Automated Structured Threat Information Expression (STIX) Document Generation with Privacy Preservation

Farhan Sadique*, Sui Cheung†, Iman Vakilinia‡, Shahriar Badsha§, Shamik Sengupta¶
Department of Computer Science and Engineering
University of Nevada, Reno, NV, USA
*fsadique@nevada.unr.edu, †scheung@unr.edu, ‡ivakilinia@unr.edu, §sbadsha@unr.edu, ¶ssengupta@unr.edu

***Abstract*—The traditional approach to cybersecurity is struggling to defend against modern, dynamic and rapidly evolving cyber-attacks. Automated generation of cyber-threat intelligence (CTI) along with effective and real-time sharing of the CTI is required by organizations to prevent major cyber-attacks. It is nearly impossible to achieve comparable defense individually without cybersecurity information sharing. The first step towards this end is to collect cyber threat data and to represent that data in a standardized format. There is plenty of raw cyber threat data available to organizations in the form of firewall logs, malware signatures, spam emails, etc. However, the automated conversion of these data into a standard format has not been studied before. In this paper, we introduce a novel, privacy-preserving, mechanism to represent raw cyber threat-data in Structured Threat Information Expression (STIX) format in an automated manner. A complexity analysis shows that this process is suitable for large-scale deployments. This general guideline can be followed to represent all types of threat data in a standardized format. This will help the security administrators get a broad picture of the threat landscape and enable them to share these data with a cybersecurity information sharing platform for advanced analytics.**

***Index Terms*—cybersecurity automation, STIX, cybersecurity information sharing**

I. Introduction

Conventional cybersecurity measures rely heavily on the creation of signatures and static rules to identify and block threats. This renders the network vulnerable to the rapidly evolving cyber threat landscape. The risk is exacerbated by the emergence of organized and state-backed cyberattack campaigns, which have left even big firms practically defenseless [1]. Consequently, a new approach towards cybersecurity is essential to thwart organized cyber-attack campaigns proactively, to anticipate and mitigate large-scale exploitations and to respond faster to emerging cyber threats. The new approach should address the two major limitations of the traditional approach: (1) the delay in signature development, and (2) the time required to disseminate this information to interested parties.

Real-time cybersecurity information sharing coupled with automated Cyber Threat Intelligence (CTI) generation is envisioned to take care of both the concerns. The automated generation of CTI from raw cyber threat data would make the signature development much faster by omitting dependence on human interaction. This coupled with real-time sharing of this information via a centralized platform can achieve improved defense against recent dynamic attacks.

This work is part of a project called CYBersecurity information EXchange with Privacy (CYBEX-P) [2]. CYBEX-P intends to achieve both these goals of real-time cybersecurity information sharing and automated generation of CTI. CYBEX-P is a cybersecurity information sharing platform with a robust operational and administration structure. It provides the service of structured information exchange. Moreover, it addresses the inefficiency of managing defense against cyber-attacks by an individual entity by collaborative effort. So, it can protect business organizations more effectively from future cyber-attacks. Furthermore, it allows for assessing the threat landscape by providing the security status of systems and devices together, to prevent large-scale cyber-attack campaigns.

In relation, it is assumed that an abundance of cyber threat data is already available with concerned entities in the form of firewall logs, spam emails, malware signatures, etc. While the necessary data is either readily available or easily generated, it is very challenging to incorporate these with a central cybersecurity information sharing platform because:

1) The data do not have a standardized or normalized format. For instance, consider two firewalls by two different vendors. Even though, they contain the same type of data they are in different formats hence cannot be analyzed together.
2) Different types of data are not easily correlated. For example, a malicious IP address may show up in both a spam email and a firewall log. However, spam email filters and firewalls usually work separately; so these data are never correlated.
3) The data are too large in volume to be sorted manually by any security administrator.
4) The data are not centrally located rather distributed across different devices across the organization.
5) The data contain privacy-sensitive information related to the organization. Hence, different policies of the organization hinder the sharing of such data with external entities.

This research is supported by the National Science Foundation (NSF), USA, Award #1739032.

Therefore, to tackle the above challenges, we need an automated process to generate effective CTI from the raw cyber threat data. Additionally, the process should deal with different types and formats of data and should work from a central location of its own. Furthermore, we need a standardized representation of these data to share them with other organizations. Finally, as there is sensitive information in the collected data, the automation mechanism should be equipped with a privacy-preserving handler which manages the sharing of sensitive information.

To represent the data uniformly we consider the standardized format Structured Threat Information Expression (STIX) [3]. STIX is a language and serialization format that enables organizations to exchange CTI in a consistent and machine-readable manner.

In this paper, we define STIX automation as the process of automatically collecting cyber threat data, with privacy preservation, in the form of STIX document, to take care of the challenges mentioned earlier. In conjunction, we propose a novel framework for the automation of CTI generation and their effective representation using STIX.

While STIX incorporates the capability to represent threat data into effective CTI, it does not provide any guideline to automate the production of CTI from different kinds of threat data. Our system automates the production of STIX document directly from raw cyber threat data taking care of all the five challenges mentioned previously.

The novel contribution of our work is the inclusion of a privacy-handler in the process. Such an integrative approach ensures improved the privacy of the data because sensitive information is removed as the CTI is generated. Therefore, intentional or unintentional sharing of this data will not harm the organization in any way. Such automation of CTI generation with integrated privacy-handling has not been studied before. Furthermore, we perform a performance test of our architecture. The test reveals that the architecture is scalable and suitable for large deployments.

II. Related Work

Extensive research has been done on the automation of security in the computer systems. A comprehensive study of the autonomic systems capable of detecting and mitigating security threats at runtime has been done in [4]. The security and privacy challenges in the automation of security have been studied in [5]. Al-Nashif et al. [6] have introduced a multi-level intrusion detection system (ML-IDS) which exploits autonomic computing to automate the control and management of intrusion detection system. Webster et al. [7] have provided a framework for extracting features of computer systems' malicious activities. Panacea [8] has been introduced to apply machine learning techniques to automatically and systematically classify attacks detected by an anomaly-based network intrusion detection system. Lacoste et al. [9] investigated the automation of security in the cloud infrastructure where they classified the cloud infrastructure threats and presented an autonomic design for the cloud security management. Finally, CyTIME [10] has been discussed as an automation framework for writing security rules that closely complements our work. Even so, our work differs from their work as STIX Automation system produces STIX 2.0 data from raw cyber threat data whereas CyTIME works with STIX 2.0 data collected using a TAXII client to generate security rules automatically. In essence, none of these works have considered the modeling of threat data exchange in a structured format. In this work, we investigate the automated generation of STIX 2.0 document from raw cyber threat data, which has not been done before.

On the other hand, plenty of research has been conducted for cybersecurity information sharing [11]–[16]. Various protocols and specifications such as TAXII, STIX, CybOX, OpenIOC, VERIS, MAEC, SCAP, and IODEF have been developed to provide a common platform for sharing cybersecurity information [17]–[19]. Authors in [12] discuss the effectiveness of cybersecurity information sharing and formulate it as a risk-based decision-making model with a directed graph.

Security and privacy challenges in cybersecurity information sharing have been studied in [13], [15], [20], [21]. A cryptographic privacy-preserving framework based on group signature scheme for sharing cybersecurity information has been presented in [15]. An attribute-based cybersecurity information exchange platform using attribute-based encryption have been introduced in [13]. Authors in [20] have investigated the trade-off between sharing cybersecurity information and privacy cost in a dynamic 3-way game model between attacker, organizations, and cybersecurity information sharing platform. Vakilinia et al. [11] have studied the application of STIX to share a set of vulnerable passwords in brute-force attempts.

Motivated from the above works, we present a framework to automatically generate effective CTI as STIX documents from raw threat data.

III. Preliminaries

A. Overview of STIX 2.0

One of the challenges threat-sharing organizations face is the ability to structure cyber threat information, yet not lose the human judgment and control involved in sharing. STIX is an information model and serialization for Cyber Threat Intelligence (CTI) [3]. STIX standardizes CTI data in a machine-readable specification to support cyber threat analysis and information sharing. In this paper, we have used STIX version 2.0.

STIX is a graph-based information model where graph nodes are described as *STIX Domain Objects* (SDO), and graph edges are described as STIX Relationship Objects (SROs). Graph-based models provide a consistent, structured, and flexible data representation which allows information exchange. STIX 2.0 defines twelve different STIX Domain Objects and two STIX Relationship Objects [22].

The suitable SDO for representing raw threat data is called *observed data*. Irrespective of the data type an *observed data* object has some common properties: *type, id, created_by_ref, created, modified, first_observed, last_observed,* and *number_observed*.

In addition to these common fields, an *observed data* contains a structured representation of raw threat data and their

Fig. 1: STIX 2.0 data structure

properties in the cyber domain. For that purpose, an *observed data* object contains one or more *cyber observable objects* (COO). Examples of COOs are IP address, email message, network traffic data, etc.

While a COO contains the raw threat data, it does not contain any information on the time of observance or source of the data. This information is contained in the *observed data* object that houses the COO. A single *observed data* may contain more than one COO if the COOs are related by context and represent a single event.

B. STIX 2.0 and JSON

STIX 2.0 does not introduce a new file format. Instead, it primarily uses JavaScript Object Notation (JSON) as the serialization format. In fact, the mandatory-to-implement serialization for STIX 2.0 is JSON. STIX 2.0 documentation specifies the necessary properties and corresponding data types of an SDO. For the rest of the paper, we will use the term STIX to denote STIX 2.0. JSON makes STIX format simultaneously human-readable and machine-readable. The figure 1 shows the structure of a STIX *observed data* object in JSON serialization. It contains a simple COO in the objects field that represents an IP address.

IV. STIX Automation Methodology

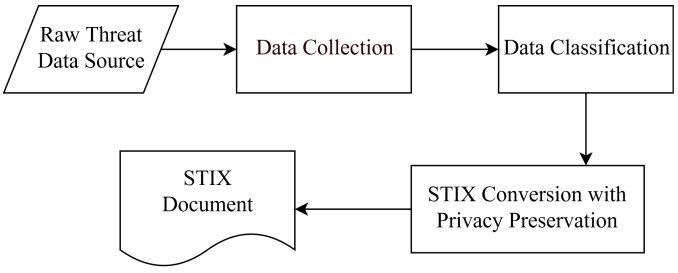

Fig. 2: STIX automation process flowchart

The flowchart in figure 2 shows the proposed framework for STIX automation. The novel approach in our system is that it incorporates a privacy handler into the data conversion module. This integrated approach ensures greater protection of sensitive information and makes it easier to share these data with other parties. The life-cycle of the data in our system along with the privacy handling mechanism is described in detail below:

A. Raw Threat Data

The input to the process is machine generated raw threat data. Examples of raw threat data are firewall and IDS/IPS logs, system logs, emails, malware signatures, suspicious URLs, etc. These threat data are automatically generated as a direct or indirect consequence of an event and contain relevant information to that event. As an illustration, the event of receiving a network packet makes a firewall generate a packet log message.

B. Data Collection

The data collector module of our STIX automation system collects the raw threat data from their specific sources as they are being generated. There are numerous types of raw threat data sources available to us. Moreover, the threat data can be delivered in different formats including plain text, XML, JSON, etc. Additionally, there are different ways to collect the data from their respective sources including TCP stream or as plain text files. The data collector module in our system takes care of these diverse considerations.

The data collector has separate plugin modules for each data source type. For example, some modern devices support exporting real-time data via web sockets whereas others have built-in API to export data on demand. In case of conventional network and Linux devices, we used the popular *syslog* protocol [23] to transfer device logs to our collector. In some exceptional cases, we had to collect the data as text files directly from the source server.

C. Data Classification

After collecting raw threat data from respective sources, the collector passes the data onto our classifier module. The collector maintains an internal buffer of the data and transfers one event at a time to the classifier. The classifier then classifies that data into the appropriate category based on the source. Whenever a particular data source is integrated via collector configuration, it is given a particular category value. The classifier later uses this category value to classify the data from heterogeneous sources.

The first thing our classifier module does is to identify the type of these data in line with STIX documentation. The second task of the classifier is to identify the formatting of the data later required for parsing. It is required because the same type of data can be represented in different formats depending on the particular software, operating system, device or vendor. This is demonstrated by the apparent dissimilarity in the logs of two different firewalls which nevertheless contain the same information.

D. STIX Conversion

The next stage in the STIX automation process is STIX conversion. The data that the STIX converter receives is already classified into appropriate categories. The conversion

module converts the data into corresponding STIX document. As discussed in section III The suitable STIX domain object (SDO) for raw threat data are *observed data*. An *observed data* contains one or more *cyber observable objects*.

We have used the Python programming language for parsing and converting raw data. The parsing scripts are written by us and are available from our GitHub repository [24]. On the other hand, OASIS TC maintains an official Python library[1] for STIX serialization. We have used built-in object definitions in the STIX library. However, we have added custom objects or custom properties to predefined objects whenever necessary. The function for creating *observed data* performs the correct serialization of the different properties of threat data. It also incorporates data validation and error checking.

Protecting Sensitive Information

As the threat data may convey private information, it is not always possible to share such information in a raw format. Thus, we need a mechanism to protect the sensitive information before sharing. To this end, we classify the data into four categories based on sensitivity:

Level 0. These data are not sensitive and are transferred without encryption. Examples are a virus file, a malicious IP address, and a malicious URL. Fuzzy Hashing[2] is used to check the homologies of larger data like files/emails. Fuzzy Hashing reduces the dimensionality of high-dimensional data and hashes input items so that similar items map to the same *buckets* with high probability.

```
"objects": {
  "0": {
    "type": "email-addr",
    "value": "scholars@us.net",
    "display_name": "Scholars"
  },
  "1": {
    "type": "email-addr",
    "value": "d4c74594d841139328695756648b6bd6",
    "display_name": "4c2a904bafba06591225113ad17b5cec
  },
  "2": {
    "type": "email-message",
    "from_ref": "0",
    "to_refs": [
      "1"
    ],
    "is_multipart": false,
    "date": "2017-12-09T10:44:06.000Z",
    "subject": "IT TRAINING SCHOLARSHIPS",
     "content_type": "text/plain; charset=us-ascii",
     "body":"IT TRAINING TUITION SCHOLARSHIPS"
  }
}
```

Fig. 3: STIX representation of a spam email where the name and email address of recipient are masked using MD5

Level 1. These data have sensitive information about the source organization which might be exploited by the attacker for reconnaissance. For instance, consider that the source wants to share information about a new attack on its database server, however sharing the detailed information about the server network configuration, version and the model of the database server helps an attacker to have a better understating of the victims underlying network infrastructure. Thus, the source applies masking or generalization techniques (such as k-anonymity [25]) to hide the underlying sensitive information.

For instance, figure 3 shows the STIX representation of a spam email message. Here, the destination email address and the name of the recipient contain privacy-sensitive information. That is why these values are masked by the source organization using md5 hash. On the other hand, the message source and the message body are related to the spammer and are shared as plain text.

Level 2. For this set of data, encryption is used to hide information for access control. In this case, only subscribers who have access to the key can decrypt the message. For instance, consider a new unpatched vulnerability has been detected, and such information should not be shared with the public.

Level 3. By sharing such information the subscribers are only able to find out if the other party has received the same message or not and no more information can be inferred. Private Set Intersection (PSI) [26] protocols are applied for this purpose. For instance, consider that an organization has received an unknown email which it cannot classify as SPAM or benign. As the email might be a normal message, the organization cannot share the message with other organizations. Hence, organizations initiate the PSI protocol to know if the message has been received by any other organization.

E. STIX Document

The output from our STIX automation system is a STIX document as shown in figure 1. The data are serialized in JSON as discussed in section III. Members of Cyber Threat Intelligence Technical Committee have built the JSON schemas. As a result, the STIX documents output from our system adheres to the predefined schema. This STIX document can now be stored in a suitable database or transferred to other systems.

V. Demonstration

This section describes a practical implementation of STIX automation. We have used a generalized structure to demonstrate the feasibility of the STIX automation process. For this demonstration, we have considered three different kinds of threat data: malicious emails, firewall logs, and login credentials from brute force. First, we describe the architecture of our system followed by a detailed discussion of the STIX automation of each data source.

A. System Architecture

Figure 4 shows the architecture of our system. This architecture identifies the position of the STIX automation module in a typical security system. It also clarifies the data flow in the system to aid the integration of the STIX automation module into an existing system.

Our system pipeline consists of three stages. Firstly, every attack that an attacker makes on our system is considered an event. An event can be as simple as receiving a single

[1]https://github.com/oasis-open/cti-python-stix2

[2]https://ssdeep-project.github.io/ssdeep/index.html

malicious packet or can be as complex as receiving a malware attached to an email. In the second stage, the event is recorded as threat data in our data collection module. Finally, these threat data are passed onto the analysis and alert sub-system. The analysis and alert subsystem performs further analysis on the threat data depending on the particular needs of the organization.

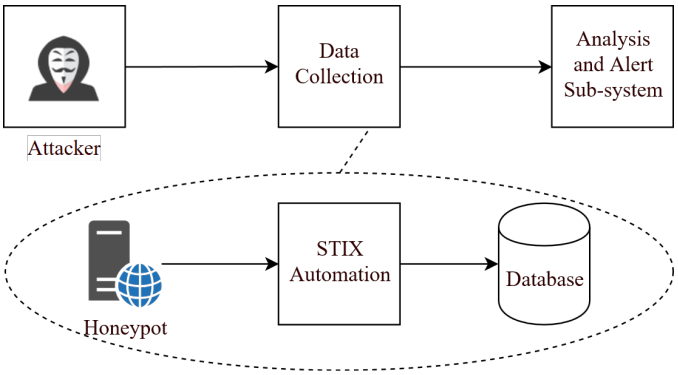

Fig. 4: System architecture

The data collection sub-system further consists of three parts. The first part is a research honeypot that we set up for this work. The honeypot collects different kinds of raw threat data depending on the type of attack. The second part is the STIX automation module. The STIX automation module interfaces directly with the sources of raw threat data. This module collects raw threat data from our honeypot and outputs appropriate STIX document. Finally, the STIX documents are stored in a central database for future use.

B. Technical System Description

Figure 5 shows the network design of our system. We have set up a honeypot on a virtualized environment with Kernel-based Virtual Machine (KVM). KVM is an open source, full virtualization solution for Linux that can run multiple virtual machines. On top of KVM, we have set up three more hosts using Quick Emulator (QEMU). QEMU is another open source hosted hypervisor for Linux that performs hardware virtualization. We have kept the hosts separate because of security reasons. Running them in separate hosts ensures that data in one server is not threatened even if another server is compromised.

The three hosts have three different services running in them. The first one is an SSH server that is facing the internet and listening on the default port making it an easy target for brute force or dictionary attacks. The server records the attacker's IP address along with the usernames and passwords that are used for login attempts.

The second server in the honeypot is an email server that hosts some dummy email addresses. We purchased two domains for this work. The dummy email addresses are registered under those two domains. These email addresses are published on different dummy sites on the internet.

Finally, the third server in the honeypot is a web server. This web server hosts two dummy static websites. We have posted

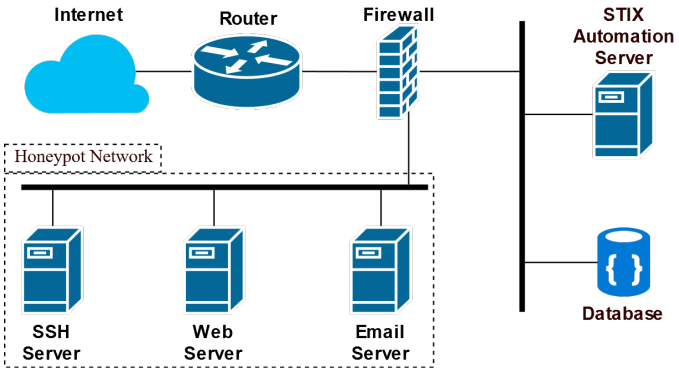

Fig. 5: Network diagram of data collection sub-system

our honeypot email addresses to these websites to make them discoverable by web-crawlers and attackers.

Additionally, there is a network firewall configured in front of the honeypot network. The firewall acts as the gateway for all the servers in the honeypot. This firewall filters incoming and outgoing packets to allow access to only the required ports in a particular server. It also serves as a threat data source by logging all network layer activities and by generating network layer threat alerts.

Our STIX automation server runs on a different network than the honeypot server. This server also uses the firewall as the gateway. The STIX automation server periodically collects the raw threat data from the firewall and other servers using different methods as examined in section IV.

C. STIX Automation of Firewall Log Data

Firewalls are principal sources of cyber threat data in many organizations. Firewalls log all incoming and sometimes outgoing network packets depending on the configuration. Following is a detailed explanation of how STIX automation is performed on this data source:

a) Raw Threat Data: In reference to figure 2, the input to the STIX automation process is raw threat data. In our project we have set up an *iptables* firewall, in a Linux host, that captures all packets destined to our honeypot network. The firewall generates one *syslog* [23] message describing each received packet. A sample *syslog* message generated by *iptables* looks like that shown in fig 7.

The parameters of interest in the message are either IP header fields or transport layer header fields (TCP in this example). The same data are included in other firewall logs in a different arrangement.

b) Data Collection: The second step in STIX automation is data collection. The *syslog* protocol supports the logging of *syslog* messages in a remote server over a network. We have used this facility to collect *syslog* messages directly into our collector module. The collector stores the messages temporarily in its storage. It transfers the messages to the data classifier one at a time and deletes them afterward.

c) Data Classification: The next step after data collection is data classification. When the collector receives the *syslog* message from the firewall, it tags the data with a unique

```
"objects": {
  "0": {
    "type": "ipv4-addr",
    "value": "61.129.45.35"
  },
  "1": {
    "type": "ipv4-addr",
    "value": "11.11.11.69"
  },
  "2": {
    "type": "network-traffic",
    "src_ref": "0",
    "dst_ref": "1",
    "src_port": 80,
    "dst_port": 6487,
    "protocols": [
      "TCP"
    ],
    "src_byte_count": 40,
    "ipfix": {
      "ipTTL": 107,
      "flowDirection": 0,
      "tcpWindowSize": 0,
      "ipClassOfService": 0,
      "tcpUrgentPointer": 0
    },
    "extensions": {
      "tcp-ext": {
        "src_flags_hex": "14"
      }
    }
  }
}
```


Fig. 6: STIX representation of firewall log

```
Feb 1 00:53:10 bridge kernel: INBOUND TCP: IN=br0
PHYSIN=eth0 OUT=br0 PHYSOUT=eth1 SRC=61.129.45.35
DST=11.11.11.69 LEN=40 TOS=0x00 PREC=0x00 TTL=107 ID=25935
PROTO=TCP SPT=80 DPT=6487 WINDOW=0 RES=0x00 ACK RST URGP=0
```

Fig. 7: Sample log message generated by *iptables* firewall

label. The classifier then makes use of this label to identify the data as a *syslog* message that contains packet log generated by an *iptables* firewall.

d) STIX Conversion: Next, the STIX conversion module parses the plaintext *syslog* message using Python. Most of the TCP/IP header field values are extracted as is except for the timestamp. This is because STIX 2.0 timestamp field must be a valid RFC 3339 [27] formatted timestamp using the format `YYYY-MM-DDTHH:mm:ss[.s+]Z` in UTC timezone.

The suitable cyber observable object to represent a network packet is the STIX *network-traffic* object. All these data can be included in the aforementioned *network-traffic* COO as shown in figure 6.

Since the data is collected from a honeypot, it does not contain any privacy-sensitive information. For this reason, it is classified as privacy level 0 and represented without any modifications.

e) STIX Document: The output of our STIX automation system is a STIX document. Figure 6 shows the final representation of the firewall log in STIX format. We have shown only the objects property of an *observed data* here because the other properties are the same as shown in figure 1.

There are three *cyber observable objects* in this example. The *network-traffic* COO references two other COOs each of type *ipv4-addr* which contain the source and destination IP addresses. The timestamp is not present in the figure because the timestamp is included in the *observed data* object that encloses this *cyber observable object*.

Most of the properties in the *network-traffic* COO are self-explanatory except for the *src_flags_hex* field. Its value specifies the source TCP flags as the hexadecimal representation of the union of all TCP flags of the packet.

D. Evaluation & Scalability

To show the feasibility of our STIX Automation process we used the public Scan 34 dataset [28] by the honeynet project. This dataset contains about 400000 lines of *iptables* firewall log. We randomly selected parts of this dataset, of various lengths, and fed it into the system for evaluation. The evaluation metric that we used is the time our system takes to process N log messages. We have measured the time taken from input to finally storing in the database. A line of best fit is drawn among the data points. The result is shown in figure 8.

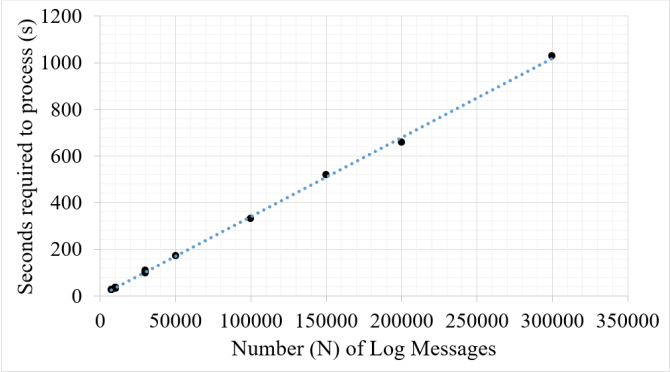

Fig. 8: Evaluation of Complexity

The test was done with a run-of-the-mill computer, to show the complexity of our mechanism and hence forecast the scaling and growth of the system. For this reason, the processing was done using a single core and the absolute times are of less interest here.

It can be inferred from the test that the complexity is linear because the processing time grows linearly with the input size. This is desirable in a system like this which is going to deal with a large volume of data. This makes the system suitable for horizontal scaling using distributed computing systems. Furthermore, each log message is processed independent of each other making the process ideal for distributed computing.

VI. Conclusion

In this work, we have studied STIX automation to facilitate and accelerate the process of detecting threats. To achieve this goal, we have presented an architecture which receives threat data as input, normalizes the data, classifies attacks, and outputs relevant STIX document. We have also investigated this framework over three kinds of threat data to demonstrate the feasibility of the methodology.

STIX automation benefits an organization in multiple ways. First of all, the centrally collected cyber threat data allows for further processing such as aggregation and correlation. This would significantly help security administrators get a better insight into the security status of their system. Additionally, the standard representation of cyber threat data is required for an organization to share these data with other organizations via a cybersecurity information sharing platform. Furthermore, many organizations already have vast amounts of cyber threat data but have little to no use for them because they are not centrally located or represented in the same manner. Our system allows for the retrospective use of these data for advanced analysis.

STIX automation thus paves the way for total automation of cybersecurity, from the collection of raw threat data to the configuration of necessary parameters, along with instantaneous sharing of such data. This more efficient and robust approach towards cybersecurity, with reduced dependence on human interaction, is required to protect organizations from modern dynamic cyberattacks.

References

[1] J. P. Farwell and R. Rohozinski, "Stuxnet and the future of cyber war," *Survival*, vol. 53, no. 1, pp. 23–40, 2011.

[2] "CICI: CE: Implementing CYBEX-P: Helping Organizations to Share with Privacy Preservation," https://www.nsf.gov/awardsearch/showAward?AWD_ID=1739032, July 2017.

[3] S. Barnum, "Standardizing cyber threat intelligence information with the structured threat information expression (stix)," *MITRE Corporation*, vol. 11, pp. 1–22, 2012.

[4] E. Yuan, N. Esfahani, and S. Malek, "A systematic survey of self-protecting software systems," *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, vol. 8, no. 4, p. 17, 2014.

[5] D. M. Chess, C. C. Palmer, and S. R. White, "Security in an autonomic computing environment," *IBM Sys. journal*, vol. 42, pp. 107–118, 2003.

[6] Y. Al-Nashif, A. A. Kumar, S. Hariri, Y. Luo, F. Szidarovsky, and G. Qu, "Multi-level intrusion detection system (ml-ids)," in *Autonomic Computing, 2008. ICAC'08. International Conference on*.

[7] G. D. Webster, Z. D. Hanif, A. L. Ludwig, T. K. Lengyel, A. Zarras, and C. Eckert, "Skald: a scalable architecture for feature extraction, multi-user analysis, and real-time information sharing," in *International Conference on Information Security*. Springer, 2016, pp. 231–249.

[8] D. Bolzoni, S. Etalle, and P. H. Hartel, "Panacea: Automating attack classification for anomaly-based network intrusion detection systems," in *International Workshop on Recent Advances in Intrusion Detection*. Springer, 2009, pp. 1–20.

[9] M. Lacoste, A. Wailly, and H. Debar, "Self-defending clouds: Myth and realities," *C&ESAR 2012*, p. 45, 2012.

[10] 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*. ACM, 2018, p. 7.

[11] I. Vakilinia, S. Cheung, and S. Sengupta, "Sharing susceptible passwords as cyber threat intelligence feed," in *Military Communications Conference (MILCOM), MILCOM 2018-2018 IEEE*. IEEE, 2018.

[12] 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*. IEEE, 2013, pp. 1–28.

[13] I. Vakilinia, D. K. Tosh, and S. Sengupta, "Attribute based sharing in cybersecurity information exchange framework," in *Performance Evaluation of Computer and Telecommunication Systems (SPECTS), 2017 International Symposium on*. IEEE, 2017.

[14] L. Dandurand and O. S. Serrano, "Towards improved cyber security information sharing," in *Cyber Conflict (CyCon), 2013 5th International Conference on*. IEEE, 2013, pp. 1–16.

[15] I. Vakilinia, D. K. Tosh, and S. Sengupta, "Privacy-preserving cybersecurity information exchange mechanism," in *Performance Evaluation of Computer and Telecommunication Systems (SPECTS), 2017 International Symposium on*. IEEE, 2017.

[16] I. Vakilinia and S. Sengupta, "A coalitional cyber-insurance framework for a common platform."

[17] "Standards and tools for exchange and processing of actionable information," https://www.enisa.europa.eu/publications/standards-and-tools-for-exchange-and-processing-of-actionable-information/.

[18] 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*. IEEE, 2015, pp. 261–269.

[19] P. Kampanakis, "Security automation and threat information-sharing options," *IEEE Security & Privacy*, vol. 12, no. 5, pp. 42–51, 2014.

[20] I. Vakilinia, D. K. Tosh, and S. Sengupta, "3-way game model for privacy-preserving cybersecurity information exchange framework," in *Military Communications Conference (MILCOM), MILCOM 2017-2017 IEEE*. IEEE, 2017, pp. 829–834.

[21] I. Vakilinia and S. Sengupta, "A coalitional game theory approach for cybersecurity information sharing," in *Military Communications Conference, MILCOM 2017-2017 IEEE*. IEEE, 2017, pp. 237–242.

[22] "STIX™ Version 2.0. Part 2: STIX Objects. Edited by Rich Piazza, John Wunder, and Bret Jordan." 19 July 2017. OASIS Committee Specification 01. http://docs.oasis-open.org/cti/stix/v2.0/cs01/part2-stix-objects/stix-v2.0-cs01-part2-stix-objects.html. Latest version: http://docs.oasis-open.org/cti/stix/v2.0/stix-v2.0-part2-stix-objects.html.

[23] R. Gerhards, "The syslog protocol," Tech. Rep., March 2009. [Online]. Available: https://doi.org/10.17487/rfc5424

[24] F. Sadique, "Cici," https://github.com/qclassified/cici/, 2018.

[25] L. Sweeney, "k-anonymity: A model for protecting privacy," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 05, pp. 557–570, 2002.

[26] B. Pinkas, T. Schneider, and M. Zohner, "Faster private set intersection based on ot extension." in *USENIX Security Symposium*, vol. 14.

[27] G. Klyne and C. Newman, "Rfc 3339: Date and time on the internet: Timestamps," *The Internet Society, Request for Comments Jul*, 2002.

[28] A. Chuvakin =, "Scan 34 - the honeynet project," http://log-sharing.dreamhosters.com/SotM34-anton.tar.gz, February 2005.