Future Generation Computer Systems 135 (2022) 30–43

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Boosting Cyber-Threat Intelligence via Collaborative Intrusion Detection

Massimo Guarascio *, Nunziato Cassavia, Francesco Sergio Pisani, Giuseppe Manco

Institute for High Performance Computing and Networking, Italian National Research Council, ICAR-CNR, Via P. Bucci, 87036, Rende (CS), Italy

ARTICLE INFO

Article history:
Received 4 November 2021
Received in revised form 20 April 2022
Accepted 23 April 2022
Available online 30 April 2022

Keywords:
Cyber Threat Intelligence architecture
Security data enrichment
Active Learning
Intrusion Detection System
Threat analytics
SIEM

ABSTRACT

Sharing threat events and Indicators of Compromise (IoCs) enables quick and crucial decision making relative to effective countermeasures against cyberattacks. However, the current threat information sharing solutions do not allow easy communication and knowledge sharing among threat detection systems (in particular Intrusion Detection Systems (IDS)) exploiting Machine Learning (ML) techniques. Moreover, the interaction with the expert, which represents an important component to gather verified and reliable input data for the ML algorithms, is weakly supported. To address all these issues, ORISHA, a platform for ORchestrated Information SHaring and Awareness enabling the cooperation among threat detection systems and other information awareness components, is proposed here. ORISHA is backed by a distributed Threat Intelligence Platform based on a network of interconnected Malware Information Sharing Platform instances, which enables the communication with several Threat Detection layers belonging to different organizations. Within this ecosystem, Threat Detection Systems mutually benefit by sharing knowledge that allows them to refine the underlying predictive accuracy. Uncertain cases, i.e. examples with low anomaly scores, are proposed to the expert, who acts with the role of *oracle* in an Active Learning scheme. By interfacing with a honeynet, ORISHA allows for enriching the knowledge base with further positive attack instances and then yielding robust detection models. An experimentation conducted on a well-known Intrusion Detection benchmark demonstrates the validity of the proposed architecture.

1. Introduction

Nowadays, organizations and users face an enormous amount of sophisticated, targeted and widespread cyberattacks. Malicious actors were proven able to compromise government computer systems as well user devices causing various types of damages. Phishing, identity theft, information leakage, DDOS and botnet represent some examples of popular threat occurred in 2020 [1]. The outbreak of COVID-19 has further exacerbated this situation. As the virus spread during the early part of the 2020, the number of cyberattacks against organizations grew exponentially, reaching a peak in April [2,3]. The pandemic unveiled different vulnerabilities of well-known platforms, applications and systems, and simultaneously stimulated the interest for promoting the usage of information sharing technologies to increase the degree of security for enterprises and organizations.

In this context, timely sharing threat events and Indicators of Compromise (IoCs) among organizations can be crucial in order to make quick decisions and set up effective countermeasures. In particular, coupling proactive threat information sharing and defensive mitigation strategies allow for strengthening the resilience of the entities belonging to the trusted community by yielding a herd immunity against new (possibly unknown) attacks and malware. Hence, the need to design platforms, tools and methodologies for accessing and sharing threat events in order to react promptly and prevent further damages. *Cyber Threat Intelligence* (CTI) platforms are considered valuable tools for easing the management of threat information [4]. These solutions allow organizations to easily handle the whole process of gathering, preprocessing, enriching, correlating, analyzing and sharing threat events and associated data [5]. According to Dandurand and Serrano [6], the main requirements for a *Threat Intelligence Platform* (TIP) can be summarized into (*i*) providing services for information sharing, (*ii*) automatizing the process, (*iii*) enabling functionalities for collaborative threat data analysis. Although TIP technologies can potentially bring important benefits to the organizations, developing a comprehensive Cyber Threat Intelligence platform able to handle information from different sources is difficult to achieve [7]. Standardization, privacy and reliability of the shared information are just some examples of the open challenges in defining a fully operational platform. Some recent works

* Corresponding author.
E-mail addresses: massimo.guarascio@icar.cnr.it (M. Guarascio), nunziato.cassavia@icar.cnr.it (N. Cassavia), francescosergio.pisani@icar.cnr.it (F.S. Pisani), giuseppe.manco@icar.cnr.it (G. Manco).

https://doi.org/10.1016/j.future.2022.04.028

proposed interesting solutions for easing threat data sharing but they only focused on addressing some of the above-mentioned issues [8,9]. Wagner et al. [10] provide an extensive overview of the current state-of-the-art and highlight open technical and non-technical challenges for the cyber threat intelligence.

When we consider also Machine Learning (ML) based threat detection systems, the situation is further exacerbated by some specific challenges:

- The quality of threat feeds and events is not guaranteed and there is a need for a reliable and automated threat analysis and mitigation. This is particularly problematic for Threat Detection Systems (TDS, such as Intrusion Detection Systems) based on AI/ML techniques, which are typically affected by high false positive rates thus nullifying both their detection capabilities and the informative content of their detection.
- Combining Threat Intelligence Platforms with AI based Threat Intelligence solutions in a single comprehensive framework is a challenging task since the former ones usually work in an event-based fashion while the latter are typically data-driven. In particular, there is no simple way for two machine learning algorithms to share, analyze and compare their findings within a standardized framework that can boost their attack detection and mitigation capabilities.

Contribution and organization. In this work, we propose ORISHA, a platform for ORchestrated Information SHaring and Awareness particularly focused on the two above mentioned issues. ORISHA has the twofold objective of (i) improving the accuracy of TDS in detecting incoming attacks, and (ii) enabling the sharing of reliable and relevant threat information among organizations and threat detection algorithms. The guiding principle is that TDSs can benefit each other mutually by sharing knowledge, since a threat feed shared by a TDS can be exploited by another one to improve its threat modeling strategies. At the same time, the same feed can be enriched by enabling communication with other layers, which can be exploited to further characterize the underlying threats and provide important insights about them.

The framework is based on an exchange protocol where information concerning threats is published on a distributed TIP (implemented as a network of several Malware Information Sharing Platform (MISP) instances) and made accessible to other actors, which can then exploit their capabilities in order to further characterize the feed and acquire it in their knowledge base. This collaboration framework has several advantages. On one side, it allows for improving the performances of the threat prevention and detection systems, yielding more robust and effective machine/deep learning models, reducing the false alarm rate and the average time required for identifying a successful intrusion. On the other side, it enables more robust threat intelligence by allowing to better contextualize threat data and devise flexible strategies, methodologies and data formats for collaborative threat intelligence, also by improving the notification mechanisms to appropriately notify relevant stakeholders having different needs for a contextualized interpretation of threat data.

The rest of the paper is structured as follows. Section 2 provides an overview on the main platforms and solutions for threat information sharing and awareness. We also review some relevant works which focus on the exploitation of external sources to improve the performance of TDS and the role of Active Learning in cybersecurity solutions. Section 3 describes the devised ORISHA platform for threat event sharing. We detail the data exchange format and discuss how it integrates data-driven intrusion detection system. In particular, Section 3.3 exemplifies the whole information flow and illustrates the benefits in adopting the proposed protocol for the overall threat information sharing process both, in terms of attack detection capability and quality of the shared information. In Section 4 we describe a suite of experiments which demonstrate the benefits of adopting ORISHA within an intrusion detection scenario. Finally, Section 5 concludes the paper and outlines future research directions.

2. Background and related works

Threat Intelligence refers to the task of gathering data concerning attacks or breaches (e.g. context, methods, indicators, devices, etc.) for enabling the organizations to set up countermeasures on the basis of a wide range of information [11]. In order to enhance prevention and detection of new threats, organizations can collaborate by sharing information about recent discovered threats. This information is usually made available under the form of *Indicator of Compromises* (IoCs). Basically, an IoC is a piece of forensic data identifying potentially malicious activity on a system or network. Typical examples of IoCs are the source ip address of an attack, the hash of a malicious executable file or the URL of a phishing web site.

Threat Intelligence represents an emerging and relatively new research line in the field of cybersecurity and, as highlighted in [12], there is a growing interest in this topic by both academic and industrial entities. Cooperation and data sharing allow for improving the security of the computer networks and reducing the risk of compromising. The recent research has mainly focused on devising tools for the threat information sharing, so that in the last years we have observed a proliferation of threat intelligence platforms [13]. The lack of standards and solid approaches resulted in different combinations of products and methodologies frequently erroneously labeled as threat intelligence. Johnson et al. [4] provided some tentative guidelines, by devising information sharing goal for organizations, identifying threat information sources and proposing rules for managing the publication and distribution of the threat information of an organization. However, there is no consensus among researchers and practitioners on the usage of a specific methodology or technology since there is no comprehensive solution to the standardization, privacy and reliability issues related to the sharing process.

In the following, we provide an overview of the main standards and technologies for the threat intelligence, we survey some relevant works exploiting the collaboration among different network security tools and analyze some emerging approaches in this research field.

Standards and solutions for threat information sharing. While in the past, channels such as mail messages, phone calls, ticket systems, or face-to-face meetings were the main ways to quickly share threat information, with the exponential growth cyberattacks these channels have been replaced with advanced solutions able to automate the whole process. Different standards, such as Structured Threat Information CybereXpression (STIX) [14], Cyber Observable eXpression (CybOX) [15], Incident Object Description Exchange Format (IODEF) [16] and Trusted Automated eXchange of Indicator Information (TAXII) [17], have been proposed to simplify the sharing of these indicators. As above mentioned, a large number of platforms based on these standards have been proposed [5,12] and here we review some of those dedicated to threat information sharing [18]:

- **MISP** (Malware Information Sharing Platform)[1] is an open source software solution for collecting, storing, distributing and sharing cybersecurity indicators and threat information [19]. The platform encompasses several public MISP communities, available and interconnected via MISP API, sharing threats or cybersecurity indicators worldwide.

[1] https://misp-project.org/.

Table 1
Threat intelligence solution comparison.

Platform	Supported standards	Support for cooperation	Extensibility and support for custom solutions	SIEM and IDS integration	Documentation	Licence
MISP	STIX, CybOX, TAXII	High	✓	✓	Advanced	Open source (GNU General Public License)
MITRE CRITs	STIX, TAXII, OpenIOC, Send/receive information through Facebook's ThreatExchange	High	×	×	Medium	Open source (GNU General Public License)
CIF	STIX, CybOX	Medium	×	×	Basic	Open source (GNU General Public License)
EclecticIQ platform	STIX, TAXII	Medium	Partial	×	Medium	Commercial
LookingGlass cyber	STIX and TAXII via *scoutPRIME*	High	×	×	Basic	Commercial

- **MITRE CRITs** (Collaborative Research Threats)[2] is an open source malware and threat repository that leverages different open source software to create a unified tool for analysts and security experts engaged in threat defense [20]. This tool employs a hierarchy to structure cyber threat information, thus enabling the analysts to perform complex queries and discover previously unknown related content.
- **CIF** (Collective Intelligence Framework) is an open source cyber threat intelligence platform that allows gathering data from different sources and exploiting them for threat identification, detection, and mitigation. IP addresses, domains, and URLs can be stored and preprocessed as threat data within the platform. It permits to ingest many different sources of data sets such as feeds of malicious domains.
- **EclecticIQ Platform**[3] is a commercial platform gathering and interpreting intelligence data from open sources, commercial suppliers and industry partnerships. This platform is based on STIX and TAXII standards and provides analyst-friendly workflows.
- **LookingGlass Cyber**[4] provides two commercial solutions, scoutPRIME and scoutSHIELD. These tools provide utilities for both collecting threat information and handling threat responses, by making available collaboration and sharing tools for the threat analysis.

Table 1 compares the TIP solutions illustrated above and highlight the main features that characterize them. Among the dimensions of comparison, we mention support for cooperation, extensibility and integration with advanced tools.

Honeypot technology. Honeypots are well-consolidated technologies in the cybersecurity domain and allow for logging precious information on the behavior of attacks and malware. Basically, a honeypot is a computer system devised as a bait for hackers, with the purpose to gather information about prospective attacks. The main idea consists in monitoring the actions performed by a malicious user to probe, attack and compromise the decoy system. From this simple concept, different implementations of honeypots can be derived and when two or more honeypots are deployed in a single environment they form a honeynet [21]. A relevant and widely adopted classification of honeypots is based on the *level of interaction*: *low*, *medium* and *high*.

Low interaction honeypots [22] allow for simulating one or more services exposing simple functions without providing access to the operating system. The main benefits in using these solutions are low risk, cost and maintenance. However, they can be easily identified by a human attacker therefore they are not totally reliable and the gathered information can be limited.

High interaction honeypots can include high level functionalities such as, e.g., access to the operating system [22], usage of real devices, etc. Moreover, there exist advanced implementation based on complex virtual environments that allow for emulating both services and devices.

By contrast, medium interaction honeypots represent an intermediate solution, since they provide access to the (possibly emulated) operating system and allow for emulating more services and functionalities than low interaction honeypots.

Although the interaction level represents a key feature, there exist other relevant aspects that characterize honeypots families, such as *purpose*, *role*, *resource level*. A more detailed overview can be found in [23].

Improving threat detection via information sharing and data enrichment. The idea to support the collaboration among different Threat Detection Systems is not new. In Intrusion Detection scenarios, *Collaborative Intrusion Detection Systems* were proposed to improve the effectiveness of (local) IDSs. The idea is that several monitors cooperate in either a hierarchical or peer-to-peer way to discover malicious behaviors, playing both the role of sensors and data collectors. Vasilomanolakis et al. [24] provide a comprehensive survey of the main techniques and solutions proposed in this setting. Although this collaboration allows for improving the performances of singleton IDSs, data trust and privacy represent two relevant issues. Moreover, the missing of a standard data exchange format limits the capabilities of these approaches.

A different approach to improve the accuracy of TDSs consists in integrating information from honeypots. In [25], a hybrid and adaptable honeypot-based approach is proposed that improves the IDSs for protecting networks from intruders. The main idea consists in recording and analyzing the intruder's activities and using the results to take administrative actions for protecting the network. The authors provide an overview of the main components of the systems and illustrate some performance and load testing scenarios.

Baykara and Das [26] introduce a honeypot-based approach aiming at reducing the false positive rate. A honeypot based IDS is implemented able to monitor in real-time the network traffic on specific servers. The system represents a hybrid honeypot combining the capabilities of low and high interaction honeypots in a single structure. The resulting log files are analyzed to identity new zero-day attack and to increase IDS knowledge.

Khosravifar and Bentahar [27] propose a new architecture combining distributed agents and honeypots. The approach aims at reducing the false alarm rate in attack detection. Anomalous connections, initially detected by the IDSs, will be re-routed to a

2 https://crits.github.io/.
3 https://www.eclecticiq.com/platform.
4 https://www.lookingglasscyber.com/.

honeypot network for a more accurate analysis. If, as a result of this investigation, a misclassification of the IDS is discovered, the connection will be redirected to the original destination so that the interaction can be completed.

Sibi Chakkaravarthy et al. [28] illustrate Intrusion Detection Honeypot, an approach combining Honeypots and IDS in a single system. It includes three main components: Honeyfolder, Audit Watch, and Complex Event Processing. The former is a decoy folder exploiting the Social Leopard Algorithm (SoLA), Audit Watch is a module in charge to verify the entropy of the files and folders, and the last allows for aggregating data from different security systems to confirm the ransomware behavior.

Active learning solutions for threat detection. *Active Learning* (AL) refers to a family approaches and algorithms where new instances to be labeled are interactively chosen by means of specific queries [29]. Basically, the idea consists in providing unknown (unlabeled) examples, which can be extracted by adopting different strategies, to an *oracle* that will correctly label them. Active Learning can be used in any scenario where there is a scarcity of labeled data or the latter is highly skewed [30]. In cybersecurity for example, gathering, labeling and sharing data to train any kind of engine/classifier represents a hard and expensive task. As a consequence, Active Learning strategies were recently explored, aimed at improving the effectiveness of the underlying IDS. A comprehensive overview on Active Learning based methods and techniques for detecting anomalous behaviors (e.g., intrusion detection, fault detection, etc.) can be found in [31]. Dang [32] proposes the use of active learning in online configuration to reduce the labeling cost but maintaining the classification performance. Different from other existing active learning algorithms, the author focuses on the labeling of rare events deemed more important for the learning phase. The resulting sampling strategy shows substantial improvements. The adoption of active learning strategies for wireless intrusion detection is studied in [33]. The authors survey the main sampling strategies for AL and then evaluate the improvements due to the usage of the human in the loop approach on an intrusion detection benchmark.

Active learning can be considered as a special case of Reinforcement Learning. Here, the oracle is replaced by the interaction with the environment, where the model is rewarded subject to its ability provide the best action to be performed. Lopez-Martin et al. [34] propose a Deep Reinforcement Learning (DRL) framework able to provide more accurate predictions in comparison with standard machine learning techniques.

While AL is widely considered a consolidated approach, combining AL strategies with DL architectures represents a new relevant research line (i.e., Deep Active Learning). In [29] the authors survey some relevant approaches proposed in this direction.

3. The ORISHA Platform

In this section, we provide an overview of the proposed ORISHA architecture. Fig. 1 illustrates the idea behind the platform. There are essentially three actors that cooperate: *Distributed TIP*, *TDS Layer* and *Honeynet*. The Threat Intelligence Platform represents the core component of the whole architecture. It plays a two-folds role: (i) storing data coming from heterogeneous sources in an encrypted and distributed way; and (ii) delivering the gathered information to the other components. In practice, several MISP instances can cooperate and share data about upcoming threat events from different modules integrated in the platform. The choice of MISP is due to different benefits and capabilities provided by this platform [35,36], as illustrated in Table 1: (*i*) integration with SIEMs and IDSs capability; (*ii*) easily extensible and flexible architecture enabling the possibility to define custom solutions; (*iii*) support for different standards (e.g., STIX, TAXI, etc.); (*iv*) availability of a rich and very detailed documentation; (*v*) a number of active communities, and (*vi*) distributed with open source license.

Different types of Threat Detection Systems (e.g., Intrusion detection and prevention systems) can interface with the TIP by providing information concerning incoming attacks and feeding it with new intrusion events/statistics. In addition, they can exploit information stored in the TIP to improve their models/rules/signatures. We consider a TDS layer here, which can include several tools and techniques for real-time and off-line detection. Honeypots are deployed with the aim to collect additional information concerning new attacks. The deployment can be enabled by TIP, in order to enrich existing events coming either from the IDS or other sources.

The general concept relies essentially on two main components in order to actually enable information sharing: a data exchange format and the layers that orchestrate the communication between TDSs and TIP. In the following, we instantiate such components to work with ML-Based IDSs aimed at detecting anomalous flow connections. We next illustrate how ORISHA enables an Active Learning framework and finally we describe the interaction between the honeynet and the other components. We remark here that the platform can be extended to embed other threat detection systems, provided that suitable adaptation to both the data exchange format and the communication layers is guaranteed.

Notably, in order to make the platform scalable with the number of threats to process, ORISHA can exploit a distributed approach in which several MISP instances (possibly belonging to the same organization) are used together to handle high loads. Moreover, the requests can be replicated on multiple nodes to efficiently process them.

3.1. Sharing threat events: ORISHA data exchange format

The ORISHA components exploit a common interface implemented as a custom MISP Object in JSON format. A MISP object represents the data structure adopted by MISP to store shared threat events. The general template can be easily extended to include further relevant information on specific threat events. Our objective is to devise a custom ORISHA MISP template, which can embed the whole set of data of relevance for a TDS. These include:

- Access to reliable threats, that is, threats for which a general consensus and interpretation can be achieved. The available information can then be used to improve the detection capabilities. For traditional IDS like *Suricata* or *Snort* this is achieved by devising new security policy rules. However data-aware IDS should be able to exploit such information for extending their training sets and rebuilding the underlying ML models with improved accuracy.
- Export of the discovered threats to the MISP network, for further investigation and confirmation analysis. In particular, information concerning threats should allow further labeling/categorization by other actors/TDS accessing it.

The components of the Custom MISP object are illustrated in Table 2. Some of them are particularly relevant for collaborative ML-based Network Intrusion Detection:

- `pcap_file`: This file contains the (anomalous) network activity between two devices in ".pcap" Wireshark format. It is essentially the information exploited by the IDS to detect the threat and as such it is fundamental for the purposes of the sharing philosophy of the platform. In fact, it is the main piece of information to analyze for confirming the threat. Also, in case a confirmation is available, it can be exploited

Table 2
ORISHA MISP object.

Name	Type	Description	Multiple
creation-date	Datetime	Threat event date	×
ip_dst	ip-dst	Destination IP	✓
ip_dst_port	Port	Destination port	✓
ip_src	ip-src	Source IP	✓
pcap_file	Attachment	PCAP file	×
verified	Boolean	*True* if the occurrence of the attack has been verified (e.g., operator check, consensus of a TDS committee, etc.)	×
signature_type	Text	Type of signature (e.g. md5, sha1, etc.)	×
signature	Text	Optional detected file signature	✓
attack_type	Text	A JSON containing information on IDS classification	×
anomaly_details	Attachment	Optional JSON file containing anomaly flow statistics	×

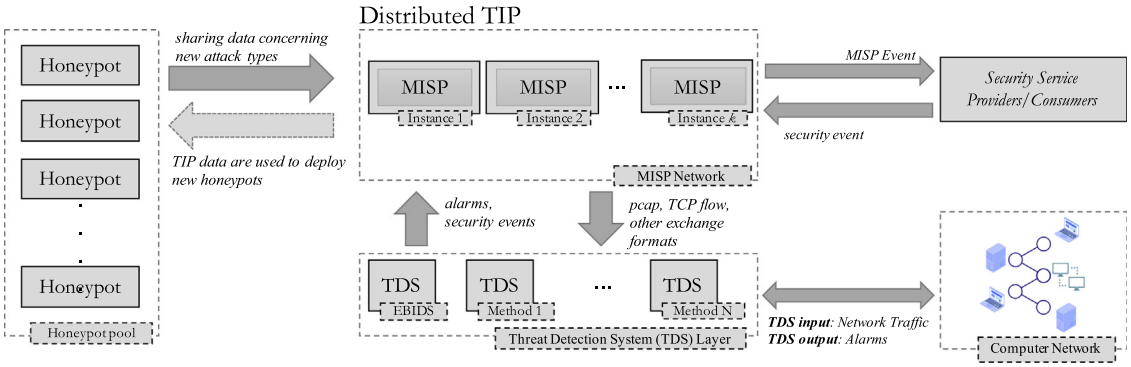


Fig. 1. ORISHA platform.

for enriching the knowledge-base of all the TDS accessing to it. Since it can potentially contain sensitive information, it can be encrypted for secure sharing with trusted-only peers.

- `attack_type`: This field stores, in JSON format, relevant information about the threat. It includes a list of classification events relative to the pcap associated with the MISP object. Each event can be relative to a specific IDS and Multiple IDS can contribute with entries within `attack_type`. For example, if algorithm *X* detected a DOS attack within the pcap file, then `attack_type` includes an entry containing the attack label (DOS), a confidence value/anomaly score, and the signature of the detection algorithm.
- `anomaly_details`: An optional JSON fragment containing aggregate statistics and further information exploited by the IDS for characterizing the threat. For example, it can include some summary statistics extracted from the pcap file, or the signatures used to characterize it.

Fig. 2 shows an example instance. Here, `attack_type` contains JSON instantiated with a specific entry, in lines 34–38. It refers to a specific ML-Based IDS, which relatively to the attached pcap file (`anomaly_pcap`, as declared in the `pcap_file` attribute on lines 39–43), detects an anomalous traffic flow (labeled as `ANOMALY`) with a 98% confidence level.

The protocol only allows representing single events and does not model correlations of diverse events that are possibly related to the same attack. Nevertheless, in Section 4, we show how these data can be used to capture different up-to-date types of attack e.g., DDOS, Bot, Infiltration, etc.

3.2. Integrating TDSs with the TIP

As shown in Fig. 1, the distributed TIP interfaces with a pool of Threat Detection Systems (grouped within the TDS Layer in figure). The latter provide in-the-wild information (e.g. *traffic flow anomalies*, *attack classification*, *malware signatures*, etc.) about anomalies and/or attacks on the monitored system, to be shared


```
{"Object": [{"id": "18919",
    "name": "security_event_object",
    "description": "CS4E Security Event Object",
    "uuid": "ba14028e-03a6-47d2-b35f-8147381493cf",
    "timestamp": "1615454674",
    "Attribute": [{"id": "262154",
      "type": "ip-src",
      "category": "Network activity",
      "object_relation": "ip_src",
      "value": "192.168.1.1"
      },
     {"id": "262155",
      "type": "ip-dst",
      "category": "Network activity",
      "object_relation": "ip_dst",
      "value": "192.168.1.2"
      },
     {"id": "262156",
      "type": "port",
      "category": "Network activity",
      "object_relation": "ip_dst_port",
      "value": "445"
      },
     {"id": "262157",
      "type": "datetime",
      "category": "Other",
      "object_relation": "creation-date",
      "value": "2021-03-11T10:24:34.194148+0000"
      },
     {"id": "262158",
      "type": "text",
      "category": "Other",
      "object_relation": "attack_type",
      "value": {"EBIDS":
        {"version": "0.2",
          "reference": "https://github.com/ebids/",
          "attacks": [{ "attack_type": "ANOMALY", "confidence": "0.98153543"}]}}
        },
     {"id": "262159",
      "type": "attachment",
      "category": "External analysis",
      "object_relation": "pcap_file",
      "value": "anomaly.pcap"
      },
     {"id": "262160",
      "type": "attachment",
      "category": "External analysis",
      "object_relation": "anomaly_details",
      "value": "anomaly.json"
      },
     {"id": "262161",
      "type": "boolean",
      "category": "Other",
      "object_relation": "verified",
      "value": "0"
      }]}]}
```


Fig. 2. An example MISP security event object.

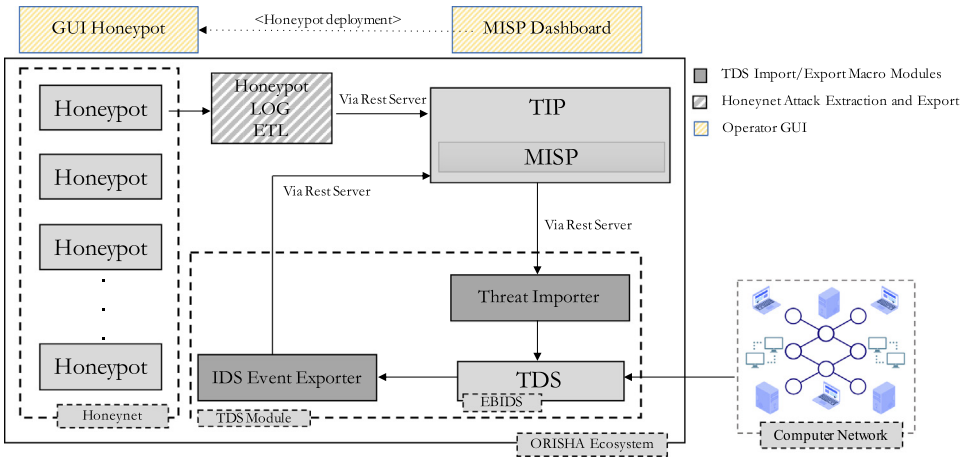


Fig. 3. TDS Integration and honeypot interface.

through the distributed TIP. More in general, single TDSs or TDS layers distributed on several machines and belonging to different organization cooperate by exploiting the TIP to share relevant data on detected anomalies or attacks. Particularly relevant here are the interactions among ML-based threat detection systems, which are detailed as follows.

For the purpose of illustration, we assume that the TDS layer includes EBIDS (Ensemble Based IDS) [37], a ML-based Intrusion Detection technique adopting specialized ensembles of classification models to identify undetected attacks by analyzing traffic flow statistics extracted from network logs. Here, we use pcap format to share network flow information, but the proposed security event object, described in the previous section, is flexible and allows for supporting data shared in other formats. With respect to the proposed architecture, EBIDS requires as input one or more pcap files to compute statistics concerning the network traffic flows. In practice, following a widely adopted approach in the literature [38], EBIDS addresses the (binary) classification problem of deciding whether a given network *connection* (or network *flow*) is associated with an intrusion attack or not. The decision is based on a fixed-length representation of the connection itself, which summarizes and aggregates its features. This representation is exploited during the training phase of the classification ensemble. The learned model allows for discovering anomalous behaviors (potentially related with different types of cyberattacks) that are shared with the other software modules and organizations under the form of MISP security events.

Each TDS can act both as a data provider and consumer. The key for enabling this behavior is the information stored within the security events. As a consumer, a TDS can retrieve custom security events from the MISP, and exploit the enclosed information to improve the detection capabilities of the underlying classification model. This is achieved by feeding it with new labeled data concerning the attacks represented by the security events. As a producer, a TDS can identify/classify new attack instances from which to build and deliver the related security events. Such events can hence be consumed by other TDSs or exploited to deploy specific honeypots tailored for the underlying attacks. Furthermore, it can analyze security events already stored within the MISP, and enrich them with additional `attack_type` labeling.

In Fig. 3, we explicit the interactions among the main components. Basically, three macro-modules realize the import/export facilities: (i) Log Extraction and Transformation (Honeypot Log ETL) module, (ii) Threat Importer and (iii) IDS Event Exporter. The former is aimed at gathering data (e.g. pcap files) from the honeynet and at delivering the associated security events to the MISP. The Threat Importer collects data (pcap files) from the security events and prepares them for the learning phase of EBIDS. Finally, the IDS Event Exporter produces security events discovered by EBIDS and shares them with the MISP. All communications within the platform occur via Rest Server. The values included within the attribute "`attack_type`" are the responsibility of the modules who deliver and/or enrich it, as illustrated in Fig. 2. For example, EBIDS labels each flow (gathered from the network traffic into pcap files via TCPDump and preprocessed) as either normal or attack. Then, a sample of (probably) benign flows (i.e., flows exhibiting an anomaly score less than a threshold value) is used to update the local knowledge base, while abnormal flows are shared with the Distributed TIP. This means that for each abnormal flow (associated with a pcap file), a security event is created with the "`attack_type`" attribute instantiated. We model "`attack_type`" as a list, to ensure flexibility. Thus, each event (and the associated pcap file) can be associated with multiple labels (and the respective degree of confidence). In the example shown in Fig. 2, EBIDS can classify the pcap flow as "`ANOMALY`". The event is shared with MISP and made available to other IDSs which can enrich it with further information. Benign flows do not trigger events, but events already labeled as attacks can also be further labeled as normal by other TDSs: in such cases, the label added to "`attack_type`" will be "`NONE`", with the associated confidence value. This interaction is crucial for ensuring feedback on false positives. A sophisticated example is given by the fragment in Fig. 4. The same pcap file has been analyzed by three different TDSs: two instances of EBIDS (trained with proprietary data from organizations XXX and YYY), and a different IDS (named `OTHERIDS`). The labels associated with the event are: "`ANOMALY`" (with confidence 90%) by the first EBIDS instance, "`DDOS`" (with confidence 70%) and "`SMURF`" (with confidence 100%) by "`OTHERIDS`"; and "`NONE`" (i.e., no anomaly, with confidence 66%) by the second EBIDS instance. In the same vein, intrusions (i.e., unauthorized accesses to the deployed services) logged by honeypots trigger events describing the network flow and the most suitable label (encoded with the "`attack_type`" attribute) triggered by the forensic analysis.

Since EBIDS will be used in Section 4 to demonstrate the capabilities of ORISHA, it is worth looking at the details of its detection model. We next describe the components used to interface the TDSs with the TIP. EBIDS can be considered an overall neural network integrating two components: a number of (weak) deep models sharing the same neural architecture (but trained against different data chunks), and a combiner sub-net employed for yielding the overall anomaly score from the predictions of the base classifiers. In more detail, the base model architecture

```
{"EBIDS": {
    "version": "1.0",
    "reference": "https://github.com/ebids/v1 ,learned by XXX",
    "attacks": [
      { "attack_type": "ANOMALY",
        "confidence": "0.9"}]},
  "OTHERIDS": {
    "version": "2.1",
    "reference": "https://ids.other",
    "attacks": [{
        "attack_type": "DDOS",
        "confidence": "0.7"},
      { "attack_type": "SMURF",
        "confidence": "1.0"}]},
  "EBIDS": {
    "version": "1.1",
    "reference": "https://github.com/ebids/v1.1 ,learned by  YYY",
    "attacks": [
      { "attack_type": "NONE",
        "confidence": "0.66"}]}}
```


Fig. 4. Cooperation example: the same pcap, shared with the TIP, is analyzed by different TDSs.

consists of a stack of different kinds of sub-nets: (*i*) an *input* layer devoted to handling the input i.e. the relational representation of the network flow obtained by extracting the features introduced above; (*ii*) a *feature-engineering* block, consisting in two layers respectively providing an extended view of the initial features (including several non-linear functions) and a compressed representation of the extended view; (*iii*) a number $m > 1$ of *Residual Block* sub-net instances which is devoted to yielding a flexible hierarchy of abstract data features; (*iv*) a final *Decision Layer*, equipped with a *sigmoid* activation function which maps any given data instance to an anomaly score. The outputs of these models are combined by a further sub-net which can adopt different strategies based on trainable or non-trainable functions.

EBIDS works in a continuous stream within two phases: learning and deployment. In the former the classification model is trained based on the data collected from the MISP. Fig. 5(a) shows the logical flow of the learning phase: a set of classifiers are trained against Threat and Regular Traffic flows stored in a local database (named PCAP DB in figure). The software component, named DB Manager, is devoted to filling the PCAP DB with relevant threat events extracted from the TIP and normal traffic flow examples gathered from the computer network. Moreover, it updates the DB by sampling and discarding out-of-date "normal traffic" and possibly also obsolete "threat" data. The examples stored in the PCAP DB are first processed through a data extraction tool (*CICFlowMeter* [39] in the figure) that extracts statistical summaries from the pcap network information. The resulting relational representation then feeds a training set, which is exploited to update EBIDS. The learning phase is periodically performed to keep the model up-to-date.

The deployment phase exploits the learned model to detect anomalous behaviors and store them as MISP security events. We consider a classical Network Intrusion detection scenario: the IDS analyzes network traffic extracted from a network exploiting specific tools (e.g. *packet capture*) with the aim to monitor the devices connected to a network. Fig. 5(b) illustrates the components of the underlying architecture and the typical process: the network flow is stored into a PCAP file and its preprocessed version is provided to EBIDS for the threat detection. If an anomaly (i.e. a possible new threat) is detected, then it is shared with the TIP via the IDS Event Exporter. Both the pcap and the preprocessed data are stored within the event, through the `pcap_file` and `anomaly_details` attributes. Upon request, pcap files imported from MISP security events can be analyzed and an updated MISP Security Event object is created and stored into the TIP with the additional labeling provided by EBIDS.

3.3. Exploiting TDS cooperation and active learning

In this section, we illustrate how different TDSs interfacing with ORISHA can cooperate to improve their detection capabilities and improve digital evidence on threats useful for the decision making process. Fig. 6 exemplifies the whole information flow.

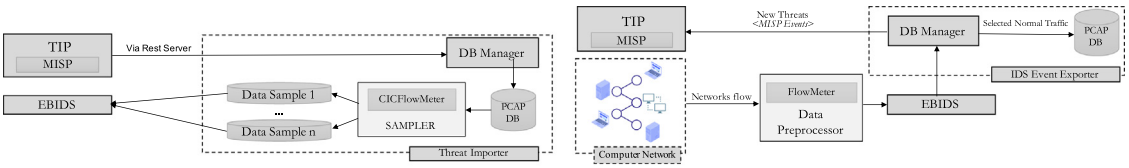
The starting point is represented by the monitored system. Here, traffic flow from the computer network is periodically analyzed by the components of the TDS Layer. In this scenario, a specific anomaly detector (TDS_1 in figure), analyzes the pcap files including a specific logged traffic flow and detects an anomaly. A MISP security event is then produced and delivered. The event is distributed in the TIP, where a different IDS (TDS_2 in figure) accesses the event, analyzes the embedded pcap files and provides an additional labeling (in agreement with the labeling from TDS_1) within the `attack_type` field. The updated MISP object, now with two consensus labels, is analyzed by an expert who validates the threat classification. The validated event can now be retrieved from other IDS (e.g., TDS_n in the figure) and included in their training set for improving the robustness of the underlying classification model.

Now let us consider the situation where the event released by TDS_1 is classified differently (non-anomalous) by TDS_2. Again, the domain expert inspects the event with dissimilar scores and realizes that the event represents a false alarm. The event is then returned to TDS_1 which can then include the validated event in its training set and refine the underlying model for better accuracy and improve its false positive rate.

The above described approach resembles the well-known Query-By-Committee (QbC) strategy [40], with the difference that here, we foresee that the expert validates both the agreement (in the first situation) and the disagreement (in the second one). More sophisticated validation criteria can be adopted, to implement different optimization objectives. For example, in order to reduce the human intervention, automatic validation can be introduced, that confirms the agreements based on confidence values and reduces human analysis to the most uncertain cases based, e.g., on label entropy.

Honeypot-based data enrichment. In addition to the information provided by the TDSs, the framework allows for gathering further attack examples by means of a honeynet so to enrich the knowledge base used for training of the ML-Based TDSs. In more details, each deployed honeypot gathers data concerning new intrusions in a log file (in *JSON* or *XML* format). Notably, a honeypot can be considered an environment where some vulnerabilities are deliberately introduced in order to monitor and gather data on attacks and intrusions. Specifically, they simulate legitimate services which should be normally never called, therefore if a connection with these services is established then it can be labeled as a positive example (i.e. an attack) in the knowledge base.

Honeypot LOG ETL, introduced in Section 3.2, is the component devoted to parsing and analyzing the honeypot log files. Basically, it allows for extracting new threats and interfacing with a MISP instance to make available these new attack data in the TIP. The log files directory of each honeypot is synchronized by means of `rsync` with another one available on the Gateway Router. A TCP Dump Script periodically stores the gathered networks flows as PCAP files. The honeypot's PCAP shared with the distributed TIP can hence be used to enrich the set of positive examples in the learning phase of IDS, as shown in Fig. 3. In ORISHA medium to high interaction honeypots are used to capture the blackhat attack behavior.



(a) Threat Importer module. (b) IDS Event Exporter module.

Fig. 5. TDS Integration example: Learning, deployment and usage of EBIDS.

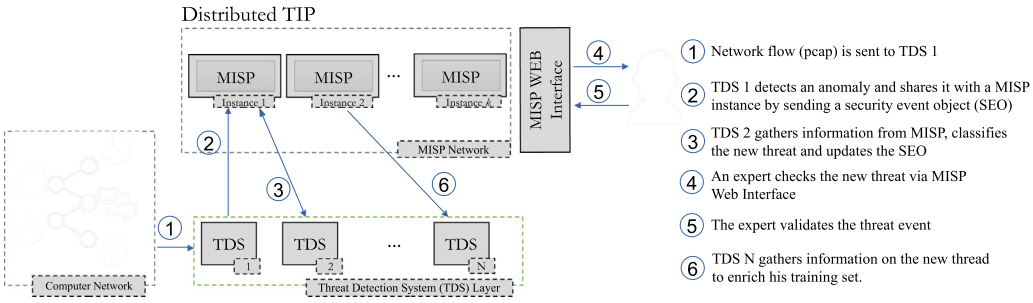


Fig. 6. An example of execution flow.

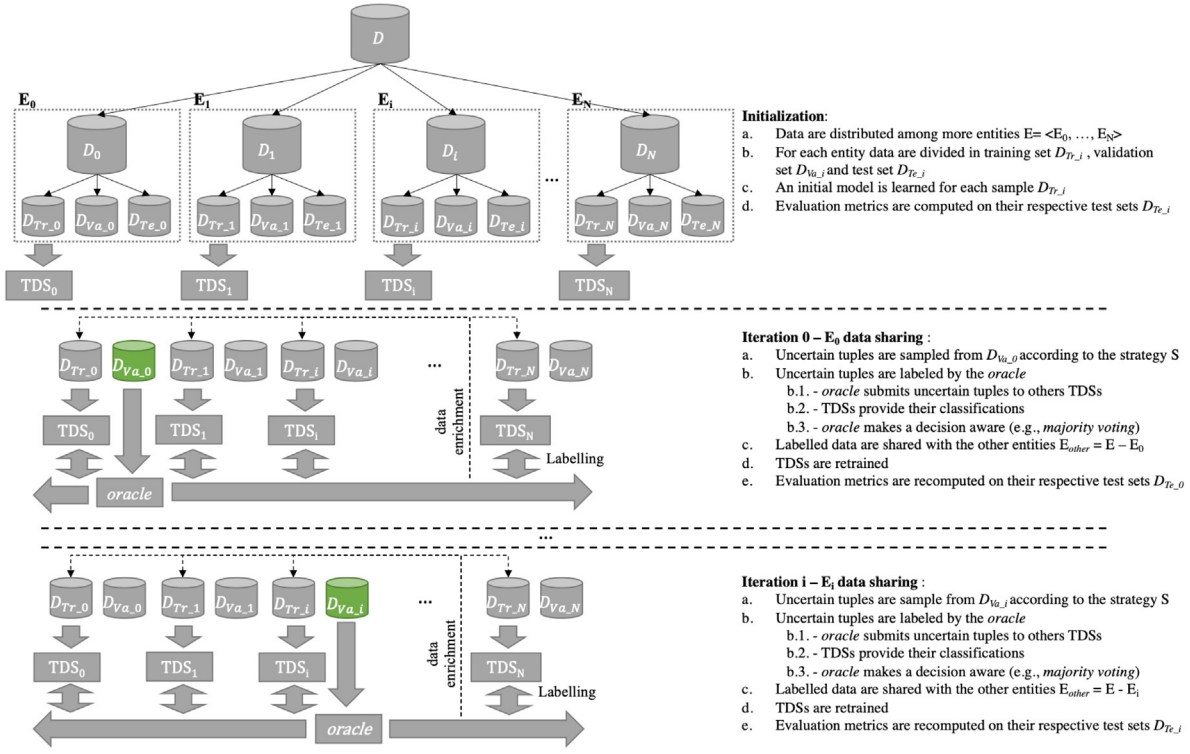


Fig. 7. Evaluation methodology.

4. Experimental analysis

This section illustrates a series of experiments that we conducted to evaluate the effectiveness of our approach. The purpose is to demonstrate that the sharing features of the collaborative platform can be fruitfully exploited to improve the accuracy of each connected peer. ORISHA is a general platform, devised to ensure flexibility and extensibility and address different types of problems. Notwithstanding, in our experimentation we consider a specific supervised intrusion detection scenario where a number of binary classifiers are exploited for discovering intrusions in computer networks. Also, given this context, we do not experiment with honeypots whose main role in the architecture is to enrich the knowledge base with new (unknown) attacks.

4.1. Methodology

The main idea consists in simulating a cooperative environment where different entities share anomalies detected by their own TDSs. This is illustrated in Fig. 7. Each TDS is trained on a specific subset D_i of a global dataset D. The subset D_i relative to entity i is further split in train D_{tr_i}, validation D_{va_i} and test D_{te_i}. The train is the partition that the TDS uses for training and

Table 3
Distribution of the attacks for CICIDS dataset.

Day	Attack families	Number of flows	Percentage of attacks
Monday	*Only legitimate traffic*	529,918	0.000%
Tuesday	*FTP-Patator, SSH-Patator*	445,909	3.103%
Wednesday	*Dos attacks, Heartbleed*	692,703	36.476%
Thursday	*Web attacks, Infiltration*	458,968	0.483%
Friday	*Bot, DDos, Port Scan*	704,245	41.025%

the test is the partition where the evaluation of the TDS takes place. The validation set has a specific role here, representing the "operational data" that the TDS has to inspect and that can in principle be shared through ORISHA. In practice, TDS_i classifies the whole D_{va_i}.

In this scenario, Active Learning represents a natural strategy to improve the overall accuracy of the underlying TDSs: the data gathered from specific TDSs are made available to domain experts via the MISP Network, and they are responsible for verifying uncertain cases and can eventually resolve them. In fact, in Section 3.3 we describe how the interaction takes place. Notice that we are considering an operational scenario, where the incoming flows are unlabeled and hence uncertainty in the response can play a prominent role in deciding which flow to be analyzed by the expert. In fact, as also described in Section 3.3, the approach resembles a Query-By-Committee (QbC) strategy, with the only difference that here we omit to describe the criteria to select and submit the most uncertain tuples.

Models interact with the system in three ways: by submitting events, by labeling events submitted by other models, and by collecting validated events. The validation process is external and delegated to a sovereign expert, and it can be accomplished either automatically (when possible) or manually by human intervention. Of course, human intervention can include forensic analysis and it should be minimized, by devising specific policies based on the degree of disagreement of the responses of the various models.

Thus, for a subset of the malignant flows (i.e., those flows classified as anomalous with confidence $> .5$) some MISP events are created. These events are inspected by the oracle, with the objective of revising their classification. Based on this revision, the newly labeled sample can hence be shared to all the peers and added to $D_{tr_1}, \ldots, D_{tr_N}$ and the underlying ML models are retrained.

We exploit ORISHA to implement a simple oracle strategy that exploits a majority voting. In practice, each event shared through the MISP is classified by all the remaining entities and a final class label (stored within the `attack_type` SEO field) is obtained by combining their scores. This approach departs from the standard QbC strategy, but it allows us to evaluate the overall performance even in a totally automated situation, where a certain degree of uncertainty is still present. In our experiments we tested a number of data sharing iterations equal to the number of entities, which was set to 5 models instantiated.

4.2. Dataset and parameters

The tests were conducted on the CICIDS2017 dataset,[5] [41] a well-known IDS benchmark provided by Canadian Institute of Cybersecurity, containing several up-to-date cyberattacks.

The dataset is partitioned into five samples (one for day), from Monday to Friday and includes six attack profiles based on the last updated list (up to 2017) of common attack families. Specifically, on Monday, only normal flows are recorded, while the other days contain instances of different types of attacks [42], summarized in Table 3. In our experiments we only considered samples containing attacks. Moreover, each data sample is further split in training, validation and test set respectively with 50%/25%/25% percentages. Each model is trained, validated and evaluated against data samples respectively with size of 230k, 115k and 115k instances. For each iteration a data subset ($\sim$15%) of the validation set is sampled uniformly according to the anomaly confidence, and shared through the MISP. The samples thus include both events with a high degree of uncertainty and events which, according to the model, represent attacks with high probability. The automated verification, through a pool of IDSs in charge to make decisions, is particularly well-suited given the large number of examples. However, this strategy can be easily tuned in an operational scenario, by selecting an ad-hoc number of the most uncertain events for the human expert.

We provide an instance of EBIDS to each entity belonging to the network. In the following we will refer these detectors as $TDS_0, \ldots, TDS_N$. These instances are initialized by using the same parameters described in [37]. For the architecture of the base model, (i) the Extended Input layer produces the transformations $\sqrt{x}, x^2, \log(x+1)$ and $sin(x)$ for every original data feature X; (ii) the Embedding Layer maps its input onto a 96-dimensional vector space; (iii) 3 Residual Blocks are used, for a total of 6 Building Blocks; (iv) in every Building Block, the Dense layer consists of 32 neurons, and the dropout rate is set to 0.01. Finally, a *Mixture-Of-Expert* (MOE) [43] is used to combine the different scores provided by the base models. To assess the quality of ORISHA approach (no matter what detection model adopted) we conducted a suite of experiments by varying the ML-techniques used by each TDS.

4.3. Evaluation metrics

Recall and *Precision* are typically used as metrics to estimate the detection capability of Threat Detection Systems since they provide a measurement on their accuracy in identifying upcoming attacks and to avoiding false alarms. Specifically, the former one represents the percentage of real attacks classified as such whereas the second one is the fraction of real attacks correctly discovered by the model among the number of issued alarms. These metrics are frequently combined into the *F-Measure*, computed as their harmonic mean and summarizing the overall performances of the model. Although F-Measure represents a state-of-the-art measure for classification tasks, it is not well suited when the classes are strongly unbalanced. Therefore, we reported two further metrics, respectively named *AUC* and *AUC-PR*, are widely used to estimate the performances of a classification model in presence of skewness on the data (e.g., NIDSs logs).

The *AUC* measures the area under the ROC curve, computed comparing the False Positive Rate (i.e., the percentage of risen false alarms) and the True Positive Rate (i.e., the Recall). The *AUC-PR* (i.e. the area under the Precision–Recall curve) is computed by plotting precision and recall for different class probability values. *AUC-PR* is typically used in each case where the classification of the positive class is more relevant than the negative one, since *AUC* could overestimate the quality of the model.

4.4. Experimental results

A first suite of experiments aims at comparing the behavior of different TDSs when they are feed with further information shared by other entities. After an initialization phase where each TDS has available only a limited sample of data (locally stored),

[5] https://www.unb.ca/cic/datasets/ids-2017.html.

Table 4
Cooperative learning performances (by choosing the better model at each iteration). The values in bold represent an improvement of the model at the correspondent iteration. EBIDS is used as TDS.

Iteration	Model	AUC	AUC-PR	F-Measure
Initialization	TDS_0	0.999	0.997	0.982
	TDS_1	0.832	0.740	0.689
	TDS_2	0.884	0.821	0.784
	TDS_3	1.000	1.000	0.996
	TDS_4	0.932	0.928	0.901
	Average	0.929	0.897	0.870
0	TDS_0	0.999	0.997	0.982
	TDS_1	**0.990**	**0.973**	**0.910**
	TDS_2	0.884	0.821	0.784
	TDS_3	1.000	1.000	0.996
	TDS_4	0.932	0.928	0.901
	Average	**0.961**	**0.944**	**0.915**
1	TDS_0	0.999	0.997	0.982
	TDS_1	0.990	0.973	0.910
	TDS_2	0.884	0.821	0.784
	TDS_3	1.000	1.000	0.996
	TDS_4	0.932	0.928	0.901
	Average	0.961	0.944	0.915
2	TDS_0	0.999	0.997	0.982
	TDS_1	0.990	0.973	0.910
	TDS_2	**0.977**	**0.940**	**0.936**
	TDS_3	1.000	1.000	0.996
	TDS_4	0.932	0.928	0.901
	Average	**0.980**	**0.968**	**0.945**
3	TDS_0	0.999	0.997	0.982
	TDS_1	**0.999**	**0.998**	**0.990**
	TDS_2	**0.999**	**0.998**	**0.988**
	TDS_3	1.000	1.000	0.996
	TDS_4	0.932	0.928	0.901
	Average	**0.986**	**0.984**	**0.971**
4	TDS_0	0.999	0.997	0.982
	TDS_1	0.999	0.998	0.990
	TDS_2	0.999	0.998	0.988
	TDS_3	1.000	1.000	0.996
	TDS_4	0.932	0.928	0.901
	Average	0.986	0.984	0.971

Table 5
Cooperative learning performances (by choosing the better model at each iteration). The values in bold represent an improvement of the model at the correspondent iteration. A Bagging-based detector is used as TDS.

Iteration	Model	AUC	AUC-PR	F-Measure
Initialization	TDS_0	0.540	0.469	0.520
	TDS_1	0.879	0.894	0.910
	TDS_2	0.568	0.616	0.560
	TDS_3	0.640	0.723	0.670
	TDS_4	0.703	0.773	0.750
	Average	0.666	0.695	0.679
0	TDS_0	**0.728**	**0.778**	**0.772**
	TDS_1	0.879	0.894	0.910
	TDS_2	0.568	0.616	0.560
	TDS_3	**0.694**	**0.768**	**0.740**
	TDS_4	0.703	0.773	0.750
	Average	**0.715**	**0.766**	**0.744**
1	TDS_0	0.728	0.778	0.772
	TDS_1	0.879	0.894	0.910
	TDS_2	**0.573**	**0.640**	**0.567**
	TDS_3	0.694	0.768	0.740
	TDS_4	0.703	0.773	0.750
	Average	**0.715**	**0.771**	**0.745**
2	TDS_0	**0.792**	**0.808**	**0.829**
	TDS_1	0.879	0.894	0.910
	TDS_2	**0.648**	**0.721**	**0.677**
	TDS_3	**0.712**	**0.781**	**0.756**
	TDS_4	0.703	0.773	0.750
	Average	**0.747**	**0.795**	**0.783**
3	TDS_0	0.792	0.808	0.829
	TDS_1	0.879	0.894	0.910
	TDS_2	**0.651**	**0.733**	**0.682**
	TDS_3	0.712	0.781	0.756
	TDS_4	0.703	0.773	0.750
	Average	**0.748**	**0.798**	**0.784**
4	TDS_0	0.792	0.808	0.829
	TDS_1	0.879	0.894	0.910
	TDS_2	**0.694**	**0.770**	**0.735**
	TDS_3	**0.714**	**0.782**	**0.758**
	TDS_4	0.703	0.773	0.750
	Average	**0.757**	**0.805**	**0.795**

the first entity E_0 begins to share data sampled from the validation set on the basis of strategy described in Section 4.1. Then, this process is iteratively performed by each entity E_i belonging to the MISP network.

In Table 4, the results of this analysis are reported. First, we can see that after 5 iterations the averaged performances of the TDSs belonging to the network are considerably increased in terms of all evaluation metrics, in particular the initial value of the averaged F-Measure 0.870 increases to 0.971. Notably, some underperforming TDSs (TDS_1 and TDS_2, which exhibit poor predictive performances in the initialization step) benefit substantially from the data sharing and enrichment. In fact, the collaborative framework allows for a substantial improvement of their detection capabilities, enabling the achievement of results comparable to the ones of the other TDSs.

As mentioned above, the same analysis has been conducted by changing the underlying TDS instances. In particular, the additional experiments consider two further ML-based IDSs relying respectively on bagging and boosting ensembles. In both models, a decision tree is used as a weak learner classifier and all the models are set up with default parameters. The results are summarized in Tables 5 and 6. Once again, we can observe an overall improvement of the TDS belonging to the network after 5 iterations and, in particular, the weaker models (i.e., the models with poor performances in the initialization stage) benefit more from threat sharing.

In a further set of experiments, we modify the underlying active learning strategy with a twofold aim. First, we aim at measuring whether a minimal amount of manual labeling can still provide benefits. Second, we would like to compare such benefits to those of the automated approach based on majority voting. In these experiments, a QbC protocol is actually implemented for each iteration, as follows. First of all, we compute the entropy score for each of the events submitted to the MISP and classified by the pool of IDS. Then, we select the 100 events with the highest entropy and add them (with the actual label) to all IDS. This simulates the response by an external expert. The Table 7 summarizes the results of the evaluation. As we can see, the QbC approach still guarantees substantial performance gain for each TDS. Once again, we can see the benefits in terms of predictive accuracy due to the adoption of the ORISHA approach, although the improvements are lesser than the ones obtained with the fully automated solution shown in Table 4.

In the experiments showed so far, each TDS is only aware of the local training set. During the iterations, the newly added examples are labeled according to the majority vote (and hence no new labeled dataset is disclosed). We compared this situation with a flat situation where, for each model, the training set is extended with all the tuples from the local validation set. The results are shown in Fig. 8 . We can see that the accuracy is comparable, even though the models trained through ORISHA only exploit the training set and (unlabeled) portions of the validations sets.

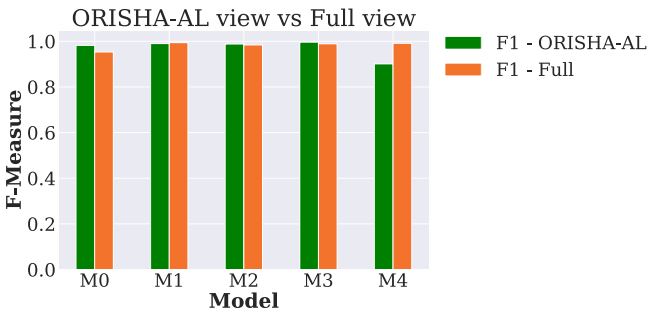


Fig. 8. Comparison between the model performances trained by using ORISHA AL-Based approach and the performance of the ones trained on the union of the training and validation sets (full view).

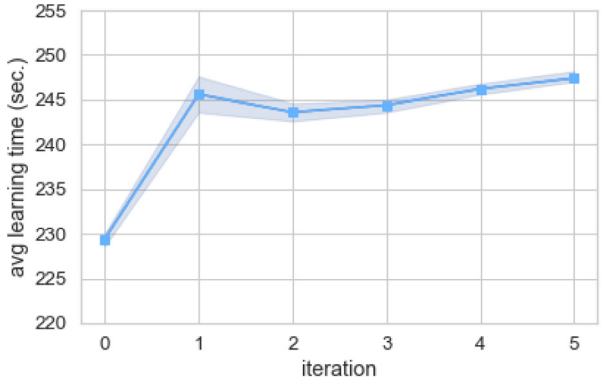


Fig. 9. Average learning time for each iteration.

As regards the computation times of ORISHA for each iteration, they mainly depends on the learning times of the single models and the threat sharing process. The training times computed for Table 4 are essentially constant, as shown in Fig. 9. The overhead for storing or retrieving a single event depends on the specific network latency and thus it does not influence the overall performance, which instead depends on the number of entities and the maximum number of events shared by each entity. As mentioned above, a distributed architecture based on connected MISP instances can scale up the overall process by enabling parallel processing.

5. Conclusions

Security intelligence and data analytics techniques can be used to strengthen the capabilities of cybersecurity applications in various vertical domains and use cases. These techniques can largely benefit from mechanisms to share digital evidence and ensure interoperability. The current Threat Intelligence platforms do not provide native mechanisms to incorporate such mechanisms, especially when data-driven and AI powered threat detection systems are involved. ORISHA is a first attempt to enable a sharing and interoperability protocol among such components, based solely on a data-oriented approach. This simple, flexible strategy and data formats for collaborative threat intelligence can trigger specific advantages: Improving the alert effectiveness by reducing the amount of false positive alerts; better contextualizing threat data with the contribution of multiple actors; boosting trust among producers and consumers of threat intelligence information; and strengthening the robustness of machine learning and deep learning models adopted by security applications.

Table 6
Cooperative learning performances (by choosing the better model at each iteration). The values in bold represent an improvement of the model at the correspondent iteration. A Boosting-based detector is used as TDS.

Iteration	Model	AUC	AUC-PR	F-Measure
Initialization	TDS_0	0.690	0.742	0.734
	TDS_1	0.797	0.850	0.842
	TDS_2	0.730	0.791	0.800
	TDS_3	0.680	0.756	0.718
	TDS_4	0.702	0.768	0.744
	Average	0.721	0.781	0.763
0	TDS_0	0.690	0.742	0.734
	TDS_1	**0.843**	**0.880**	**0.883**
	TDS_2	0.730	0.791	0.800
	TDS_3	**0.731**	**0.795**	**0.776**
	TDS_4	0.702	0.768	0.744
	Average	**0.740**	**0.795**	**0.782**
1	TDS_0	**0.776**	**0.829**	**0.822**
	TDS_1	0.843	0.880	0.883
	TDS_2	0.730	0.791	0.800
	TDS_3	**0.736**	**0.799**	**0.782**
	TDS_4	0.702	0.768	0.744
	Average	**0.758**	**0.814**	**0.801**
2	TDS_0	**0.849**	**0.885**	**0.888**
	TDS_1	0.843	0.880	0.883
	TDS_2	0.730	0.791	0.800
	TDS_3	**0.737**	**0.801**	**0.783**
	TDS_4	0.702	0.768	0.744
	Average	**0.772**	**0.825**	**0.815**
3	TDS_0	0.849	0.885	0.888
	TDS_1	0.843	0.880	0.883
	TDS_2	0.730	0.791	0.800
	TDS_3	0.737	0.801	0.783
	TDS_4	0.702	0.768	0.744
	Average	0.772	0.825	0.815
4	TDS_0	**0.852**	**0.887**	**0.890**
	TDS_1	0.843	0.880	0.883
	TDS_2	0.730	0.791	0.800
	TDS_3	0.737	0.801	0.783
	TDS_4	0.702	0.768	0.744
	Average	**0.773**	**0.826**	**0.816**

An experimental evaluation, conducted on a well-known IDS benchmark, demonstrates how merging data sharing and active learning strategies can improve the detection capabilities of the MISP network allowing to discover undetected attacks.

Novelty and limitations. Although the design principles of ORISHA are general, the current implementation focuses on sharing digital evidence among Intrusion Detection Systems. Moreover, the idea of sharing data to fit different models in a decentralized fashion is adopted by other emerging frameworks. As an example, *Federated Learning* [44] (FL) is a family of methods and techniques enabling the learning of ML models in decentralized way, usually performed on edge devices or servers storing data. By fostering the cooperation of different actors, FL allows for learning robust and effective predictive models without the need to share data in a centralized way. However, many FL frameworks employ multiple rounds of communication between devices and the central server, which increases the communication overheads. In particular, while FL is mainly devised to distribute the computation on more devices so to make scalable and feasible the learning even when the computational resources are not sufficient to handle the massive amount of yielded data, the main objective of ORISHA is to share IoCs that can be used by domain experts to take aware decisions and enable effective countermeasures, and also by automatized ML-Based systems to improve their predictive performances.

Table 7
Full active learning approach. A Query By Committee approach exploiting entropy-based criterion is used to select (∼100) uncertain tuples for the expert verification. EBIDS is used as TDS.

Iteration	Model	AUC	AUC-PR	F-Measure
Initialization	TDS_0	0.999	0.997	0.982
	TDS_1	0.832	0.740	0.689
	TDS_2	0.884	0.821	0.784
	TDS_3	1.000	1.000	0.996
	TDS_4	0.932	0.928	0.901
	Average	0.929	0.897	0.870
0	TDS_0	**0.999**	**0.997**	**0.988**
	TDS_1	**0.883**	**0.782**	**0.719**
	TDS_2	**0.972**	**0.944**	**0.911**
	TDS_3	1.000	1.000	0.996
	TDS_4	**0.934**	**0.921**	**0.902**
	Average	**0.958**	**0.929**	**0.902**
1	TDS_0	0.999	0.997	0.988
	TDS_1	0.883	0.782	0.719
	TDS_2	0.972	0.944	0.911
	TDS_3	1.000	1.000	0.996
	TDS_4	0.934	0.921	0.902
	Average	0.958	0.929	0.902
2	TDS_0	0.999	0.997	0.988
	TDS_1	**0.856**	**0.769**	**0.765**
	TDS_2	0.972	0.944	0.911
	TDS_3	1.000	1.000	0.996
	TDS_4	0.934	0.921	0.902
	Average	**0.952**	**0.926**	**0.912**
3	TDS_0	0.999	0.997	0.988
	TDS_1	**0.944**	**0.908**	**0.847**
	TDS_2	0.972	0.944	0.911
	TDS_3	1.000	1.000	0.996
	TDS_4	0.934	0.921	0.902
	Average	**0.970**	**0.954**	**0.928**
4	TDS_0	0.999	0.997	0.988
	TDS_1	0.944	0.908	0.847
	TDS_2	0.972	0.944	0.911
	TDS_3	1.000	1.000	0.996
	TDS_4	0.934	0.921	0.902
	Average	**0.970**	**0.954**	**0.929**

A different approach to exploit gathered knowledge is adopted by *Transfer Learning* (TL). In this case the idea consists in re-using deep models, learned on a different domain or for tackling a different task, by fine-tuning them for addressing the own learning problem. This approach perfectly marries the ORISHA philosophy since the models learned by the organizations belonging to the MISP network are themselves relevant IoCs. As an example, in [45], an interesting study illustrates a solution based on the integration of MISP, TheHIVE and Cortex devised to enable the sharing of ML Models in a MISP network.

Privacy aspects. Trust and privacy are two major issues in Threat Information Sharing. Due to the sensitive nature of the involved data, targets of cyberthreats are not always inclined to share this information unless they are obliged to be compliant with mandatory incident reporting regulations. Although these aspects are not the main focus of the work, these issues can be addressed by adopting recent approaches proposed in literature, which can be easily integrated with MISP (and hence be exploited by ORISHA). For example, in [46,47] the authors illustrate TATIS (Trustworthy APIs for Threat Intelligence Sharing with User Managed Access (UMA) and CP-ABE), a solution for fine-grained access control to protect threat intelligence APIs using UMA and Ciphertext-Policy Attribute-Based Encryption (CP-ABE). TATIS services are made available under the form of REST API and can be invoked by each organization belonging to the sharing network to strengthen the sharing process of ORISHA in terms of privacy, trust and reliability. All these complementary services (e.g., *privacy preserving*, *risk assessment*, etc.), which can be included and make ORISHA fully operational, are already represented in Fig. 1 with the name "*Security Service Providers/Consumers*".

Future works. We plan to investigate the integration of additional data-aware threat analysis tools (e.g., Malware detection systems which do not necessarily rely on the analysis of network flow). Also, mechanisms for incorporating and sharing higher-level information, such as machine learning models, can further boost security intelligence capabilities. Some initial attempts have been proposed [45]. However, the lack of a common representation format, make this task challenging and worth further studies.

Finally, the adoption of Deep Active and Deep Reinforcement Learning requires further investigation since can improve the performances by simultaneously reducing the expensive interaction with the domain expert.

CRediT authorship contribution statement

Massimo Guarascio: Conceptualization, Methodology, Investigation, Writing – original draft, Writing – review & editing. **Nunziato Cassavia:** Conceptualization, Methodology, Investigation, Writing – original draft, Software. **Francesco Sergio Pisani:** Conceptualization, Investigation, Writing – review & editing. **Giuseppe Manco:** Conceptualization, Methodology, Investigation, Writing - original draft, Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

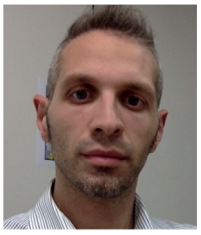
Acknowledgments

This work has been partially supported by EU H2020-SU-ICT-03-2018 Project No. 830929 CyberSec4Europe (cybersec4europe.eu).

References

[1] ENISA, Enisa threat landscape 2020 - list of top 15 threats, 2020, https://www.enisa.europa.eu/publications/enisa-threat-landscape-2020-list-of-top-15-threats.
[2] Interpol, Covid-19 cybercrime analysis report, 2020, https://tinyurl.com/6wek2rk.
[3] M.D.T.I. Team, Exploiting a crisis: How cybercriminals behaved during the outbreak, 2020, https://tinyurl.com/cybercrime-during-outbreak.
[4] C.S. Johnson, M.L. Badger, D. Waltermire, J. Snyder, C. Skorupka, Guide to cyber threat information sharing, NIST Special Publ. 80 (2016) 0–150.
[5] S. Brown, J. Gommers, O. Serrano, From cyber security information sharing to threat management, in: Proceedings of the 2nd ACM Workshop on Information Sharing and Collaborative Security, 2015, pp. 43–49, http://dx.doi.org/10.1145/2808128.2808133.
[6] L. Dandurand, O.S. Serrano, Towards improved cyber security information sharing, in: 2013 5th International Conference on Cyber Conflict (CYCON 2013), 2013, pp. 1–16.
[7] A. Zibak, A. Simpson, Cyber threat information sharing: Perceived benefits and barriers, in: Proceedings of the 14th International Conference on Availability, Reliability and Security, Association for Computing Machinery, 2019, http://dx.doi.org/10.1145/3339252.3340528.
[8] D.W. Chadwick, W. Fan, G. Costantino, R. de Lemos, F. Di Cerbo, I. Herwono, M. Manea, P. Mori, A. Sajjad, X.S. Wang, A cloud–edge based data security architecture for sharing and analysing cyber threat information, Future Gener. Comput. Syst. 102 (2020) 710–722, http://dx.doi.org/10.1016/j.future.2019.06.026.
[9] S. Qamar, Z. Anwar, M.A. Rahman, E. Al-Shaer, B.T. Chu, Data-driven analytics for cyber-threat intelligence and information sharing, Comput. Secur. 67 (2017) 35–58, http://dx.doi.org/10.1016/j.cose.2017.02.005.

[10] T.D. Wagner, K. Mahbub, E. Palomar, A.E. Abdallah, Cyber threat intelligence sharing: Survey and research directions, Comput. Secur. 87 (2019) 101589, http://dx.doi.org/10.1016/j.cose.2019.101589.
[11] V. Mavroeidis, S. Bromander, Cyber threat intelligence model: An evaluation of taxonomies, sharing standards, and ontologies within cyber threat intelligence, in: 2017 European Intelligence and Security Informatics Conference (EISIC), 2017, pp. 91–98, http://dx.doi.org/10.1109/EISIC.2017.20.
[12] C. Sauerwein, C. Sillaber, A. Mussmann, R. Breu, Threat intelligence sharing platforms: An exploratory study of software vendors and research perspectives, Wirtschaftsinformatik Angew. Inform. (2017).
[13] M.C. Libicki, Sharing Information about Threats is not a Cybersecurity Panacea, RAND Corporation, Santa Monica, CA, 2015.
[14] B. Jordan, R. Piazza, T. Darley, Stix™version 2.1 committee specification 01, 2020.
[15] T. Darley, I. Kirillov, R. Piazza, D. Beck, Cybox™version 2.1.1. part 01: Overview - committee specification draft 01/ public review draft 01, 2016.
[16] R. Danyliw, J. Meijer, Y. Demchenko, The incident object description exchange format, in: RFC 5070 (Proposed Standard), 2007.
[17] T. Darley, I. Kirillov, R. Piazza, D. Beck, Taxii™version 2.1 committee specification 01, 2020.
[18] ENISA, Exploring the opportunities and limitations of current threat intelligence platforms, 2017.
[19] C. Wagner, A. Dulaunoy, G. Wagener, A. Iklody, Misp: The design and implementation of a collaborative threat intelligence sharing platform, in: Proceedings of the 2016 ACM on Workshop on Information Sharing and Collaborative Security, Association for Computing Machinery, 2016, pp. 49–56, http://dx.doi.org/10.1145/2994539.2994542.
[20] M. Goffin, Crits: Collaborative research into threats, 2014, https://crits.github.io/. [Online].
[21] M. Bringer, C. Chelmecki, H. Fujinoki, A survey: Recent advances and future trends in honeypot research, Int. J. Comput. Netw. Inform. Secur. (4) (2012) http://dx.doi.org/10.5815/ijcnis.2012.10.07.
[22] M.F. Razali, M.N. Razali, F.Z. Mansor, G. Muruti, N. Jamil, Iot honeypot: A review from researcher's perspective, in: 2018 IEEE Conference on Application, Information and Network Security (AINS), 2018, pp. 93–98, http://dx.doi.org/10.1109/AINS.2018.8631494.
[23] J. Franco, A. Aris, B. Canberk, A. Selcuk Uluagac, A survey of honeypots and honeynets for internet of things, industrial internet of things, and cyber–physical systems, 2021, CoRR abs/2108.02287.
[24] E. Vasilomanolakis, S. Karuppayah, M. Mühlhäuser, M. Fischer, Taxonomy and survey of collaborative intrusion detection, ACM Comput. Surv. (47) (2015) http://dx.doi.org/10.1145/2716260.
[25] H. Artail, H. Safa, M. Sraj, I. Kuwatly, Z. Al-Masri, A hybrid honeypot framework for improving intrusion detection systems in protecting organizational networks, Comput. Secur. 25 (2006) 274–288, http://dx.doi.org/10.1016/j.cose.2006.02.009.
[26] M. Baykara, R. Das, A novel honeypot based security approach for real-time intrusion detection and prevention systems, J. Inform. Secur. Appl. 41 (2018) 103–116, http://dx.doi.org/10.1016/j.jisa.2018.06.004.
[27] B. Khosravifar, J. Bentahar, An experience improving intrusion detection systems false alarm ratio by using honeypot, in: 22nd International Conference on Advanced Information Networking and Applications (Aina 2008), 2008, pp. 997–1004, http://dx.doi.org/10.1109/AINA.2008.44.
[28] S. Sibi Chakkaravarthy, D. Sangeetha, M.V. Cruz, V. Vaidehi, B. Raman, Design of intrusion detection honeypot using social leopard algorithm to detect iot ransomware attacks, IEEE Access 8 (2020) 169944–169956, http://dx.doi.org/10.1109/ACCESS.2020.3023764.
[29] P. Ren, Y. Xiao, X. Chang, P. Huang, Z. Li, B. Gupta, X. Chen, X. Wang, A survey of deep active learning, ACM Comput. Surv. (2021) 54, http://dx.doi.org/10.1145/3472291.
[30] B. Settles, Active Learning. Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan & Claypool Publishers, 2012, http://dx.doi.org/10.2200/S00429ED1V01Y201207AIM018.
[31] A. Shahraki, M. Abbasi, A. Taherkordi, A.D. Jurcut, Active learning for network traffic classification: A technical survey, 2021, arXiv:2106.06933.
[32] Q.V. Dang, Active learning for intrusion detection systems, in: 2020 RIVF International Conference on Computing and Communication Technologies (RIVF), 2020, pp. 1–3, http://dx.doi.org/10.1109/RIVF48685.2020.9140751.
[33] K. Yang, J. Ren, Y. Zhu, W. Zhang, Active learning for wireless iot intrusion detection, IEEE Wirel. Commun. 25 (2018) 19–25, http://dx.doi.org/10.1109/MWC.2017.1800079.
[34] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, Application of deep reinforcement learning to intrusion detection for supervised problems, Expert Syst. Appl. 141 (2020) 112963, http://dx.doi.org/10.1016/j.eswa.2019.112963.
[35] G. González-Granadillo, M. Faiella, I. Medeiros, R. Azevedo, S. González-Zarzosa, Etip: An enriched threat intelligence platform for improving osint correlation, analysis, visualization and sharing capabilities, J. Inform. Secur. Appl. 58 (2021) 102715, http://dx.doi.org/10.1016/j.jisa.2020.102715.
[36] W. Tounsi, H. Rais, A survey on technical threat intelligence in the age of sophisticated cyber attacks, Comput. Secur. 72 (2018) 212–233, http://dx.doi.org/10.1016/j.cose.2017.09.001.
[37] F. Folino, G. Folino, M. Guarascio, F. Pisani, L. Pontieri, On learning effective ensembles of deep neural networks for intrusion detection, Inf. Fusion 72 (2021) 48–69, http://dx.doi.org/10.1016/j.inffus.2021.02.007.
[38] M.H. Bhuyan, D.K. Bhattacharyya, J.K. Kalita, Network anomaly detection: methods, systems and tools, IEEE Commun. Surv. Tutor. 16 (2013) 303–336.
[39] A.H. Lashkari, G.D. Gil, M.S.I. Mamun, A.A. Ghorbani, Characterization of tor traffic using time based features, in: Proceedings of the 3rd International Conference on Information Systems Security and Privacy - Volume 1: ICISSP, SciTePress., 2017, pp. 253–262, http://dx.doi.org/10.5220/0006105602530262.
[40] D.A. Cohn, L.E. Atlas, R.E. Ladner, Improving generalization with active learning, Mach. Learn. 15 (1994) 201–221, http://dx.doi.org/10.1007/BF00993277.
[41] I. Sharafaldin, A. Habibi Lashkari, A.A. Ghorbani, Toward generating a new intrusion detection dataset and intrusion traffic characterization, in: Proceedings of the 4th International Conference on Information Systems Security and Privacy - ICISSP, INSTICC, SciTePress, 2018, pp. 108–116, http://dx.doi.org/10.5220/0006639801080116.
[42] R. Panigrahi, S. Borah, A detailed analysis of CICIDS2017 dataset for designing intrusion detection systems, Int. J. Eng. Technol. 7 (2018) 479–482.
[43] S. Masoudnia, R. Ebrahimpour, Mixture of experts: a literature survey, Artif. Intell. Rev. 42 (2014) 275–293.
[44] T. Li, A.K. Sahu, A. Talwalkar, V. Smith, Federated learning: Challenges, methods, and future directions, IEEE Signal Process. Mag. 37 (2020) 50–60, http://dx.doi.org/10.1109/MSP.2020.2975749.
[45] D. Preuveneers, W. Joosen, Sharing machine learning models as indicators of compromise for cyber threat intelligence, J. Cybersecur. Privacy 1 (2021) 140–163, http://dx.doi.org/10.3390/jcp1010008, https://www.mdpi.com/2624-800X/1/1/8.
[46] D. Preuveneers, W. Joosen, Tatis: Trustworthy apis for threat intelligence sharing with uma and cp-abe, in: A. Benzekri, M. Barbeau, G. Gong, R. Laborde, J. Garcia-Alfaro (Eds.), Foundations and Practice of Security, Springer International Publishing, Cham, 2020, pp. 172–188.
[47] D. Preuveneers, W. Joosen, J.B. Bernabé, A.F. Skarmeta, Distributed security framework for reliable threat intelligence sharing, Secur. Commun. Netw. (2020) 8833765:1–8833765:15, http://dx.doi.org/10.1155/2020/8833765.



Massimo Guarascio holds a Ph.D. in Systems and Computer Science Engineering and a master's degree in Computer Science Engineering, both from the University of Calabria. He is currently researcher at the Institute for High Performance Computing and Networking of the National Research Council (ICAR-CNR) and shareholder of OKT s.r.l., a spin-off of University of Calabria. He co-authored over 50 papers published in international conference proceedings, chapters and journals. His research mainly focuses on machine learning, anomaly detection and explanation, process mining, data analytics methods for geosciences and remote sensing, knowledge discovery and data mining for cybersecurity and fraud detection. He has participated in European and national research projects concerning machine learning and cybersecurity.

Nunziato Cassavia holds a Ph.D. and an M.Sc. in Computer Engineering from the University of Calabria, Italy. Currently, he is a research fellow at the Institute for High Performance Computing and Networking (ICAR-CNR) of the National Research Council (CNR) in Italy. His research interests include Cybersecurity, Big Data, Distributed Computing System, Data Warehouse and Relational and NoSQL Databases. He co-authored several papers published in international conference proceedings, chapters and journals and participated in European and national research projects concerning Big Data and Cybersecurity.

Francesco Sergio Pisani holds a Ph.D. and an M.Sc. in Computer Engineering from the University of Calabria, Italy. Currently, he is a research fellow at the Institute for High Performance Computing and Networking (ICAR-CNR) of the Italian National Research Council. His research mainly focuses on machine and deep learning, recommender system, ensemble learning, knowledge discovery and data mining for cybersecurity, computer vision, anomaly detection. Currently he is involved in national projects concerning machine learning and cybersecurity, and on an industrial project to build an advanced platform for customer engagement e satisfaction.

Giuseppe Manco graduated summa cum laude in computer science and received a Ph.D. degree in computer science from the University of Pisa. He is currently a Director of Research at the Institute for High Performance Computing and Networking (ICAR-CNR) of the National Research Council of Italy and a contract professor at University of Calabria, Italy. His current research interests include knowledge discovery and data mining, Recommender Systems and Social Network Analysis, Deep Learning. He has been the coordinator of several national and international research projects. He has been serving in the program committee of several international/national conferences, including: IJCAI, AAAI, IEEE ICDM, ECMLPKDD, SIAM SDM, PAKDD. He served as a program co-chair of ECML-PKDD 2016. He is serving as an associate editor for the Journal of Intelligent Information Systems, Knowledge and Information Systems and Machine Learning Journal.