Mining Attributed Graphs for Threat Intelligence

Hugo Gascon
Technische Universität Braunschweig

Bernd Grobauer
Siemens AG

Thomas Schreck
Siemens AG

Lukas Rist
Symantec Corporation

Daniel Arp
Technische Universität Braunschweig

Konrad Rieck
Technische Universität Braunschweig

ABSTRACT

Understanding and fending off attack campaigns against organizations, companies and individuals, has become a global struggle. As today's threat actors become more determined and organized, isolated efforts to detect and reveal threats are no longer effective. Although challenging, this situation can be significantly changed if information about security incidents is collected, shared and analyzed across organizations. To this end, different exchange data formats such as STIX, CyBOX, or IODEF have been recently proposed and numerous CERTs are adopting these *threat intelligence* standards to share tactical and technical threat insights. However, managing, analyzing and correlating the vast amount of data available from different sources to identify relevant attack patterns still remains an open problem.

In this paper we present MANTIS, a platform for threat intelligence that enables the unified analysis of different standards and the correlation of threat data trough a novel type-agnostic similarity algorithm based on attributed graphs. Its unified representation allows the security analyst to discover similar and related threats by linking patterns shared between seemingly unrelated attack campaigns through queries of different complexity. We evaluate the performance of MANTIS as an information retrieval system for threat intelligence in different experiments. In an evaluation with over 14,000 CyBOX objects, the platform enables retrieving relevant threat reports with a mean average precision of 80%, given only a single object from an incident, such as a file or an HTTP request. We further illustrate the performance of this analysis in two case studies with the attack campaigns *Stuxnet* and *Regin*.

Keywords

Threat Intelligence; Advanced Persistent Threat; Graph Mining; Information Retrieval

CODASPY '17, March 22–24, 2017, Scottsdale, AZ, USA.

ISBN 978-1-4503-4523-1/17/03…$15.00

DOI: http://dx.doi.org/10.1145/3029806.3029811

1. INTRODUCTION

Targeted attacks pose a serious threat to the security of individuals, companies and organizations. In contrast to regular malware aiming at widespread infections, these campaigns are tailored to maximize the impact in the systems and networks of their victims. In many occasions, such campaigns are conducted by experienced and even government-sponsored actors that are specialized in industrial espionage and invest considerable resources in the preparation of their attacks. For example, according to the eye-opening investigation of the security company Mandiant, a group of attackers code-named "APT1" successfully infiltrated 141 companies over a period of 7 years and obtained access to several terabytes of company data [24]. The group was believed to include several developers and operators, likely with the support of a nation-state actor. Since then, reports about newly disclosed operations tailored against companies, governments and individuals have become increasingly common in the media [e.g. 32, 36, 37].

Unfortunately, the detection and analysis of attack campaigns is a daunting task: First, due to the focused operation of the campaigns, only few traces of the attackers are available for forensic investigation. Second, the employed malware often makes use of novel exploits and infiltration techniques. As a consequence, conventional security defenses such as intrusion detection systems and anti-virus scanners fail frequently to spot these type of threats. Specially because detection patterns become available only with significant delay, if at all. It has become evident then, that isolated efforts to detect attack campaigns within companies and organizations are mostly ineffective against organized threat actors.

As a remedy, security research has recently started to explore means for collecting, sharing and analyzing threat information across organizations—evidence-based knowledge referred to as *threat intelligence* [e.g., 3, 11, 18, 27]. As part of this process, different exchange formats have been proposed to provide a standardized way for describing security incidents, forensic traces and observations related to attack campaigns. Examples of these formats are STIX [1], IODEF [9] and OpenIOC [23], which are gradually adopted by national and enterprise CERTs in combination with commercial and open source databases for storing knowledge about ongoing attacks such as *Alien Vault's Open Threat Exchange* [28] or the *Collective Intelligence Framework* [6].

However, collecting and sharing information alone is not sufficient for mitigating the threat of attack campaigns. Although such threat intelligence platforms enable searching for indicators of compromise that exactly match a query, the actual crux is to correlate the vast amount of available data and pinpoint similar characteristics of novel campaigns that can help eliminating existing infections as well as craft detection patterns more efficiently.

In this paper, we present MANTIS, an analysis platform that enables the aggregation and correlation of threat data into a unified representation based on attributed graphs. In particular, the platform is able to merge information from different exchange formats, solving the problem of analysing data contained in heterogeneous or overlapping standards. Furthermore, different threat objects that are typically analysed independently are correlated through a data type-agnostic representation. Such an approach allows unveiling high-level relations not visible within individual threat reports and linking unconventional patterns shared between seemingly unrelated attack campaigns.

At the core of our platform lies a novel graph-based similarity algorithm that allows discovering similarities between threat data objects at different levels of granularity. This analysis allows a security analyst to search the attributed graphs for threats related to individual observations—similar in spirit to a search engine. For example, given an object from a security incident, such as a suspicious file or an HTTP request, the platform can identify related nodes in the graphs and traverse them to the corresponding threat reports, ultimately returning information about the underlying attack campaign. In addition, MANTIS supports authoring reports for new incidents that can be used for searching and correlating existing information, as well as extending existing threat data with new insights.

We evaluate the utility of MANTIS as an information retrieval system for threat intelligence in a quantitative and qualitative fashion. To this end, we make use of a large data set of malware observed in the wild and collected by a security vendor at the end-point systems of different companies and organizations. We base our evaluation on the threat reports created during the analysis of such samples.

As a result, we show how given an object from a security incident, our platform is able to retrieve associated data to the corresponding malware with a mean average precision of 80% in a set of 14,000 standardized threat objects. This means that 4 out of 5 results returned to the security analyst are relevant to her query. We further illustrate the performance of this analysis in two case studies based on threat intelligence from highly targeted attack campaigns: *Stuxnet*, the well-known joint endeavour of several west nations to sabotage Iran's nuclear program and *Regin*, a sophisticated espionage tool allegedly sponsored by a state-nation and distributed worldwide to selected individuals and organizations.

To the best of our knowledge, MANTIS is the first practical solution for performing similarity-based analysis of multi-format and structured data for threat intelligence. While the platform requires the interplay with other techniques for stopping attack campaigns, the analysis and query capabilities alone already provide a valuable tool for assessing the impact of security incidents. MANTIS is available as an open-source project and is currently used at a large CERT for managing threat data in day-to-day business.

In summary, we make the following contributions:

- *Unified representation of threat intelligence reports.* We present an open-source platform for threat intelligence that merges different standard exchange formats and provides a unified representation of threat reports as attributed graphs.
- *Similarity analysis of threats.* We introduce a similarity algorithm for attributed graphs that enables uncovering relations between threats at different levels of granularity.
- *Information retrieval for threat intelligence.* By incorporating the similarity analysis into our platform, we devise an information retrieval system that is capable of retrieving related reports given individual observations from security incidents.

The rest of the paper is organized as follows: we introduce the concept of threat intelligence and its standards in Section 2. We then proceed to present our system for analysis and retrieval of threat data in Section 3. We evaluate its effectiveness with real-world threat data in Section 5 and discuss its limitations in Section 6. Related work is discussed in Section 7 and Section 8 concludes the paper.

```
<stix:STIX_Package (...) id="package-37e">
  <stix:STIX_Header>
    <stix:Title>APT1</stix:Title>
    <stix:Description>
      This package contains the IOCs referenced
      in Appendix G of the APT1 report.
    </stix:Description>
  </stix:STIX_Header>
  <stix:Observables>
    <cybox:Observable id="Observable-9ba">
      <cybox:Object id="URI-9ba">
        <cybox:Properties type="URL">
          <URIObj:Value condition="contains">
            /mci.jpg
          </URIObj:Value>
        </cybox:Properties>
      </cybox:Object>
    </cybox:Observable>
    <cybox:Observable id="Observable-2b2">
      <cybox:Object id="File-2b2">
        <cybox:Properties type="File">
          <FileObj:Name>gdocs.exe</FileObj:Name>
          <FileObj:Extension>exe</FileObj:Extension>
          <FileObj:Size>261822</FileObj:Size>
          <FileObj:Attributed_List>
            <cybox:Object condition="contains">
              v1.0 No Doubt to Hack You, Writed
              by UglyGorilla, 06/29/2007
            </cybox:Object>
          </FileObj:Attributed_List>
        </cybox:Properties>
      </cybox:Object>
    </cybox:Observable>
  </stix:Observables>
  <stix:Indicators>
    <stix:Indicator id="Indicator-a42">
      <indicator:Title>
        MANITSME
      </indicator:Title>
      <indicator:Description>
        This family of malware will beacon out at
        random intervals to the remote attacker.
        The attacker can run programs, execute
        arbitrary commands, and easily upload and
        download files.
      </indicator:Description>
      <indicator:Type>Backdoor</indicator:Type>
      <indicator:Observable id="Observable-a42">
        <cybox:Observable_Composition operator="OR">
          <cybox:Observable idref="Observable-9ba">
          </cybox:Observable>
          <cybox:Observable idref="Observable-2b2">
          </cybox:Observable>
        </cybox:Observable_Composition>
      </indicator:Observable>
      <indicator:Related_Campaigns>
        <indicator:Related_Campaign>
          <stixCommon:Campaign idref="Campaign-a58"/>
        </indicator:Related_Campaign>
      </indicator:Related_Campaigns>
    </stix:Indicator>
      ...
      ...
      ...
      ...
      ...
  </stix:Indicators>
</stix:STIX_Package>
```

Figure 1: Exemplary STIX package for the "APT1" report by Mandiant [24]. Note that several identifiers and XML elements have been simplified for presentation.

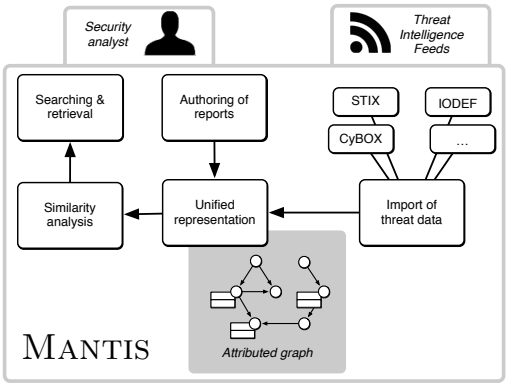


Figure 2: Schematic overview of the MANTIS architecture.

2. THREAT INTELLIGENCE

Companies and organizations dealing with security-sensitive data usually employ different security measures for protecting their infrastructure, including systematically monitoring network and host events. While this monitored data can be searched for security incidents on a regular basis, appropriate detection and search patterns are only available for known threats, leaving infrastructure vulnerable to novel and unknown attack campaigns. This situation, however, can be significantly changed if information about incidents, is collected, shared and analyzed across organizations. Although this approach may not be sufficient for spotting extremely focused attacks, it enables hunting down threat actors that re-use or gradually evolve their techniques and strategies.

However, information regarding security incidents, related observations, and threat actors is very heterogeneous and difficult to transmit without a lack of context. In order to overcome this problem, different standard formats have been recently proposed to provide a structured representation of threat data that can be easily shared and processed. These standardised but diverse threat insights constitutes what has been known as *threat intelligence*. Examples of these standards are *IODEF*, developed by members of the IETF [9], *OpenIOC*, implemented by Mandiant in many of its products [23], and *STIX* with its associated family of formats, like *CyBOX* or *MAEC* [1]. In particular, the *STIX* standard is currently leading the adoption by national and enterprise CERTs. In the following, we briefly cover its design as an illustrative example of the structured representations implemented by all of the mentioned threat intelligence standards.

The STIX standard comprises a family of XML schemes whose development is driven by the security community under supervision of the MITRE Corporation. The individual STIX formats and constructs allow to describe numerous types of threat information in a structured way and for different use cases. For example, observations related to threats can be described as *Observables*, ranging from registry keys and file names to network addresses and strings in URLs. These Observables can be combined with logical operators to form *Indicators* that reflect and describe concrete threats. Other constructs include representations for *Incidents*, *Courses of Action*, *Attack Campaigns* and *Threat Actors*. A detailed description of the different constructs is provided in the STIX specification [1].

As an example, let us consider the STIX package shown in Figure 1 which covers a tiny and simplified fragment of the indicators for the "APT1" campaign. This campaign was uncovered in February 2013 and comprised a series of targeted attacks against several companies and organizations [24]. Some common constructs of the STIX standard can be seen in the example: An Observable matching the content of a URI (line 10–18), another Observable corresponding to a particular file (line 19–33), and an Indicator combining the two (line 36–61) that describes the malware family and references the underlying attack campaign. Note that although not included here, the original report in OpenIOC format covers over 3,000 Observables and 40 different Indicators for the attack campaign.

The use of threat intelligence standards allows to share and process a large amount of complex and enriched threat data in a standard and machine readable format. This has encouraged some companies with a large distributed infrastructure and a global view of the threat landscape to aggregate feeds that are made available to smaller organizations. However, the information received through these sources is highly heterogeneous and still needs to be put into context by the analyst. In our work, we aim at making this analysis much more efficient by providing a platform that integrates different standards into a unified representation and allows for exploring and searching structured threat data for relevant information.

3. THE MANTIS FRAMEWORK

As a first step for analyzing and understanding attack campaigns, we present MANTIS, an analysis platform for storing, authoring and managing threat data. The platform implements support for several common threat intelligence standards, including STIX and OpenIOC, two of the standards with the largest adoption in the security community. To support this adoption and encourage further research, MANTIS is available as an open-source project[1] and readily applicable for experimenting with threat data at organizations and CERTs and the implementation of new importers for additional standards.

To provide a flexible and platform-independent design, MANTIS is structured as a set of *Django* applications. Figure 2 shows a schematic view of its architecture. In the typical use case, the security analyst documents the findings of an investigation using the authoring interface, while at the same time accesses related information about already documented threats through the retrieval interface. Both interfaces provide different views for managing the creation and the collaborative maintenance of threat reports. Additionally, the platform supports receiving data feeds in different formats from other tools, organizations and security companies. The data contained in these feeds is jointly stored with authored reports and thereby enables an analyst to document her findings in the context of already known threats and attack campaigns.

3.1 Unified Data Model

To provide a joint view on the threat data collected, MANTIS expresses the different XML standards as directed graphs and links together constructs describing the same type of information.

Table 1: Example of flattened facts for an Observable.

Id	Fact term (key)	Fact value
f_1	Properties/File_Name	`gdocs.exe`
f_2	Properties/File_Extension	`exe`
f_3	Properties/Size_In_Bytes	`261822`
f_4	Properties/File_Attributed_List/Object@cond...	`Contains`
f_5	Properties/File_Attributed_List/Object	`v1.0 No Doub...`

As a result, related data describing campaigns at different levels, such as generic attack strategies and concrete malicious payloads, are merged into a single view and can be accessed by simply traversing the edges of the graphs.

[1] MANTIS— https://github.com/siemens/django-mantis

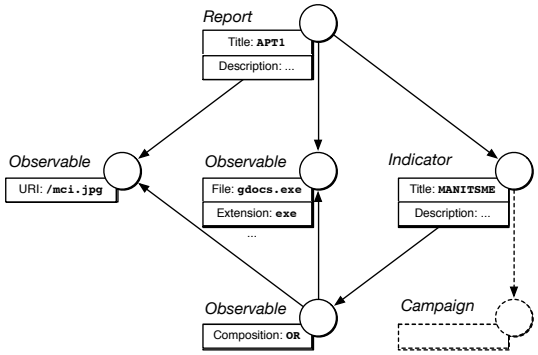


Figure 3: Attributed graph for STIX package in Figure 1.

Formally, we define this directed graph as a tuple $G = (V, E, L)$, where each node $v \in V$ symbolizes a standard construct from an original XML document. Two nodes $v, u \in V$ are connected by a directed edge $(v, u) \in E$, if the construct corresponding to u is either contained or referenced by the construct represented by v. Moreover, we attach a list of facts $l \in L$ to each node. This enables us to store unstructured data in the graph, assigning a set of attributes to each node. Each list $l \in L$ has the form $l = (f_1, f_2, \ldots, f_n)$ where a fact f_i results from flattening the inner structure of a standard construct into facts of key-value pairs.

As an example of this unified representation, Figure 3 depicts the attributed graph that abstracts the relations between objects and data in the STIX report from Figure 1, including the two Observables, their composition and the corresponding Indicator. Note how several substructures have been flattened into facts, such as the title of the report or the URI pattern.

In addition Table 1 shows the complete list of flattened facts for the Observable at the center of Figure 3. Note that the flattening is conducted recursively and the fact terms are built using a hierarchical structure. This generic representation within the nodes of the graph will let us compare effectively different threat reports and traverse between objects even if their are of different type, such as from an observed URI pattern to the corresponding attack campaign.

Each fact value in the platform is stored exactly once and referenced from any object containing the fact. This de-duplication saves storage space and, more importantly, enables an efficient calculation of correlation based on fact equality. Thus, the analyst can retrieve all nodes related to particular facts with a single query, for example to get a listing of all executable files with a size of 261,822 bytes. However, while equality-based searches already provide a powerful instrument for mining the collected threat data, it is obvious that more complex relations cannot be uncovered by focusing on exact fact matching alone. In the following, we introduce our analysis method, which as part of MANTIS allows the analyst to perform similarity-based queries on top of its unified graph data model.

4. SIMILARITY ANALYSIS

When working with threat intelligence, a security analyst investigating an incident begins by documenting any suspicious findings. Then, the analyst wonders if such an event has been observed in the past and, specially if relevant documentation about the incident already exists.

However, obtaining an answer to this question is far from trivial. Threat reports can be large and heterogeneous and contain data that, without being identical, are linked to related events. For example, consider the case of several Observables containing HTTP requests of similar URLs. While being slightly different in the URI or host name, such objects may be associated to the same threat actor and thus should be retrievable to help investigating the new incident.

As a consequence, the analyst requires a method that can identify and retrieve similar objects regardless of their structure, size or content given a query object. Thus, we strive for a similarity-based search that is capable of identifying similar facts, nodes and subgraphs on top of the unified representation of MANTIS. In particular, we implement our approach in two steps: First, we draw on our unified representation and devise a method that enables non-exact matching based on *fingerprints* computed using the *simhash* algorithm (Section 4.1). Second, we implement a retrieval system to efficiently identify all fingerprints similar to a given query (Section 4.2).

4.1 Simhash Fingerprinting

To measure the similarity between arbitrary objects in our representation, we make use of the bag-of-words concept from the information retrieval field [30]. In its original form, this model is intended for text documents in order to obtain a numerical vector representation based on the words or phrases they contain. However, threat data is heterogeneous and may range from simple file names to code fragments and textual descriptions. Therefore, we employ *byte n-grams* to characterize the content of an object [8, 40]. This means that a fact f is represented by all byte strings of length n contained in the fact value. Similarly, a node v is characterized by all n-grams contained in its associated facts l and a subgraph rooted at a node u is represented by the n-grams of all nodes reachable from u.

While the extracted n-grams provide a versatile and generic representation of the underlying content, they are not suitable for an efficient analysis, as they require variable-size storage and cannot be compared in constant time. For example, if new data introduced into the platform contained previously unseen n-grams, the existing vector representation of the bag-of-n-grams model should be recomputed for all objects to accommodate the new n-grams. As a remedy, we employ the *simhash* algorithm introduced by Charikar [5], an approximation technique that maps an arbitrary set of objects to a fixed-bit fingerprint.

The simhash algorithm ensures that although each object is represented by a hash of its n-grams, similar objects have similar fingerprints. More specifically, the design of the algorithm guarantees that the Hamming distance [15] of fingerprints computed from similar objects is small. This property allow us to articulate the problem of finding a similar construct in MANTIS given an input query and its fingerprint F as the problem of finding those fingerprints that differ from F in at most b bits.

The algorithm proceeds as follows: First, each object is hashed to an m-bit value. Second, the bits at each position i in the hash values are counted, where a 1-bit is interpreted as +1 and 0-bit as -1. Finally, the resulting m count values are converted into an m-bit fingerprint by setting all positive counts to 1 and all negative counts to 0. In our setting we apply the simhash algorithm to compute m-bit fingerprints for the sets of n-grams associated with facts, nodes and subgraphs, where we set $n = 3$ and $m = 64$. Accordingly, the fingerprint F_f of a fact f is computed by

$$F_f = \text{simhash}(N(f))$$

where N is the set of n-grams contained in the fact value. Similarly, we compute the fingerprint F_v of a node v as

$$F_v = \text{simhash}\Big(\bigcup_{f \in l(v)} N(f) \Big)$$

where $l(v)$ is the list of facts associated with v, and arrive at the fingerprint F_g of a subgraph rooted at a node u by

$$F_g = \text{simhash}\Big(\bigcup_{v \in r(u)} \bigcup_{f \in l(v)} N(f) \Big)$$

where the auxiliary function $r(u)$ returns all nodes reachable from u. Figure 4 shows a complete example of this computation for a fact containing the value `/mci.jpg`.

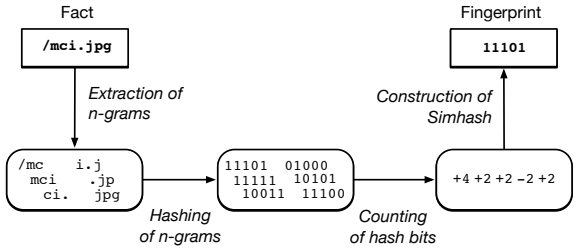


Figure 4: Computation of the simhash fingerprint of a fact.

The value is first represented by a set of 3-grams and then mapped to a set of 5-bit hash values. These values are finally aggregated to form the fingerprint $F_f = 11101$.

Note that n-grams are agnostic to the type of each fact, what results in determining similarity at a lexical level. This means that, in the same way as a search engine works, our method is not limited to measuring the similarity between constructs of the same type (e.g. two IP addresses), but between all possible types. This comparison enables to find relations in cases where standards are incorrectly filled or the types of data are unknown. For instance, a construct including a fact that describes the name of a file can be matched to a report including a description where this file is mentioned.

4.2 Hamming Distance-based Queries

When an large number of threat reports is loaded into the system the number of constructs that need to be analyzed can rapidly increase. For this reason, computing the Hamming distance between the query fingerprint and all queries in the platform can be computationally expensive.

As a remedy and to avoid precomputing the distance between all existing fingerprints at a maximum of b bits we follow the strategy proposed by Manku et al. [25]. In their approach, an index contains a series of *buckets* where each *bucket* has associated an integer p and a permutation of bits π. Each *bucket* is filled by first applying its permutation to all existing fingerprint and then sorting the resulting set of permuted fingerprints. Given a query fingerprint F and an integer b, we identify all permuted fingerprints in each *bucket* whose top p bits match the top p bits of $\pi(F)$. From these fingerprints, the ones that differ at most k bits from $\pi(F)$ are retrieved as result. Such approach can be completed in $O(p)$ and does not required the computation of a large distance matrix of fingerprints. For discussion on the optimal number of *buckets* and other implementation details, we refer the reader to the original description of the indexing approach introduced by Manku et al. [25].

For our particular application, we build three indexes: one for the fingerprints of individual facts, a second one for the fingerprints of nodes (i.e. individual constructs with their own semantics in the threat intelligence standard) and a third one for the fingerprints of subgraphs rooted at the different nodes. When a new report is imported into the system we first represent its data as an attributed graph. Then, we compute the fingerprints of its facts and constructs and add them to the corresponding index. When the analyst queries the system, the fingerprint of the query is computed and depending of its type, the results obtained from the corresponding index are retrieved. Moreover, retrieval results are sorted according to their Hamming distance and therefore their predicted relevance. This means that even in case that a query returns a large list of results, the analyst can rapidly identify the most relevant entries and keep conducting a focused investigation. In the following, we proceed to evaluate the efficacy of our approach using real-world threat data.

5. EVALUATION

In this section we evaluate our method for similarity-based searches through a quantitative and qualitative analysis. In particular, we first explore the performance of the system responses when every object and fact value is used as the input query introduced by the analyst. Second, we evaluate the results provided by the system in two specific scenarios. These involve threat data from the targeted and, therefore, more elusive *Stuxnet* and *Regin* attack campaigns.

5.1 Data Set

We consider for our evaluation a dataset of STIX packages automatically generated from malware samples collected in the wild by a security vendor in June 2015 at the end-point systems of different companies and organizations. The samples cover a wide range of malicious activity, including common botnets, backdoors and attack campaigns. Each sample is analyzed in a sandbox environment, where the results of the underlying static and dynamic analysis are automatically converted to CyBOX objects and grouped in STIX packages.

Table 2: Raw dataset indexed by MANTIS.

Standard	Construct	Size
STIX	STIX Package	2,621
STIX	Observable	7,282
STIX	Indicator	2,764
CybOX	Observable	255,941
CybOX	DNSQueryObject	2,583
CybOX	FileObject	12,334
CybOX	ProcessObject	17,914
CybOX	SemaphoreObject	244
CybOX	WinMutexObject	18,513
CybOX	WinRegistryObject	186,990
CybOX	WinThreadObject	22,347

Based on results provided by VirusTotal [39], we assign a label to each STIX package according to the hash of the analysed binary. As the names assigned to different malware families by AV vendors vary, we use a majority voting strategy and select those reports with a consensus of more than 5 vendors. The resulting 2,621 STIX reports are then loaded into MANTIS for analysis. Table 2 contains a summary of the constructs present in the original data.

Moreover, we take into considerations certain characteristics of the data that are relevant for the analysis: First, we exclude all objects and facts that are unsuitable for a similarity search, such as local timestamps, identifiers and hash sums, reducing the size of the attributed graphs to 14,987 individual nodes. Note that although these types of objects are not included to evaluate the algorithm, they are still in the system and are thus, searchable. Second, if several objects in one or several STIX reports contain the same value, the importer stores this value only once in MANTIS. As a result, nodes in the unified representation contain only references to their values, saving storage space if a certain value occurs more than once. The de-duplication performed by our platform, results in a total of 46,015 unique facts being stored in the system for similarity analysis.

5.2 Quantitative Evaluation

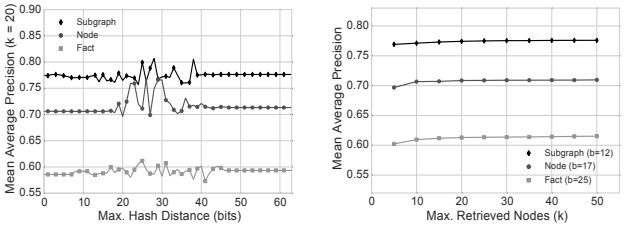
From the perspective of the security analyst, our platform resembles the operation of an information retrieval system: an analyst enters a query and retrieves a list of relevant nodes from the attributed graphs. So in essence, MANTIS functions like a search engine and its performance will be as good as the relevance of the results retrieved. Accordingly, in order to evaluate its performance qualitatively we make use of a metric that is widely employed to assess the performance of search algorithms: the *mean average precision* (MAP) [26]. The MAP averages the precision of a retrieval system over a set of queries Q for different numbers k of retrieved results. Formally, it is defined as

$$\text{MAP}(Q) = \frac{1}{|Q|} \sum_{j=1}^{|Q|} \frac{1}{m_j} \sum_{k=1}^{m_j} \text{Precision}(R_{jk}),$$

where Q is the set of queries, m_j the number of relevant nodes to the query $q_j \in Q$ and R_{jk} the top retrieved nodes for the query q_j up to the k-th relevant node. Moreover, we consider a node to be relevant, if it is associated with the same AV label as the object used as the query. For a single query, the average precision is the mean of the precision values obtained for the set of top k documents. This average value is then averaged over all possible queries [26], in our case all available facts, nodes or subgraphs.

To understand the intuition behind this metric we consider again the example of a search engine. The performance of a query is better when more relevant results are returned on the first page of the search engine, that is, we get a high precision value for the top k results [26]. Furthermore, the MAP score can be interpreted as the percentage of relevant objects in the returned results. For example, a MAP of 75% implies that 3 out of 4 returned results are relevant to the query.

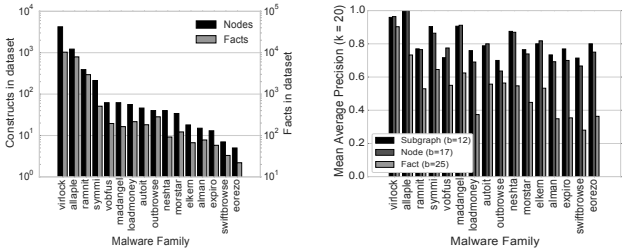
We compute the MAP for our analysis platform MANTIS by considering all facts, all nodes or all subgraphs reachable from a node as queries to the system. To gain further insights into the similarity analysis, we repeat the queries with different number of retrieved objects k and different numbers of bits to match between the fingerprints. The results of this experiment are presented in Figure 5, where the MAP is plotted for the different experimental setups. We note that the quality of the returned results depends on the complexity of the query. If subgraphs are used as query, MANTIS is able to achieve a MAP value of 80%, such that 4 out of 5 returned results are relevant and constitute similar threats. If the analyst enters only a node or a fact as query, the MAP decreases. However, even when entering only single facts, our platform attains a MAP of at least 50%, thus providing retrieval results where every second result matters. Moreover, our platform reaches a good MAP already at 15 retrieved items (Figure 5b) which is a reasonable amount of information to display on the first results page of the search interface.



(a) MAP vs. maximum Hamming distance between fingerprints.

(b) MAP vs. maximum number of retrieved results.

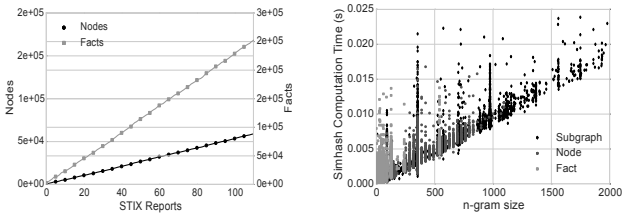
Figure 5: Mean average precision (MAP) for queries of different complexity.



(a) Total number of constructs and facts per family.

(b) MAP for each family with best b and k = 20.

Figure 6: Data distribution and mean average precision per malware family.



(a) Number of constructs and facts created per number of STIX reports imported in MANTIS.

(b) Computation Time of the Simhash Fingerprint vs. the number of *n-grams* in the object.

Figure 7: Scalability measurements respect to data size and fingerprint computation time.

As our dataset comprises a wide range of malware samples, we study how the diversity in the data affects the performance. Some samples, for instance, originate from small attack campaigns, while others are part of more common botnet and phishing activity. We evaluate then the results returned by the system when the queries belong to individual malware families. Figure 6a shows the amount of nodes and facts in each of the families. Note the logarithmic scale, that indicates a skewed distribution of samples per type of malware. Nonetheless and as shown in Figure 6b, the unbalance distribution has only a limited effect on the performance of our approach. Individual facts are retrieved with a MAP above 50% for most of the malware families, that is, every second returned result corresponds to the same malware family as the query. This is a remarkable result given that only individual facts, such as file names or URLs, are used to query the system.

Finally, scalability is another concern when designing an information retrieval system that is intended to accommodate large amounts of data. Figure 7a shows the evolution of the number of nodes and facts that need to be stored in the system per number of STIX reports imported. In both cases, a linear relation exist. As a result, we can expect our fingerprint indexes to also grow linearly with the number of imported reports. Moreover, every time the analyst introduces an individual fact or several facts as part of a construct, the fingerprint for each of them needs to be computed. As mentioned in Section 4.2, finding matching fingerprints for a query fingerprint F can be completed in $O(p)$, but the time computation of the fingerprint for the query object is directly related to its size. Figure 7b shows how even for large subgraphs with more than 2000 n-grams, the simhash fingerprint can be computed in less than 20 milliseconds with a linear dependency to the number of n-grams.

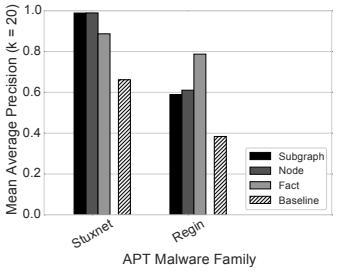
5.3 Qualitative Evaluation

To evaluate our approach qualitatively, we consider a small set of STIX packages from the *Stuxnet* and *Regin* attack campaigns. Such highly targeted APTs are characterized by disparate indicators of compromise and are typically very elusive to identify. Stuxnet, for

instance, which was initially discovered in 2010, is a sophisticated malware developed by west state-nations in order to sabotage the nuclear program of Iran. After remaining undetected for some time, its uncontrolled propagation through several attack vectors led to the identification of different variants in systems worldwide [22, 34]. The Regin trojan, on the other hand, is an advanced espionage tool that was used to surveil several companies and government entities including the European Council. Due to its stealth techniques, different variants of the malware remained unnoticed for several years until their discovery in 2011 [19, 35].

Thus, we evaluate the performance of our method when the analyst tries to retrieve such indicators from among more generic threat data. After loading a set of 31 and 10 STIX reports of the *Stuxnet* and *Regin* campaigns, respectively, we measure the mean average precision of the results when objects from these campaigns are used as queries. Additionally, we compare our method with the performance of searches based on exact fact matchings, as this type of retrieval strategy is the default approach used in threat intelligence engines and standard databases.

Table 3 shows the number of APT reports and objects loaded into the system in relation to the total number of objects present in the platform. Thus, in Figure 8, each column indicates the MAP over all queries when similarity is measured through a specific type of object hash. The objects from the *Stuxnet* APT are retrieved with a MAP over 85% for subgraph-, node- and fact-based queries, while in the case of the Regin APT, fact-based queries allow to retrieve correct results with a MAP of 79%. Unlike in the previous case, the complexity of larger queries like subgraphs and nodes, do not compensate in average for the small numbers of objects present in the database, making simpler fact queries more effective. Furthermore and as shown by the baseline performance, queries based on the similarity of objects in attributed graphs offer a more effective alternative than generic searches based on exact matchings of facts.



	Stuxnet	*Regin*	**Total in Database**
STIX Reports	31	10	2,662
Subgraphs	1,052	132	20,692
Nodes	557	76	15,620
Facts	4,785	1,395	52,195

Figure 8 & Table 3: MAP for query objects of APT families and comparison with baseline performance of standard search engines based on exact strings matching. Raw APT dataset indexed by MANTIS.

6. LIMITATIONS

The previous evaluation demonstrates the efficacy of MANTIS and our method to provide relevant similarity-based results for threat data queries. However, there exist certain limitations.

First, search results are always bounded to the data present in the system when a query is issued so an object cannot be retrieved if it has not been imported. Although an inherent limitation of every threat intelligence platform, this can be a disadvantage if an actor executes a highly targeted attack. In such a situation, it is likely that the attack will not be documented and become part of a repository or feed of threat data. For such events where no correlation is possible, reactive solutions like intrusion detection or behavioral analysis can be more effective to prevent and thwart the attack. Second, as other systems that aim at analyzing threat data, MANTIS is also subject to possible evasion attacks. For instance, an actor targeting an organization could use several types of attacks as part of a unique but large campaign. If such attacks are chosen to be different enough, it is possible that the events can not be linked to each other through our similarity analysis, even if each one of them is well documented. Finally, our method compares n-grams and therefore determines similarity at a lexical level. This can lead to false positives when unrelated objects share certain n-grams. Yet, as described in Section 4.1, this type of feature representation enables to correlate heterogeneous data even when the standard is used incorrectly or the type of the data is unknown like, for example, in the case of binary strings that are part of indicators of compromise.

7. RELATED WORK

The body of work addressing threat intelligence issues has seen a surge in recent years thanks to the development of new sharing formats. Interestingly however, almost no previous work has been concerned with unifying and comparing the data described through different standards, as we do in this paper. Yet, some active and relevant areas exist that explore related research questions:

Threat intelligence. As discussed by Barnum [1], the community effort to design and extend the STIX format constitutes the most relevant and recent work to define a language that can add context and represent threat information in a structured and holistic way. Being this a recent development, current academic research is yet trying to understand the ecosystem of threat intelligence data by creating taxonomies and models [3]. Most researchers recognize the benefits of these technologies but their focus still lies on the design and implementation of efficient sharing systems [20, 31, 33] and the privacy implications resulting from distributing sensitive security data across heterogeneous organizations [10, 16]. Moreover, practitioners acknowledge the potential improvements for situational awareness [11, 27] that comes from the sharing, storage and analysis of threat data but also the difficulty to ensure consistent interpretation without the need of the analyst. Kampanakis [18], for example, presents an analysis of all the standards under current development and points to the underestimated challenge of data collection and automatic analysis. This is the exactly the field of operation of MANTIS. Most approaches in this direction stem from non-academic initiatives and are being developed both by the security community and by vendors like Microsoft [13], which holds large amounts of security data from its customers. For instance, the open-source framework CRITS [7] presents some resemblance to MANTIS. In particular, bucket lists and relationships can be assigned to top level objects in order to identify campaigns and attributions. However, these assignments need to be done manually by the analyst whereas finding such correlations and matchings automatically is precisely the main goal of our method. Finally, Woods et al. [41] have recently proposed a system to infer similarity relationships and functional clusters of indicators using information about reporting patterns. Although close to our work in its goal and methodology, their approach relies on data not based on standardized open formats.

Information retrieval for security. Tangential to our research is the field of information retrieval which covers a huge body of previous research and work. For brevity, we herein consider only previous research which like ours, makes use of information retrieval and data mining techniques for solving security problems. In particular, there exist several authors that deal with the question of how to efficiently detect and analyze new malware variants which have been submitted to application stores or analysis platforms by analyzing the output reports of their dynamic and static analysis [e.g., 2, 4, 14, 17]. For example, Graziano et al. [14] make use of ssdeep fuzzy hashes and code-based features to cluster malware

binaries and identify new families. Although this approach also tries to identify similar strains of malware their scope is limited to types of malware, where our broader view allow us to pinpoint disconnected elements from the same campaign. Closer to our method is the work introduced in [12], which also aims at finding similarities between graphs, in this case, call graphs extracted from malware samples. Our method, however, differs in that we are not inspecting simple instances of malware, but instead model a global picture of the reported threat data. Another line of security research that has combined information retrieval techniques and analysis of structured data focuses on the identification of similar segments in code of large software projects [21, 29, 38]. In particular, Uddin et al. [38] demonstrate that the simhash algorithm can help detecting similar code regions. While different in scope to our method, their approach also proves the effectiveness of Charikar's algorithm as the basis to implement techniques that can identify similar entities in large repositories of data.

8. CONCLUSION

In this paper we present MANTIS, a system that enables the authoring, collection and, most importantly, the analysis and correlation of threat intelligence data. To the best of our knowledge, MANTIS is the first open-source platform to provide a unified representation of threat data constructs from different standards that allows for assessing the similarity between heterogeneous reports at different levels of granularity regardless of their content, size or structure. The security analyst can initiate a search for similar constructs to a related incident with a query that goes from a simple string to a full report describing a multi-faceted attack.

We evaluate the performance of MANTIS in a series of experiments where given a security incident with a malware family, the similarity search integrated in MANTIS allows to retrieve related objects with a mean average precision of over 80%. That is, 4 out of 5 returned results correspond to the same malware family as the query. Finally and based on data from the attack and espionage campaigns *Stuxnet* and *Regin*, we show how MANTIS can be effectively used to assist the security analyst in the investigation of highly targeted security incidents.

References

[1] S. Barnum. Standardizing cyber threat intelligence information with the structured threat information expression (STIX). Technical report, MITRE Corporation, 2014.

[2] U. Bayer, P. M. Comparetti, C. Hlauschek, C. Kruegel, and E. Kirda. Scalable, behavior-based malware clustering. In *Proc. of Network and Distributed System Security Symposium (NDSS)*, 2009.

[3] E. W. Burger, M. D. Goodman, P. Kampanakis, and K. A. Zhu. Taxonomy model for cyber threat intelligence information exchange technologies. In *Proceedings of the 2014 ACM Workshop on Information Sharing & Collaborative Security*, pages 51–60. ACM, 2014.

[4] S. Chakradeo, B. Reaves, P. Traynor, and W. Enck. Mast: Triage for market-scale mobile malware analysis. In *Proc. of ACM Conference on Security and Privacy in Wireless and Mobile Networks (WISEC)*, 2013.

[5] M. S. Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 380–388. ACM, 2002.

[6] CIF. Collective intelligence framework. http://csirtgadgets.org/collective-intelligence-framework, visited August, 2016.

[7] CRITS. Collaborative research into threats. http://crits.github.io, visited July, 2016.

[8] M. Damashek. Gauging similarity with n-grams: Language-independent categorization of text. *Science*, 267(5199):843–848, 1995.

[9] R. Danyliw, J. Meijer, and Y. Demchenko. The incident object description exchange format (IODEF). Technical report, IETF RFC 5070, 2007.

[10] G. Fisk, C. Ardi, N. Pickett, J. Heidemann, M. Fisk, and C. Papadopoulos. Privacy principles for sharing cyber security data. In *Proceedings of the IEEE International Workshop on Privacy Engineering*, May 2015.

[11] P. Fonash. Using automated cyber threat exchange to turn the tide against ddos. http://rsaconference.com, 2014.

[12] H. Gascon, F. Yamaguchi, D. Arp, and K. Rieck. Structural detection of android malware using embedded call graphs. In *Proceedings of the 2013 ACM workshop on Artificial intelligence and security*, pages 45–54. ACM, 2013.

[13] C. Goodwin, J. P. Nicholas, J. Bryant, K. Ciglic, A. Kleiner, C. Kutterer, A. Massagli, A. Mckay, P. Mckitrick, J. Neutze, T. Storch, and K. Sullivan. A framework for cybersecurity information sharing and risk reduction. Technical report, Microsoft Corporation, 2015.

[14] M. Graziano, D. Canali, L. Bilge, A. Lanzi, and D. Balzarotti. Needles in a haystack: Mining information from public dynamic analysis sandboxes for malware intelligence. In *USENIX*, 2015.

[15] R. W. Hamming. Error-detecting and error-correcting codes. *Bell System Technical Journal*, 29(2):147–160, 1950.

[16] J. L. Hernandez-Ardieta, J. E. Tapiador, and G. Suarez-Tangil. Information sharing models for cooperative cyber defence. In *Cyber Conflict (CyCon), 2013 5th International Conference on*, pages 1–28. IEEE, 2013.

[17] J. Jang, D. Brumley, and S. Venkataraman. Bitshred: feature hashing malware for scalable triage and semantic analysis. In *Proc. of ACM Conference on Computer and Communications Security (CCS)*, pages 309–320, 2011.

[18] P. Kampanakis. Security automation and threat information-sharing options. *Security & Privacy, IEEE*, 12(5):42–51, 2014.

[19] Kaspersky. The Regin Platform: Nation-State Ownage of GSM Networks. Kaspersky Lab, November 2014.

[20] M. Korczynski, A. Hamieh, J. H. Huh, H. Holm, S. R. Rajagopalan, and N. H. Fefferman. DIAMoND: Distributed intrusion/anomaly monitoring for nonparametric detection. In *Proceedings the 24th International Conference on Computer Communications and Networks*, pages 1–8, 2015.

[21] J. Krinke. Identifying similar code with program dependence graphs. In *Proceedings of the Eighth Working Conference on Reverse Engineering (WCRE'01)*, 2001.

[22] R. Langner. Stuxnet: Dissecting a cyberwarfare weapon. *IEEE Security and Privacy*, 9(3), May 2011.

[23] Mandiant. Sophisticated indicators for the modern threat landscape: An introduction to OpenIOC. Technical report, Mandiant Whitepaper, 2013.

[24] Mandiant. APT1: Exposing one of China's cyber espionage units. Technical report, Mandiant Intelligence Center, 2013.

[25] G. S. Manku, A. Jain, and A. Das Sarma. Detecting near-duplicates for web crawling. In *Proceedings of the 16th international conference on World Wide Web*, pages 141–150. ACM, 2007.

[26] C. D. Manning, P. Raghavan, H. Schütze, et al. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.

[27] M. Orlando. Threat intelligence is dead. long live threat intelligence! http://rsaconference.com, 2015.

[28] OTX. Open threat exchange. https://www.alienvault.com/open-threat-exchange, visited August, 2016.

[29] A. Sæbjørnsen, J. Willcock, T. Panas, D. Quinlan, and Z. Su. Detecting code clones in binary executables. In *Proceedings of the Eighteenth International Symposium on Software Testing and Analysis*, 2009.

[30] G. Salton, A. Wong, and C. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[31] O. Serrano, L. Dandurand, and S. Brown. On the design of a cyber security data sharing system. In *Proceedings of the ACM Workshop on Information Sharing & Collaborative Security*, pages 61–69. ACM, 2014.

[32] Spamfighter/Der Spiegel. Top german official infected by regin malware. http://www.spamfighter.com/News-19917-Top-German-Official-Infected-by-Regin-Malware.htm, visited August, 2016.

[33] J. Steinberger, A. Sperotto, M. Golling, and H. Baier. How to exchange security events? overview and evaluation of formats and protocols. In *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*, pages 261–269. IEEE, 2015.

[34] Symantec. Stuxnet 0.5: The Missing Link. Symantec Security Response, February 2013.

[35] Symantec. Regin: Top-tier espionage tool enables stealthy surveillance. Symantec Security Response, August 2015.

[36] The Guardian. Uk company's spyware used against bahrain activist. https://www.theguardian.com/world/2013/may/12/uk-company-spyware-bahrain-claim, visited August, 2016.

[37] The New York Times. Computer systems used by clinton campaign are said to be hacked, apparently by russians. http://www.nytimes.com/2016/07/30/us/politics/clinton-campaign-hacked-russians.html, visited August, 2016.

[38] M. S. Uddin, C. K. Roy, K. A. Schneider, and A. Hindle. On the effectiveness of simhash for detecting near-miss clones in large scale software systems. In *WCRE*, 2011.

[39] VirusTotal. https://www.virustotal.com/.

[40] K. Wang, J. Parekh, and S. Stolfo. Anagram: A content anomaly detector resistant to mimicry attack. In *Recent Adances in Intrusion Detection (RAID)*, pages 226–248, 2006.

[41] B. Woods, S. Perl, and B. Lindauer. Data mining for efficient collaborative information discovery categories and subject descriptors. In *Proceedings of the 2nd ACM Workshop on Information Sharing and Collaborative Security*. ACM, 2015.