them as well. As there is nothing to compare to, the tests are more straightforward. We check whether the solution can identify and represent malicious chains by outputting their resulting high-level CTI as structured reports over a broad set of scenarios.

B. Results

In this section, we present the results of our tests.

1) Validation: Figure 5 and Figure 6 show the results of applying the solution to REvil and Cerber respectively. The header line denotes an existing report, with its TTPs marked as squares. Other lines represent samples used in our validation, and TTPs generated automatically by our methodology as circles, with a summary of additional TTPs on the right.

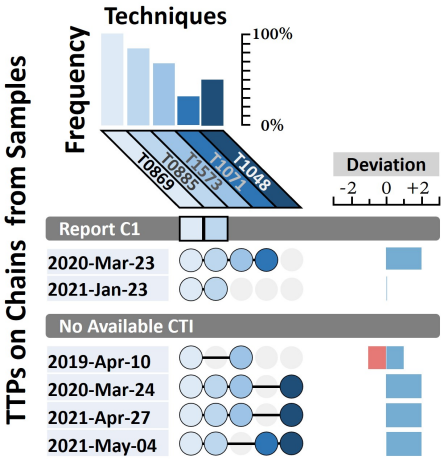


Fig. 5: Results for Validation with REvil

(a) Completeness: Figure 5 shows the results for the REvil samples. The validation resulted in five different TTPs detected across the malware instances. The automated reports generated for (*2020-Mar-23* and *2021-Jan-23*) achieved maximum completeness. They produce at least all TTPs expected from existing reports. The same happens with Cerber samples, as seen in Figure 6. The report automatically generated for the sample *2016-Oct-04* matches the single TTP in existing reports about that sample [38]. It is interesting to note that, for both families, there is a match of the techniques detected even in the samples not related to the original report, with the sole exception of a single deviation on REvil's *2019-Apr-10*.

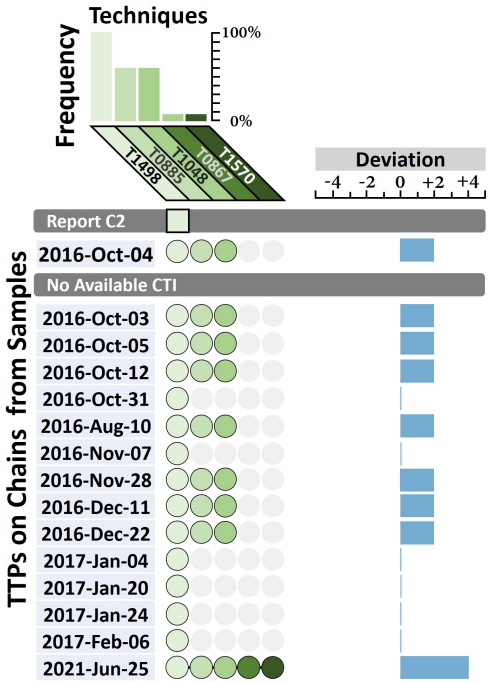
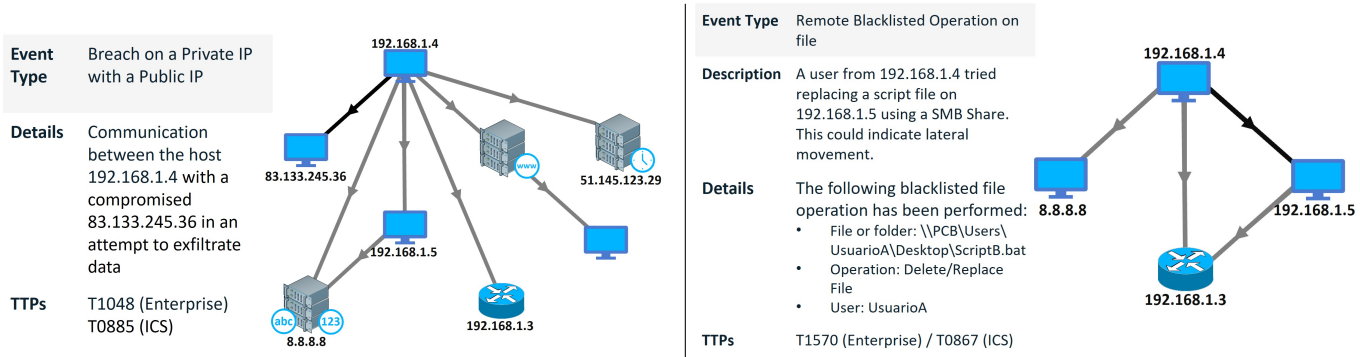


Fig. 6: Results for Validation with Cerber



(a) Additional TTPs on REvil Samples (TTP also valid for Cerber)

(b) Additional TTPs on Cerber Samples

Fig. 7: Results for Validation of Additional TTPs with REvil and Cerber Samples

(b) Correctness: We validate the additional TTPs for correctness and we give some example analysis on Figure 7 and Figure 8. By asserting that the additional TTPs automatically generated are sound, we guarantee an improvement over existing manual reports.

As seen on Figure 5, the report automatically generated from the sample *2020-Mar-23* gives two additional TTPs, both of which are also encountered on other samples of the same family. In this specific case, these were alerts about C&C traffic concealed in SSLv3 communication (T1573), followed by alerts of application layer protocols used while communicating with known malicious IPs (T1071). Figure 8 shows both cases in more detail.

Figure 7a shows an excerpt of the chain from the sample *2020-Mar-24*. The representation on the left side summarises the automated analysis given as output during the report creation. In this graph, pseudo-hosts represent Multicast and Broadcast to simplify the visualisation.

The report generated for *2020-Mar-24* had the additional TTPs T1048 (Exfiltration Over Alternative Protocol) and T0885 (Commonly Used Port) mapped from an alert with Type *Breach on a Private IP with a Public IP in an attempt to exfiltrate data*, between the host 192.168.1.4 with an external IP 83.133.245.36, also verified to be correct. The TTPs T1048 and T0885 are associated with the highlighted alert. In this case, a communication between 192.168.1.4 and an external compromised host 83.133.245.36[1] was detected[2].

Figure 7a also shows informational alerts from the extended chain with communications with the external IPs 8.8.8.8 and 51.145.123.29. Respectively, a known DNS server and a known Time server. These events did not contain unusual information.

For Cerber, there were comparable cases. The automatically generated report for the sample *2016-Oct-04* has additional TTPs similar to the ones in Figure 7a, yielding T0885 and T1048. Seven other samples had these same TTPs.

The report for the sample *2021-Jun-25* had the same TTPs as other samples before, but it included T1570 and T0867 as additional ones. Figure 7b summarises the analysis in the highlighted communication. These TTPs are the result of alerts related to many communications where the host 192.168.1.4 tries to delete or replace files on host 192.168.1.5. In one of such example, it tried to delete the BAT file *ScriptB.bat* in the desktop folder of the destination host through a user UsuarioA. This is actually the example shown previously in Figure 3 in the highlighted communication.

An interesting information noticed in these validations is that the automatically generated reports that include the TTP T1498 (Network Denial of Service) match the addition of Distributed Denial of Service (DDoS)[3] capabilities to Cerber in 2016 [38].

From this validation, we conclude that additional TTPs on all the samples for these malware families achieve correctness.

[1]https://otx.alienvault.com/indicator/ip/83.133.245.36
[2]https://www.joesandbox.com/analysis/803486#iocs

[3]https://www.trendmicro.com/vinfo/pl/security/news/cybercrime-and-digital-threats/cryptxxx-and-cerber-ransomware-get-major-updates

Event Type	Severity	Source IP...	Destination ...	Port
Use of insecure SSL protocol version (SSLv3)	High	192.168.1.4	79.137.75.185	443
Use of insecure SSL protocol version (SSLv3)	High	192.168.1.4	159.69.83.114	443
Blacklisted IP address	Critical	192.168.1.4	198.54.117.198	443
Blacklisted IP address	Critical	192.168.1.4	198.54.117.197	443

Fig. 8: REvil using insecure encryption, and communicating with blacklisted IPs

TABLE I: TTPs on the CTI Reports created for each family

Family	Set	Samples(#)	Size(B)	Alerts(#)	TTPs
Bart	UN	1	3GB	9	T0814, T1498, T0869, T1071
Cerber	UN	15	33GB	19046	T1498, T0885, T1048, T0867, T1570
Crylock	UN	1	578MB	9	T0881, T1071, T0869
Crysis	UN	12	12GB	110	T0881, T1498
Cryptomix	UN	4	16GB	20	T1071, T1573, T1570, T0869, T0885
DmaLocker	UN	1	821MB	9	T0885, T1048, T0869, T1071
Gafgyt	BDA	123	624MB	712	T0859, T1078, T0840, T1595, T1046
Hajime	BDA	1047	225MB	517	T0814, T0840, T1595, T0869, T0885
IRCbot	BDA	6	28MB	55	T1595, T1078, T1048
Locky	UN	10	42GB	761	T0886, T0867, T1570, T1573
Maktub	UN	1	347MB	30	T0867, T1570, T0885, T1048
Mirai	BDA	1884	6.6GB	1863	T0840, T0869, T0859, T0814, T0866
Revil	UN	6	14GB	4696	T0869, T0885, T1572, T1071, T1048
Tsunami	BDA	10	1.8MB	8	T0869, T0885
Wannacry	UN	4	19GB	1384	T1210, T0869, T1071, T0885
Others	UN	1	-	04-10	T0885, T1048

2) New CTI Reports.: We then ran the solution on the remaining families, adding the IoT-BDA dataset in these tests. The results can be seen on Table I.

In some cases with families with a single sample, the NIDS detected only TTPs related to the simulation process itself, which are linked to the way the ransomware samples were created by the original authors [38]. This was expected, as we explained on Subsection V-A. This includes the families Aleta, Bitpaymer, Crylock, Crypmix, Cryptfile2, and a few others. Thus, these families only had two TTPs which are specific to their simulation rather than to the families themselves, and between four and ten alerts. They are summarised as the "Others" group in the last line of Table I.

For the other cases, we detected multiple TTPs. In known families such as Cerber, we can confirm they include expected TTPs. The same is valid for REvil as we have seen before. For the other families, we consider detecting any network related TTPs as an improvement, as they did not have any in previous reports.

VI. Contribution in the Gap Analysis

The following was necessary to provide automated generation of machine-readable CTI about incidents involving multiple entities: (1) Work with network alerts from different hosts. Then, (2) map alerts correlated by a NIDS to relevant attack methods (TTPs). (3) Output a machine-readable CTI report that is shareable, and possibly a human-readable one. (4) Allow the validation of the reports if necessary.

Table II presents the requirements fulfilled by the most relevant works from Section II. Scarabeo et al. [26], Navarro et al. [27] and Leite et al. [11] map some CTI with alerts. None of these generate reports from the information used, as they focus only on local analysis. Landauer et al. [28] do generate CTI reports, but only for host-based information.

Based on this analysis and with our results, we assert that our work fills this gap. It can automatically generate shareable CTI reports from network incidents about multiple hosts.

	Multihost Alerts	TTP Map	Generate CTI	Validation
Scarabeo et al. [26]	✓	✓	-	-
Navarro et al. [27]	✓	✓	-	-
Leite et al. [11]	✓	✓	-	-
Landauer et al. [28]	-	✓	✓	✓
Our Solution	✓	✓	✓	✓

TABLE II: Contribution on Gap Analysis

VII. Conclusions

In this paper, we proposed a new approach to automatically generate CTI from network incidents by extracting alert chains from the network, mapping them to TTPs, and exporting them as CTI, thus addressing sub-question SQ1.

In the validation experiment, we confirmed that the automated reports match and surpass previously available ones created manually. We then validated the correctness of additional CTI generated, showing that analysts can then verify reports and prioritise information that might be more useful when needed. Everything was done with the proposed structure, fulfilling SQ2 and answering our main research question RQ by consequence.

For future work, we propose addressing some limitations. Focusing on being compatible with multiple correlation techniques makes it so that False Positive/Negative Rates are directly related to the capabilities of the NIDS. Although it would require analysing and comparing different techniques, exploring a more self-contained (and thus closed) approach that includes a fixed correlation technique could benefit the precision of the output CTI. The methodology also depends

on a good map between alerts and TTPs. Creating these maps was one of the most demanding (and thus costly) tasks. It requires specialised knowledge, time for iterations, and possibly multiple knowledgeable analysts. It could be interesting to include mapping of TTPs with event types (description) in common databases, such as ICS-CERT and NVD through community efforts. This information, likely already known by creators of alert types, would benefit any analyst using it.

REFERENCES

[1] S. Bromander, M. Swimmer, L. P. Muller, A. Jøsang, M. Eian, G. Skjøtskift, and F. Borg, "Investigating Sharing of Cyber Threat Intelligence and Proposing A New Data Model for Enabling Automation in Knowledge Representation and Exchange," *Digital Threats: Research and Practice*, vol. 3, no. 1, pp. 6:1–6:22, Oct. 2021.

[2] D. Schlette, M. Caselli, and G. Pernul, "A Comparative Study on Cyber Threat Intelligence: The Security Incident Response Perspective," *IEEE Communications Surveys Tutorials*, vol. 23, no. 4, pp. 2525–2556, 2021, conference Name: IEEE Communications Surveys Tutorials.

[3] D. Schlette, "Cyber Threat Intelligence," in *Encyclopedia of Cryptography, Security and Privacy*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2021, pp. 1–3.

[4] P. Nespoli, D. Papamartzivanos, F. Gómez Mármol, and G. Kambourakis, "Optimal Countermeasures Selection Against Cyber Attacks: A Comprehensive Survey on Reaction Frameworks," *IEEE Communications Surveys Tutorials*, vol. 20, no. 2, pp. 1361–1396, 2018.

[5] A. Groenewegen and J. Janssen, *TheHive Project: The maturity of an open-source Security Incident Response platform*, Jun. 2021.

[6] The National Council of ISACs, "NCI Principles on Mandatory Reporting," 2021.

[7] European Network and Information Security Agency, *Information sharing and analysis centres (ISACs): cooperative models.* LU: Publications Office, 2017. [Online]. Available: https://data.europa.eu/doi/10.2824/549292

[8] Cyber Threat Alliance, "Cyber Incident Reporting Framework," Nov. 2022. [Online]. Available: https://www.cyberthreatalliance.org/cyber-incident-reporting-framework/

[9] Department of Homeland Security, "Automated Indicator Sharing | CISA." [Online]. Available: https://www.cisa.gov/ais

[10] National Cyber Security Centre, "Cyber Security Information Sharing Partnership - CISP." [Online]. Available: https://www.ncsc.gov.uk/section/keep-up-to-date/cisp

[11] C. Leite, J. den Hartog, D. dos Santos, and E. Costante, "Actionable Cyber Threat Intelligence for Automated Incident Response," in *Secure IT Systems*, ser. Lecture Notes in Computer Science. Cham: Springer International Publishing, Dec. 2022, pp. 368–385.

[12] M. F. Haque and R. Krishnan, "Toward Automated Cyber Defense with Secure Sharing of Structured Cyber Threat Intelligence," *Information Systems Frontiers*, vol. 23, no. 4, pp. 883–896, Aug. 2021.

[13] I. Kotenko, D. Gaifulina, and I. Zelichenok, "Systematic Literature Review of Security Event Correlation Methods," *IEEE Access*, vol. 10, pp. 43 387–43 420, 2022, conference Name: IEEE Access.

[14] A. Tabiban, H. Zhao, Y. Jarraya, M. Pourzandi, and L. Wang, "VinciDecoder : Automatically Interpreting Provenance Graphs into Textual Forensic Reports with Application to OpenStack," 2022.

[15] Z. Li, J. Zeng, Y. Chen, and Z. Liang, "AttacKG: Constructing Technique Knowledge Graph from Cyber Threat Intelligence Reports," May 2022, arXiv:2111.07093 [cs].

[16] I. Sarhan and M. Spruit, "Open-CyKG: An Open Cyber Threat Intelligence Knowledge Graph," *Knowledge-Based Systems*, vol. 233, p. 107524, Dec. 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705121007863

[17] Y. Gao, X. LI, H. PENG, B. Fang, and P. Yu, "HinCTI: A Cyber Threat Intelligence Modeling and Identification System Based on Heterogeneous Information Network," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–1, 2020, conference Name: IEEE Transactions on Knowledge and Data Engineering.

[18] G. Husari, E. Al-Shaer, M. Ahmed, B. Chu, and X. Niu, "TTPDrill: Automatic and Accurate Extraction of Threat Actions from Unstructured Text of CTI Sources," *Proceedings of the 33rd Annual Computer Security Applications Conference*, 2017.

[19] T. Sun, P. Yang, M. Li, and S. Liao, "An Automatic Generation Approach of the Cyber Threat Intelligence Records Based on Multi-Source Information Fusion," *Future Internet*, vol. 13, no. 2, p. 40, Feb. 2021, number: 2 Publisher: Multidisciplinary Digital Publishing Institute. [Online]. Available: https://www.mdpi.com/1999-5903/13/2/40

[20] V. Legoy, M. Caselli, C. Seifert, and A. Peter, "Automated Retrieval of ATT&CK Tactics and Techniques for Cyber Threat Reports," 2020.

[21] U. Noor, Z. Anwar, T. Amjad, and K.-K. R. Choo, "A machine learning-based FinTech cyber threat attribution framework using high-level indicators of compromise," *Future Generation Computer Systems*, vol. 96, pp. 227–242, Jul. 2019.

[22] A. Modi, Z. Sun, A. Panwar, T. Khairnar, Z. Zhao, A. Doupe, G.-J. Ahn, and P. Black, "Towards Automated Threat Intelligence Fusion," in *2016 IEEE 2nd International Conference on Collaboration and Internet Computing (CIC)*. IEEE, Nov. 2016, pp. 408–416.

[23] E. Kim, K. Kim, D. Shin, B. Jin, and H. Kim, "CyTIME: Cyber Threat Intelligence ManagEment framework for automatically generating security rules," in *Proceedings of the 13th International Conference on Future Internet Technologies*. Seoul Republic of Korea: ACM, Jun. 2018, pp. 1–5.

[24] A. Berndt and J. Ophoff, "Exploring the Value of a Cyber Threat Intelligence Function in an Organization," in *Information Security Education. Information Security in Action*, ser. IFIP Advances in Information and Communication Technology. Cham: Springer International Publishing, 2020, pp. 96–109.

[25] R. Brown and R. M. Lee, "2021 SANS Cyber Threat Intelligence Survey," p. 19, 2021.

[26] N. Scarabeo, B. C. Fung, and R. H. Khokhar, "Mining known attack patterns from security-related events," *PeerJ Computer Science*, vol. 1, p. e25, Oct. 2015. [Online]. Available: https://peerj.com/articles/cs-25

[27] J. Navarro, V. Legrand, S. Lagraa, J. François, A. Lahmadi, G. d. Santis, O. Festor, N. Lammari, F. Hamdi, A. Deruyver, Q. Goux, M. Allard, and P. Parrend, "HuMa: A Multi-layer Framework for Threat Analysis in a Heterogeneous Log Environment," Oct. 2017, p. 144. [Online]. Available: https://inria.hal.science/hal-02460272

[28] M. Landauer, M. Wurzenberger, F. Skopik, G. Settanni, and P. Filzmoser, "Dynamic log file analysis: An unsupervised cluster evolution approach for anomaly detection," *Computers & Security*, vol. 79, pp. 94–116, Nov. 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167404818306333

[29] B. E. Strom, A. Applebaum, D. P. Miller, K. C. Nickels, A. G. Pennington, and C. B. Thomas, "MITRE ATT&CK: Design and Philosophy," MITRE, Tech. Rep., 2020.

[30] B. Zhu and A. A. Ghorbani, "Alert Correlation for Extracting Attack Strategies," 2006.

[31] D. Gunter, "Hunting with Rigor: Quantifying the Breadth, Depth and Threat Intelligence Coverage of a Threat Hunt in Industrial Control System Environments," p. 21, 2018.

[32] S. Roschke, F. Cheng, and C. Meinel, "A New Alert Correlation Algorithm Based on Attack Graph," in *Computational Intelligence in Security for Information Systems*, ser. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2011, pp. 58–67.

[33] MITRE, "MITRE ATT&CK Techniques Mapped to Data Sources," Tech. Rep., 2019. [Online]. Available: https://attack.mitre.org/docs/attack_roadmap_2019.pdf

[34] N. Naik, P. Jenkins, P. Grace, and J. Song, "Comparing Attack Models for IT Systems: Lockheed Martin's Cyber Kill Chain, MITRE ATT&CK Framework and Diamond Model," in *2022 IEEE International Symposium on Systems Engineering (ISSE)*, Oct. 2022, pp. 1–7.

[35] J. J. Treinen and R. Thurimella, "Application of the PageRank Algorithm to Alarm Graphs," in *Information and Communications Security*, S. Qing, H. Imai, and G. Wang, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, vol. 4861, pp. 480–494, series Title: Lecture Notes in Computer Science.

[36] C. Sauerwein, D. Fischer, M. Rubsamen, G. Rosenberger, D. Stelzer, and R. Breu, "From Threat Data to Actionable Intelligence: An Exploratory Analysis of the Intelligence Cycle Implementation in Cyber Threat Intelligence Sharing Platforms," in *The 16th International Conference on Availability, Reliability and Security*. ACM, Aug. 2021, pp. 1–9.

[37] HybridAnalysis, "Hybrid Analysis." [Online]. Available: https://www.hybrid-analysis.com/

[38] E. Berrueta, D. Morato, E. Magaña, and M. Izal, "Open Repository for the Evaluation of Ransomware Detection Tools," *IEEE Access*, vol. 8, pp. 65 658–65 669, 2020.