Journal of
Cybersecurity and Privacy

MDPI

Article

Sharing Machine Learning Models as Indicators of Compromise for Cyber Threat Intelligence

Davy Preuveneers * and Wouter Joosen

imec—DistriNet, KU Leuven, Celestijnenlaan 200A, B-3001 Heverlee, Belgium; wouter.joosen@kuleuven.be
* Correspondence: davy.preuveneers@kuleuven.be; Tel.: +32-16-327853

Received: 18 January 2021; Accepted: 22 February 2021; Published: 26 February 2021

Abstract: Cyber threat intelligence (CTI) sharing is the collaborative effort of sharing information about cyber attacks to help organizations gain a better understanding of threats and proactively defend their systems and networks from cyber attacks. The challenge that we address is the fact that traditional indicators of compromise (IoC) may not always capture the breath or essence of a cyber security threat or attack campaign, possibly leading to false alert fatigue and missed detections with security analysts. To tackle this concern, we designed and evaluated a CTI solution that complements the attribute and tagging based sharing of indicators of compromise with machine learning (ML) models for collaborative threat detection. We implemented our solution on top of MISP, TheHive, and Cortex—three state-of-practice open source CTI sharing and incident response platforms—to incrementally improve the accuracy of these ML models, i.e., reduce the false positives and false negatives with shared counter-evidence, as well as ascertain the robustness of these models against ML attacks. However, the ML models can be attacked as well by adversaries that aim to evade detection. To protect the models and to maintain confidentiality and trust in the shared threat intelligence, we extend our previous research to offer fine-grained access to CP-ABE encrypted machine learning models and related artifacts to authorized parties. Our evaluation demonstrates the practical feasibility of the ML model based threat intelligence sharing, including the ability of accounting for indicators of adversarial ML threats.

Keywords: threat intelligence sharing; security automation; trust

1. Introduction

Organizations are facing an accelerating wave of sophisticated attacks by cyber criminals that disrupt services, exfiltrate sensitive data, or abuse victim machines and networks for performing other malicious activities. These malicious adversaries aim to steal, destroy, or compromise business assets with a particular financial, reputational, or intellectual value.

To address such security challenges and eliminate security blind spots for their systems and networks, on-premise services, and cloud environments, organizations are complementing their perimeter defenses with threat intelligence platforms. Cyber threat intelligence (CTI) [1–4] is any information that can help organizations identify, assess, monitor, and respond to cyber threats. CTI can help increase the visibility into cyber threats and policy violations within the organization, aid with security-critical decisions, decrease the time to detection of both known and unknown threats, respond more effectively to an ongoing attack, or help prevent an attack from happening in the first place by sharing threat intelligence information with trusted third party organizations and communities.

To swiftly detect and thwart cyber threats, CTI platforms implement data-driven processes that ingest, analyze, and respond to a variety of cyber security data sources, including application log files, monitoring platforms, suspicious binaries, system and network events, and other threat intelligence feeds. CTI platforms have been increasingly adopting advanced data analytics, machine learning, and

deep learning methods to sift through large amounts of structured and unstructured cyber security data. For example, artificial intelligence (AI) methods are employed not only to efficiently and effectively detect and recognize threats based on low-level data and shared indicators of compromise (IoC), but also to augment that information into enriched security incident knowledge with a higher level of abstraction that is more suitable for human consumption, and to triage the high volume of incident reports by filtering irrelevant ones to mitigate alert fatigue [5,6] with security analysts.

While many commercial solutions are readily available to improve an organization's incident response and remediation strategies, recent research [7,8] indicates that these solutions are not only fairly expensive, but also that there is a limited overlap amongst one another w.r.t. the threat intelligence information that they provide, even when comparing the same cyber threat campaigns. In other words, the IoCs that these CTI solutions offer, may not always capture the full breath or essence of a cyber security threat or attack campaign, possibly leading to an abundance of false positive alerts and missed detections.

To address these concerns, we propose a framework that shares the machine learning (ML) models themselves for threat detection within security communities, and not just the results of the evaluation of these models. The advantage of sharing ML models compared to IoCs is that the former offer a more sophisticated and effective—albeit computationally more demanding—method to identify attacks, as IoCs typically characterize a threat event as a simple list of tagged and annotated attributes (e.g., the IP address of the attacker) that are possibly correlated with other threat events. The value of IoCs may also deteriorate over time. Our approach is more robust than IoCs against trivial evasion tactics by adversaries, such as an attacker moving to a different IP address to bypass a target's intrusion detection and prevention system (IDS/IPS). Obviously, the AI methods that underpin CTI platforms can be collaboratively strengthened and improved by the security community by providing examples of misclassifications, and new or improved versions of the ML models.

However, our approach of sharing ML models also comes with security and privacy threats that need to be mitigated. First of all, to address misclassifications by sharing counter examples, organizations are not always willing to share sensitive information about attacks that they have encountered. Additionally, the ML models are not meant for human consumption and they often operate as black boxes. Not only is it sometimes unclear how and why an AI algorithm makes certain decisions [9], recent developments in the area of adversarial machine learning make it more practically feasible to construct adversarial examples [10] that allow evading detection. As such, the underlying ML models themselves can be vulnerable and increase the attack surface if not robustly trained and protected. We therefore cryptographically protect the way these ML models are shared within security communities. Only authorized parties within the community can decrypt and leverage the models, but also contribute by enriching these models with false positive and false negative inputs or adversarial examples, or by incrementally fine-tuning these models using their own data if they are not willing to share any sensitive information.

To summarize, the goal of our research is to provide a framework that enables the sharing of ML models within security communities to make cyber threat detection more effective, and to also protect models against security threats. Although we will demonstrate our approach with prototypical examples of ML models, the objective of our work is not to optimize the accuracy or robustness of a single ML model targeting a single security use case (e.g., intrusion detection and prevention, malware classification, user and entity behavior analysis, etc.). The key contributions of our research are the following:

1. We present a framework for sharing ML models for threat detection that operates on top of state-of-the-art open source cyber threat intelligence platforms, i.e., MISP, TheHive, and Cortex;
2. We propose a threat taxonomy to annotate ML models and associated artifacts with tags that indicate the presence of adversarial ML threats;

3. We build upon previous work to protect shared ML models with Ciphertext-Policy Attribute-Based Encryption (CP-ABE) such that only authorized parties within a security community can leverage them;
4. We demonstrate the practical feasibility of our framework by means of prototypical examples of ML models, and evaluating the impact of our methods from a qualitative and quantitative perspective.

The remainder of this paper is structured as follows. In Section 2, we review relevant related work on threat intelligence. The design and implementation of our solution to use and improve machine learning models for DDoS detection as complementary IoCs is described in Section 3. We evaluate the practical feasibility of our solution in Section 4. We conclude in Section 5 summarizing the main insights and identifying opportunities for further research.

2. Related Work

In this section, we review relevant related research in three different areas, including (1) security best practices and limitations of cyber threat intelligence sharing, (2) the use of machine learning and deep learning models to detect and recognize cyber security threats, as well as (3) the increased attack surface when adopting machine learning to detect such cyber threats.

2.1. Cyber Threat Intelligence Sharing

To enhance detection and prevention of cyber threats, organizations collaborate to define defensive actions against complex attack vectors by sharing information about threats, such as IoCs. IoCs are pieces of forensic data that detail illicit or malicious activities on a system or network. Examples include the IP address of an attacker, the domain name involved in a phishing campaign, or the hash of a malware binary. To prevent the same security incident from happening elsewhere, enabling technologies are used to collect, examine, analyze, and possibly share digital evidence originating from a variety of digital data sources. Structured standards, such as Structured Threat Information eXpression (STIX) [11], Trusted Automated eXchange of Indicator Information (TAXII) [12], and Cyber Observable eXpression (CybOX) [13], have been proposed as a means to simplify the sharing of such cyber threat intelligence, and they have become de facto industry standards.

Ramsdale et al. [14] analyzed the capabilities of the aforementioned CTI formats to convey various CTI types, and also reviewed a variety of internal CTI sources (e.g., systems logs and network events, traffic profiles, security monitoring applications and systems, forensic analysis), and externally sourced CTI feeds and open source intelligence (OSINT). The authors indicate that while support for STIX is apparent in many platforms, they observe a trend towards directly using the APIs offered by the platforms and custom JSON or platform-specific formats whenever they appear to be a better fit for the use case at hand. Compared to the volume, the quality of the CTI feeds is equally if not more important. The authors found that most of the issues they encountered with the quality and the distribution of CTI are due to missing information, such as the time of collection and the origin of the threat data.

Li et al. [7] proposed various metrics to systematically characterize a broad range of public and commercial threat intelligence feeds. They came to the conclusion that many CTI vendors do not explain well their methodology to collect data, and that the data categorization into different classes can be rather ambiguous. Furthermore, larger feeds do not necessarily contain better data. Also, the authors found that there is little overlap in the different data sources they analyzed, and that acquiring different CTI sources is unlikely to provide better coverage and may even cause collateral damage due to false positives.

Similar research on commercial threat intelligence providers was carried out by Bouwman et al. [8]. They executed an empirical assessment of the indicators of two leading vendors, and they found almost no overlap between them nor with four large open threat intelligence feeds. Furthermore, the small number of overlapping indicators only show up in the feed of the other vendor with a delay

of about one month. As such, the authors challenge the coverage and timeliness of expensive threat intelligence feeds.

The above observations indicate that different CTI feeds may not always capture the full breath or essence of a cyber security threat or attack campaign, possibly leading to an abundance of false positive alerts and missed attacks.

2.2. Indicators of Compromise versus Machine Learning for Threat Detection

One of the main use cases for cyber threat intelligence, as confirmed by Bouwman et al. [8], is network threat detection. CTI platforms like the Malware Information Sharing Platform (MISP) [15] were developed to share such threat intelligence information. MISP is a hub and spoke sharing platform that offers a variety of attributes to describe network activity, such as IP and MAC addresses, network ports, hostnames and domain names, email addresses, urls, hashes of payloads, etc., and it offers capabilities to generate rules for well-known intrusion detection systems (IDS), such as Snort [16], Suricata [17], and Bro (currently known as Zeek) [18].

However, as previous research by Iklody et al. [19] and Mokaddem et al. [20] has pointed out, the value of IoCs decays over time due to their volatile nature. For example, the IP address of a compromised machine may change, the infected machine may be cleaned up, or a domain name may be traded and used in a different benign or malicious way.

IoCs offer merely a hint of a cyber threat, and can become inaccurate over time. Traditional and deep learning based methods have been investigated to improve the accuracy of detection. Sarker et al. [21] reviewed used cases of different cyber security incidents, ranging from unauthorized access, malware, denial of service, to phishing and zero-day attacks. They highlighted how data science and machine learning is impacting the data processing and intelligent decision making in cyber security solutions, motivating this with examples of how traditional cyber security defense strategies, such as signature-based IDS, may fall short to certain attacks. The authors listed several well-known data sets frequently used for cyber security research, and how machine learning techniques like supervised and unsupervised learning, deep learning, and other algorithms can help identify security threats. The authors discussed the impact of data quality, context-awareness in cyber security, and prioritization of security alerts as some of the more important directions for further research.

Beyond the examples referred to in the previous survey, intrusion detection researchers evaluated the effectiveness of decision trees [22], support vector machines [23], k-nearest neighbors [24], ensemble methods [25], and deep learning methods [26]. For a more detailed discussion on the use of traditional machine learning (ML) and deep learning (DL) methods for intrusion detection, we also refer to the following surveys [27–29]. Another security application where ML and DL are evaluated to mitigate cyber threats is malware detection [30–33].

Compared to IoCs that typically embody a list of annotated key-value pairs, ML models and deep neural networks (DNN) are computationally more demanding, but they have more expressive and decisive power to characterize what constitutes a cyber threat. For example, compared to a flat list of attributes, a decision tree can express the conditions over these attributes to indicate the presence of an attack. However, ML models also come with their own cyber threats that may enlarge the attack surface of any security systems that adopts these AI techniques.

2.3. Attacks on Machine Learning Models and Deep Neural Networks

The recent advances and successes in the domain of machine learning (ML) and deep learning (DL) have been key drivers behind the adoption of AI techniques in cyber security solutions, as they provide the capabilities to increase the effectiveness of security intelligence when managing and responding to digital evidence of cyber threats [34]. Still, ML and DL models are not exempt from making mistakes, as they do not always detect real security threats or they sometimes misclassify a benign event as malicious. Arp et al. [35] discuss 10 subtle pitfalls that undermine the performance of

machine learning models, making them potentially unsuitable for security use cases. Some of these pitfalls include sampling bias, label inaccuracies, the base rate fallacy, and lab-only evaluations.

Barreno et al. [36] discussed how misclassifications can be further aggravated when adversaries have the ability to attack or fool an ML model. They argue that machine learning should perform well under adversarial conditions, and they present several threats in terms of the attacker's incentives and capabilities. For example, in a causative attack, adversaries aim to influence the learning process by controlling the training data. If the latter is not possible, an adversary can still launch an exploratory attack by influencing the outcome of the ML model by obfuscating the inputs. This way, false negatives may violate the integrity of systems and services, while false positives may disrupt their availability. As ML models need to be updated to recognize new threats, adversaries may aim to poison the ML model by injecting misleading training data. Rubinstein et al. [37] and Biggio et al. [38] demonstrated such poisoning attacks against anomaly detectors and support vector machines (SVM). Specifically for mobile malware detection systems, Chen et al. [39] discussed how to automate poisoning attacks and defenses. They show how conventional machine learning classifiers can fail against weak, strong, and sophisticated attackers that target the training phase to poison the detection. They proposed a solution called KuafuDet that relies on a self-adaptive learning scheme to filter suspicious false negatives and feed them back into the training phase. Preuveneers et al. [40] investigated the poisoning of autoencoders used for anomaly detection in an IDS scenario, where the autoencoders were trained in a federated learning setting. The authors proposed the use of blockchain technology to audit the learning process as a way to have contributing parties held accountable for their DL model updates.

The evasion of ML models can be facilitated whenever adversaries are able to gain unauthorized access to the models. Even if an adversary only has blackbox access to the model and no prior knowledge of the model's parameters or training data, extracting ML models may become practically feasible when they are exposed through publicly accessible query APIs used for prediction and classification. Tramèr et al. [41] discussed how they were able to duplicate the functionality of confidential ML models using the publicly accessible query interfaces of ML-as-a-Service (MLaaS) systems. They demonstrated the ability of an adversary extracting with near-perfect fidelity target ML models on BigML and Amazon Machine Learning online services. Juuti et al. [42] proposed new model extraction attacks using new schemes for generating synthetic queries and optimizing the hyperparameters of DNNs. They proposed PRADA as a countermeasure against model extraction attacks. PRADA analyzes the distribution of consecutive API queries and triggers an alert when the distribution deviates from benign behavior. Kesarwani et al. [43] proposed a monitoring solution that aims to quantify how much of a model has been extracted by observing the API query and response streams. Furthermore, they empirically evaluated information gain and feature space coverage strategies to estimate the learning rate of single and colluding adversaries.

Other attacks focus on the data used to train ML models. If an adversary is able to infer that a subject's sensitive information is used to train a model, then the privacy of the subject may be violated. These attacks are commonly referred to as membership inference attacks. Given a data record and blackbox access to a ML model trained on Google and Amazon MLaaS providers, Shokri et al. [44] investigated the feasibility to determine if that record was in the training data set. They created multiple shadow models that imitated the behavior of the target model and for which they know the training data sets. They then used these models to distinguish the target model's outputs on members versus non-members of its training data set. Nasr et al. [45] investigated a solution to address the membership inference attack by introducing a privacy mechanism for training ML models that aims to provably guarantee membership privacy. In their approach, the prediction of the ML model on training data cannot be distinguished from predictions made on other data points from the same distribution. Formalized as a min-max game optimization problem, the authors designed an adversarial training algorithm that minimizes the classification loss of the model as well as the maximum gain of the membership inference attack against it.

A related privacy threat are model inversion attacks that can help extract particular training data from the model. Frederikson et al. [46] developed a new class of model inversion attacks that leveraged the confidence measures along with the class label. They demonstrated how they were able amongst others to recover recognizable images of people's faces given only their name and access to the deep neural network. Their attack aimed to find the image that maximized the confidence value for an image classification that also matched the target.

3. Sharing ML Models as Cyber Threat Intelligence

In this section, we elaborate on how we augmented the attribute and tagging based sharing of IoCs with MISP by sharing machine learning (ML) models for collaborative threat detection, and which security measures are put in place to share the ML model only with authorized members across communities. We will also discuss how these authorized members can then contribute by enriching these models with false positive and false negative inputs or adversarial examples, or by incrementally fine-tuning these models using their own data if they are not willing to share any sensitive information.

3.1. Sharing IoCs in MISP

In MISP, an attack is represented as an event where the IoCs (e.g., the IP address and the email address of the adversary) are structured as the attributes of the event. MISP can then correlate threat events thanks to the similarity of these attributes. To assist the security analyst, MISP can represent these correlations as a graph of interconnected events.

Attributes are grouped into 16 categories (cfr. https://www.circl.lu/doc/misp/categories-and-types/, accessed on 24 February 2021) which are again subdivided into more than 180 types. An example of an event with two attributes in MISP JSON format is shown in Listing 1.

Listing 1. Example of a MISP (Malware Information Sharing Platform) threat event with two attributes in the JSON format.

```
{
  "Event": {
    "uuid": "1fe1fb10-07e0-448b-99e6-3856c6d7d06c",
    "date": "2020-12-21",
    "threat_level_id": "1",
    "info": "This is a dummy event",
    "published": true,
    "analysis": "0",
    "distribution": "0",
    "Attribute": [
      {
        "type": "ip-src",
        "category": "Network activity",
        "to_ids": true,
        "distribution": "5",
        "comment": "This is a 1st dummy attribute",
        "value": "1.2.3.4"
      },
      {
        "type": "email",
        "category": "Network activity",
        "to_ids": false,
        "distribution": "5",
        "comment": "This is a 2nd dummy attribute",
        "value": "attacker@mail.com"
      }
    ]
  }
}
```

The sharing range of IoCs is declared in the 'distribution' field where the numeric value indicates whether an event or an individual attribute is shared (0) with members of the same organization as the owner of the event, (1) within the community (organizations on the same MISP server and organizations hosting MISP servers synchronizing with the main server), (2) with connected communities (including all organizations on synchronizing MISP servers and organizations hosting MISP servers two hops away connected to the aforementioned servers), (3) with all communities (events can propagate freely between servers), and (4) with local and/or cross-instance organizations

defined in a sharing group. Additionally, attributes can (5) inherit the 'distribution' setting from the parent event. The 'to_ids' field represents whether the attribute is meant to be used as a pattern for detection in automated processes (e.g., in a NIDS solution like Snort or Suricata) or rather for correlation analysis.

The example in Listing 2 illustrates how the published MISP event and attributes marked as IDS Signature are exported to a Suricata ruleset. It triggers an alert whenever a network packet is sent from the 1.2.3.4 IP address to the internal network.

Listing 2. Exporting MISP threat event to Suricata ruleset.

```
# MISP export of IDS rules - optimized for
#
# These NIDS rules contain some variables that need to exist in your configuration.
# Make sure you have set:
#
# $HOME_NET     - Your internal network range
# $EXTERNAL_NET - The network considered as outside
# $SMTP_SERVERS - All your internal SMTP servers
# $HTTP_PORTS   - The ports used to contain HTTP traffic (not required with suricata export)
#
alert ip 1.2.3.4 any -> $HOME_NET any (msg: "MISP e1 [] Incoming From IP: 1.2.3.4"; classtype:trojan-activity;
  sid:4000011; rev:1; priority:1; reference:url,https://localhost:8443/events/view/1;)
```

3.2. Sharing ML Models as Threat Intelligence

Rather than sharing IoCs whose effectiveness may deteriorate over time, our goal is to share ML models and guarantee interoperability across the members of a community. We therefore defined a new MISP object template that enables not only to embed the ML model but also other associated artifacts, such as examples of false positives and false negatives including adversarial examples.

The MISP object template in Listing 3 is a reduced version of our meta-model to represent and share a machine learning model for IDS threat detection. Our MISP object template has attributes to describe the input feature vectors and the output labels, to attach sample code to load, pre-process raw data into input feature vectors, and to evaluate and update the ML model. The template also includes an optional attribute to represent the version of the ML model, and an attribute to hold a digital signature of the model so that the recipient can guarantee its authenticity. However, note that this MISP object template does not specify the kind of machine learning model (e.g., decision tree, SVM, neural network), but only the type of format in which it is serialized. For the sake of simplicity, our proof-of-concept is built on a Python-based framework that leverages Scikit-Learn (cfr. https://scikit-learn.org/, accessed on 24 February 2021) for traditional machine learning models, and Tensorflow 2 (cfr. https://www.tensorflow.org, accessed on 24 February 2021) and Keras (cfr. https://keras.io, accessed on 24 February 2021) for deep learning-based models. The encoding formats for the ML models mentioned in the MISP object template in Listing 3 correspond to these frameworks.

To collaboratively improve shared ML models, we also need a way to share examples where the ML model misclassifies an input sample. Listing 4 describes a MISP object template specifically for IDS threat detection to collect PCAP traces that are incorrectly labeled as a false positive or false negative, with the ability to include the correct label. Note that the ML labels are not fixed and the user can provide custom labels beyond the sane defaults (i.e., 'true' and 'false') provided, for example, to upload and share multiclass classification models that can distinguish different threats.

Once the MISP object templates are registered within MISP, they can be instantiated by adding objects to an event, as illustrated in Figure 1, either through the web interface of MISP, or by making use of MISP's REST APIs to submit events with attributes and objects.

Listing 3. MISP object template for IDS (intrusion detection systems) machine learning models.

```
{
  "attributes": {
    "creation_timestamp": {
      "description": "ML model creation timestamp",
      "misp-attribute": "datetime"
    },
    "ml_file": {
      "description": "ML model file",
      "disable_correlation": true,
      "misp-attribute": "attachment"
    },
    "ml_encoding": {
      "description": "Encoding format of the ML model",
      "misp-attribute": "text",
      "sane_default": [ "Pickle", "Joblib", "HDF5", "JSON", "Protobuf" ],
      "multiple": false
    },
    "ml_url": {
      "description": "URL of an externally hosted ML model",
      "misp-attribute": "url"
    },
    "ml_label": {
      "description": "Classification label of the ML model",
      "misp-attribute": "text",
      "sane_default": [ "true", "false" ],
      "multiple": true
    },
    "ml_version": {
      "description": "Version number of the ML model",
      "misp-attribute": "text"
    },
    "sha256": {
      "description": "SHA256 of the ML model",
      "misp-attribute": "sha256"
    }
  },
  "description": "ML model for IDS traffic classification",
  "meta-category": "network",
  "name": "ml_ids",
  "required": [ "creation_timestamp", "ml_encoding" ],
  "requiredOneOf": [ "ml_file", "ml_url" ],
  "uuid": "fbcc20a2-54e2-11eb-ae93-0242ac130002",
  "version": 1
}
```

Listing 4. MISP object template for IDS machine learning models.

```
{
  "attributes": {
    "pcap": {
      "description": "A misclassified PCAP trace",
      "disable_correlation": true,
      "misp-attribute": "attachment",
      "multiple": true,
      "ui-priority": 0
    },
    "ml_error_type": {
      "description": "Type of binary classification error",
      "misp-attribute": "text",
      "sane_default": [ "false_positive", "false_negative" ],
      "multiple": false,
      "ui-priority": 1
    },
    "ml_label": {
      "description": "Classification label of the ML model",
      "misp-attribute": "text",
      "sane_default": [ "true", "false" ],
      "multiple": true,
      "ui-priority": 1
    }
  },
  "description": "PCAP trace for ML IDS classification",
  "meta-category": "network",
  "name": "ml_ids_input",
  "required": [ "pcap", "ml_label", "ml_error_type" ],
  "uuid": "4c9eb65e-54eb-11eb-ae93-0242ac130002",
  "version": 1
}
```

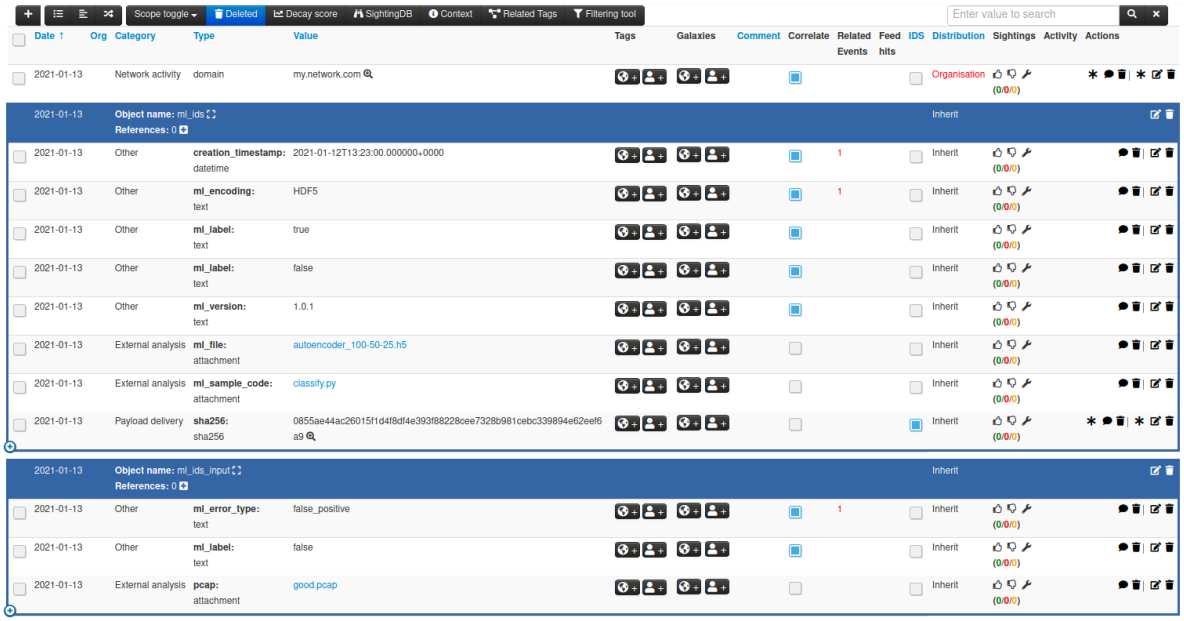


Figure 1. MISP objects for ML (machine learning) model based threat intelligence sharing.

The MISP event example illustrates a deep learning autoencoder [47] model serialized in the HDF5 encoding, which attempts to distinguish benign and malicious traffic, and sample Python code to process and classify the input data. The event is augmented with an object describing a benign PCAP example that is misclassified as a false positive. The autoencoder in Figure 1 is merely used for illustration purposes. This neural network model learns a compressed encoding of a normal data set without anomalies in an unsupervised manner. Once this neural network has learned how to represent benign behavior, it can reconstruct the original input from the compressed feature vector as depicted in Figure 2, and detect anomalies which typically have a higher reconstruction error than normal input samples.

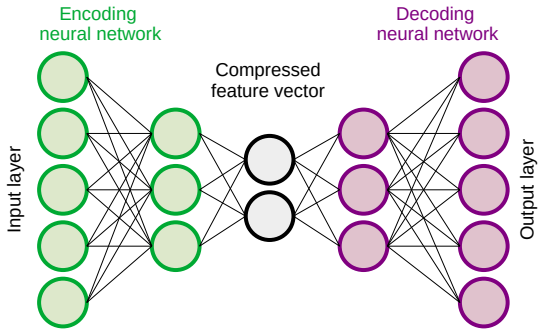


Figure 2. Feed-forward autoencoder for anomaly detection.

Note though that the purpose of this work is not to improve the accuracy of a particular traditional ML or DL model, but rather to analyze the advantages and drawbacks of sharing ML models. Indeed, sharing such models can be done for a variety of use cases that the security community is interested in (e.g., malware and APT analysis, intrusion detection, application log auditing, and user and entity behavior analytics). Even for the same security use case, one can use different kinds of traditional ML or DL models, for example, classification models trained in a supervised manner to recognize particular network attacks. In the following section, we will provide some examples of ML models and adversarial examples that could be shared within a security community in a similar manner.

3.3. ML and DL Models for DDoS Detection

Our goal is not to find the best ML or DL models, nor to suggest any methods to adversarially train these models. We consider this out of scope for this work as this is a fast moving research area for which we refer to relevant related work [48–50]. Our aim is to illustrate a scenario of ML models and associated artifacts being shared in a community of collaborating security analysts. We will illustrate our approach with a distributed denial of service (DDoS) detection use case. For this purpose, we replayed the PCAP files from the CSE-CIC-IDS2018 [51] (cfr. https://www.unb.ca/cic/datasets/ids-2018.html, accessed on 24 February 2021) cyber defense data set, more specifically those that were originally captured on Wednesday, February 21, 2018. On this day, two DDoS attacks were carried out using the high orbit ion cannon (HOIC) and low orbit ion cannon (LOIC) tools on the TCP and UDP network layers.

3.3.1. Pre-Processing the PCAP Files

We used Zeek 3.2.3 (cfr. https://zeek.org, accessed on 24 February 2021) to replay the 447 PCAP files and process the network traffic. Zeek then generates various structured log files (e.g., conn.log, http.log, dns.log, ssh.log, etc.). Subsequently, we processed the log files with Zeek Analysis Tools (ZAT) 0.4.1 (cfr. https://pypi.org/project/zat/, accessed on 24 February 2021). For the purpose of these experiments, we only used the conn.log file, and removed the source and destination IP addresses, i.e., the 'id.orig_h' and 'id.resp_h' fields, as well as the 'uid' field representing the unique identifier of the connection, and the 'history' field describing the state history of the connection. Other categorical fields were label-encoded so that each of the 13 remaining fields has a numerical representation. The outcome of this pre-processing is a data set with 6,841,238 feature vectors, of which 1,082,442 indicate a DDoS attack and 5,758,796 benign traffic. The labeling was done based on the time of day and the IP addresses involved in the attack, as documented on the data set's website. We created stratified splits of the data set to use 80% for training and 20% for testing, but also confirmed the accuracy of the obtained models through 10-fold cross-validation. We trained several traditional ML models to recognize DDoS attacks with Scikit-Learn 0.23.2, and DL using Tensorflow 2.4.1. Later on, we will elaborate on how to cryptographically protect these ML and DL models.

3.3.2. Sharing a Decision Tree Model and an Adversarial Example

The ML-based threat intelligence sharing story begins with a simple example. We trained a 'DecisionTreeClassifier' model, enforcing a maximum depth of 2. The outcome of the training phase is depicted in Figure 3. A decision is made on two fields: 'resp_bytes' indicating the number of payload bytes sent by the responder, and 'orig_pkts' counting the number of packets sent by the originator.

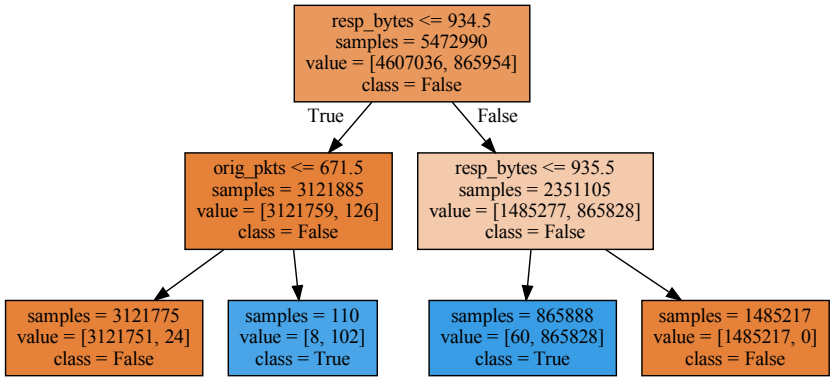


Figure 3. Decision tree with maximum depth 2.

The confusion matrix is shown in Figure 4, illustrating an almost perfect classification (i.e., an f1 score is 0.99996, and an average 10-fold cross-validation score of 0.99998). The decision tree model is rather small when serialized to disk in Joblib format, i.e., only 1713 bytes.

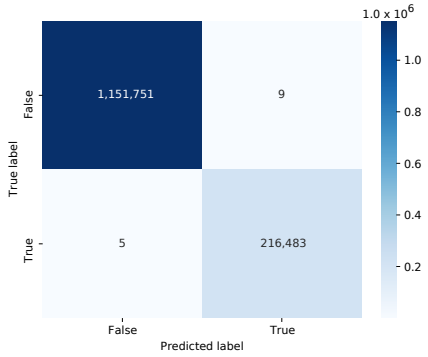


Figure 4. Confusion matrix for the decision tree model.

A security analyst could share this model through MISP in a similar manner as depicted in Figure 1, though this would be rather naive. Not only was the model trained and tested on a single data set with only a limited number of DDoS attack types, it is also fairly trivial to construct an adversarial example that can evade DDoS detection. The attacker only needs to trigger the target to reply with a payload having more than 935 bytes.

Table 1. Feature vector of a distributed denial of service (DDoS) connection correctly classified as malicious.

Field	Value	Field	Value	Field	Value	Field	Value
id.orig_p	54492	*duration*	0.102117	*missed_bytes*	0	*resp_ip_bytes*	1147
id.resp_p	80	*orig_bytes*	274	*orig_pkts*	5		
proto	0	*resp_bytes*	935	*orig_ip_bytes*	486		
service	0	*conn_state*	0	*resp_pkts*	5		

See Table 1 for a feature vector example of a DDoS connection correctly classified by the decision tree model as malicious. This is the result of a bi-directional flow of 10 packets encapsulating an HTTP request and response in the payloads of the packets, as shown in Listings 5 and 6.

Listing 5. HTTP request for the DDoS-HOIC attack.

```
GET / HTTP/1.0
Accept: */*
Accept-Language: en
Referer:  http://18.218.83.150
User-Agent:  Opera/9.80 (Windows NT 5.2; U; ru) Presto/2.5.22 Version/10.51
If-Modified-Since:  Mon, 29 Oct 2007 11:59:59 GMT
Cache-Control:  no-cache
Host: 18.218.83.150
```

Listing 6. HTTP response for the DDoS-HOIC attack.

```
HTTP/1.1 200 OK
Date: Wed, 21 Feb 2018 18:52:44 GMT
Server: Apache/2.4.18 (Ubuntu)
Vary: Accept-Encoding
Content-Length: 745
Connection: close
Content-Type: text/html;charset=UTF-8

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<html>
 <head>
  <title>Index of /</title>
 </head>
 <body>
<h1>Index of /</h1>
...
```

To bypass detection, the DDoS attack could trigger a bigger payload in the HTTP response, possibly by requesting a different URL. To confirm, we captured the PCAP packets after submitting the same HTTP request and returning the same and a slightly larger HTTP response. After replaying

these new PCAP files through Zeek to produce log entries in 'conn.log', the decision tree correctly classified the PCAP with the unmodified HTTP response as malicious, but indeed misclassified the adversarial example with the modified HTTP payload as benign.

In practice, after one security analyst shared the ML model through MISP, a second could contribute to this threat intelligence by sharing this particular PCAP file as an adversarial example using the MISP object template we proposed earlier in Listing 4.

3.3.3. Sharing a Random Forest Model for Improved Classification

Continuing the story of sharing ML-based threat intelligence, and given the rather trivial decision tree example, a security analyst decides to try a more sophisticated random forest model. Hence, we trained a 'RandomForestClassifier' model with 20 decision trees and an unbounded depth on the same data. The file size of this model is significantly larger, i.e., 171,202 bytes. The confusion matrix of this ML model is shown in Figure 5. With fewer false positives and false negatives, the random forest model performs even better than the decision tree, with an average 10-fold cross-validation score of 0.99999.

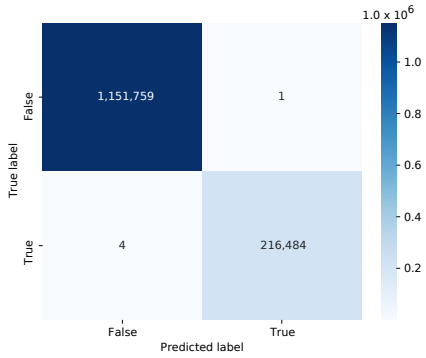


Figure 5. Confusion matrix for the random forest model.

The security analyst may contribute to the decision tree threat intelligence in MISP by also sharing their random forest model. After all, it has a better classification accuracy. However, the adversarial DDoS examples obtained through modifying the HTTP response payload are still misclassified as benign. Hence, slightly enlarging the payload with a few bytes will also fool the random forest model.

3.3.4. Detecting Attacks as Outliers with One Class Support Vector Machine Models

Another security analyst attempts to train a model that learns how benign traffic looks like in an unsupervised manner, so that any deviation could be considered as anomalous or malicious. Thus, we trained a 'OneClassSVM' model in an unsupervised manner on a subset of benign feature vectors only. This model uses an Radial Basis Function (RBF) kernel with the ν parameter set to the ratio of malicious input samples (i.e., 15.82%). The γ parameter was optimized through hyperparameter tuning via a simple grid search. Serialized to disk in Joblib format, the best ML model is 486,655 bytes large.

The confusion matrix is depicted in Figure 6. With 202,272 false positives, this model performs worse than the previous two models. The classifier has an accuracy of 0.85215, and an f1 score of 0.68154. This time, however, the adversarial DDoS examples with the modified HTTP responses are now correctly classified as malicious.

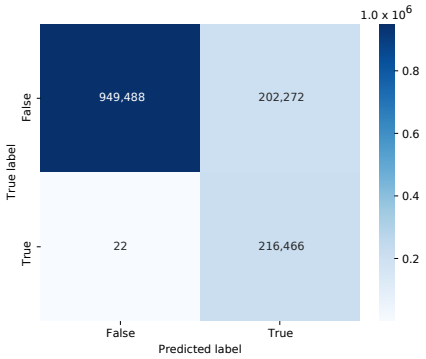


Figure 6. Confusion matrix for the OneClass SVM model.

3.3.5. Sharing a Multilayer Perceptron Model as Neural Network-Based Threat Intelligence

So far, only traditional ML models were shared within MISP, and some analysts may want to share their experience with deep learning models. Therefore, we built a simple multilayer perceptron (MLP) binary classifier. It has three fully connected feed-forward layers, uses the sigmoid activation function in the last layer, and minimizes the binary cross-entropy loss value with the Adam optimizer. The DL model is trained on the same Zeek log feature vectors as before. The serialized model in HDF5 format has a file size of 35,424 bytes. The confusion matrix is shown in Figure 7.

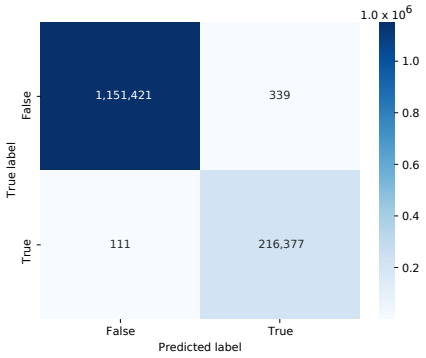


Figure 7. Confusion matrix for the multilayer perceptron (MLP) binary classifier.

The classifier has an accuracy of 0.99967, and an f1 score of 0.99896. Furthermore, the adversarial DDoS PCAP files with the modified HTTP responses were also correctly classified as malicious. So, contributing this model to the threat intelligence community seems like a good idea.

However, we were able to directly and slightly manipulate the 13 fields in the feature vector—i.e., the entry in the Zeek 'conn.log' log file—and have it misclassified by the MLP, though we did not create the corresponding PCAP file that would be pre-processed through Zeek into that particular feature vector. In that case, the security analyst does not upload a PCAP file, but could share some Python code that evaluates this feature vector to show how the MLP misclassifies it.

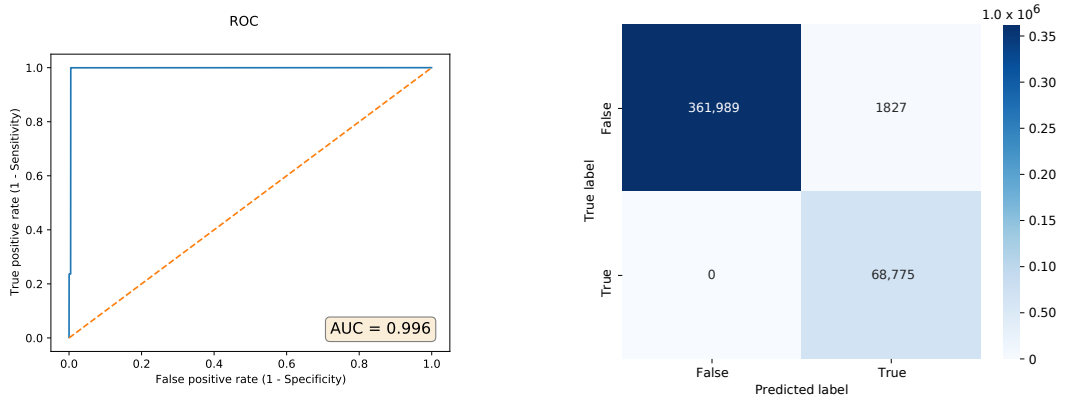
3.3.6. Detecting Attacks as Outliers with a Deep Learning-Based Autoencoder

Similar to the OneClass SVM outlier detection model, we trained a set of autoencoders, as depicted in Figure 2, in an unsupervised manner on benign traffic only. Due to the limited number of fields in the Zeek log files (i.e., input layer of size 13), the autoencoders were too small to use the reconstruction error to distinguish benign from malicious input samples. The autoencoders did not perform any better than random guessing. As an alternative, we used the pre-processed log files produced by the CICFlowMeter tool, also provided in the same CSE-CIC-IDS2018 data set [51]. This tool generates higher-level feature vectors with 77 fields that characterize complete connection flows, rather than the 13 fields in the Zeek log files. Compared to the OneClass SVM model, the autoencoder requires more data to train the model. We therefore added benign feature vectors from the other days in the training

data set, but left the test set the same. This is similar to a security analyst training a shared model on its own private data.

We tuned the hyperparameters of these neural networks by training different autoencoders with the same number of layers, but with a different number of neurons per feed-forward layer. The input layer of the encoding neural network (see Figure 2) has 77 neurons, the second layer varies between 70 and 50 neurons, the third layer between 40 and 20 neurons, and the size of the middle layer representing the compressed feature vector ranges from 30 to 10 neurons. The decoding neural network has a mirrored configuration. As such, these autoencoders are significantly larger than the ones using the Zeek log files as input.

The best neural network architecture with the highest area under curve (AUC) found through hyperparameter tuning, is the one with a configuration of 60-40-20-40-60 neurons. Its ROC curve and confusion matrix are shown in Figure 8. Serialized in the HDF5 file format, the autoencoder file is 178,568 bytes large. We also tested the adversarial DDoS examples, and the PCAP files with the modified HTTP payloads were correctly classified as malicious.



(**a**) (**b**)

Figure 8. Autoencoder with the highest AUC (area under curve) value. (**a**) ROC (receiver operating characteristics) curve; (**b**) Confusion matrix.

3.3.7. Discussion

Even if ML models are better at threat detection compared to just sharing the IP address of the attacker as an IoC, it is clear from the previous experiments that traditional machine learning and deep learning classifiers that aim to learn how both benign and malicious traffic looks like, can be fooled to misclassify malicious traffic, i.e., evade detection, by modifying the size of the payload. As such, we need accurate ML and DL models, as well as models that are also robust against adversarial examples.

Obviously, there are more sophisticated types of neural networks [52,53] that can classify DDoS network traffic with a higher accuracy. The above examples were only meant as an illustration of how security analysts could contribute different ML models and/or adversarial examples that are misclassified. Optimizing the accuracy of the ML models and neural networks is beyond the scope of this work and also not relevant given the fact that the models are trained on a synthetic data set (i.e., not captured in the wild and therefore less effective in the real world).

It is also clear that certain assets, such as adversarial examples, are too sensitive to be shared publicly, as they could help attackers learn how to evade detection. When shared through MISP as attributes, access should be restricted to authorized and trusted individuals. Also, in the above examples, we only discussed evasion attacks that fool ML and DL models. However, there are many more ML attacks. To document these threats in MISP in a structured manner, we propose a new MISP taxonomy in the next section.

3.4. Taxonomy for Threats and Attacks against ML Models

When sharing ML models used for security purposes, those models themselves may introduce vulnerabilities and hereby increase the attack surface of the security monitoring system or application. For example, the adversary may have gained unauthorized access to the ML model or abuse a public ML inference API end-point to reconstruct a shadow model to learn how to evade detection. The model may not be robust against adversarial examples, or maliciously crafted input samples may be inserted in the data to poison the training of the ML model.

To address the threat exposure of ML models, members of the community in which the ML model is shared, can contribute by identifying these threats and collaboratively improving the shared ML model. They can not only provide input samples that are misclassified to increase the accuracy of the model, but also tag ML models and related artifacts if they discover a threat against them. For example, they could contribute an adversarial example against a shared ML model with the goal to make the model more robust.

To share this kind of threat intelligence, we defined a taxonomy of adversarial machine learning threats that can be used to tag the attributes of our instantiated MISP object templates. The approach is similar to the Traffic Light Protocol (TLP) tagging scheme (cfr. https://www.us-cert.gov/tlp, accessed on 24 February 2021) for sharing threat intelligence with appropriate audiences. Our MISP taxonomy supports security analysts to tag both the ML models as well as input samples to these models to indicate the threat and the phase of the ML process that is being attacked, i.e., the training process or the inference process. Listing 7 shows a subset of our MISP taxonomy to annotate instantiations of our MISP object templates using the tags defined in the taxonomy. The full version lists several more ML threats, including model inversion and membership inference attacks, neural trojans, etc.

Consider again the example in Figure 1 that indicates that the provided 'good.pcap' input example is flagged as a false positive. The autoencoder acts as an anomaly detector by learning a representation of benign traffic, and flagging input samples as anomalies whenever their representation deviates too much from the benign representation. The 'good.pcap' could therefore be an opportunity to improve the accuracy of the autoencoder. However, imagine now that an attacker was able to craft a malicious PCAP file and insert it into the MISP system claiming it to be a false positive. The outcome of this attack is that the malicious sample would be used to further train the autoencoder. This way, the attacker would have poisoned the data set that the ML model used to learn to identify and classify benign and malicious inputs. Figure 9 illustrates the tagging of the autoencoder ML model and malicious PCAP input sample in MISP.

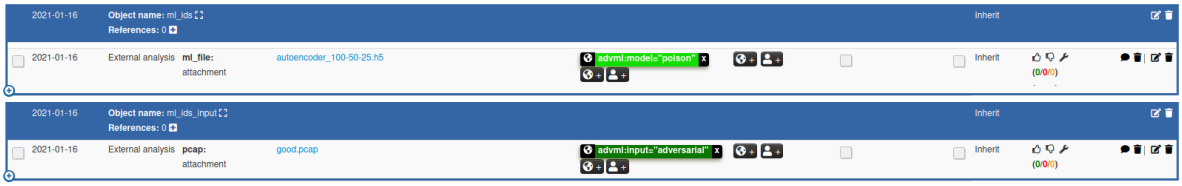


Figure 9. Tagging MISP objects with adversarial ML taxonomy.

Listing 7. MISP taxonomy for threats against machine learning models.

```
{
  "description": "The adversarial machine learning taxonomy was designed with the objective to create a
                  classification scheme for sharing machine learning threats.",
  "expanded": "Adversarial Machine Learning",
  "namespace": "advml"
  "predicates": [
    {
      "value": "input",
      "expanded": "Model input threat"
    },
    {
      "value": "model",
      "expanded": "Model threat"
    }
  ],
  "values": [
    {
      "predicate": "input",
      "entry": [
        {
          "value": "unauthorized_access",
          "expanded": "Unauthorized access",
          "description": "Access to input samples was granted to unauthorized individual"
        },
        {
          "value": "adversarial",
          "expanded": "Adversarial sample intentionally designed to cause the ML model to make a mistake"
        }
      ]
    },
    {
      "predicate": "model",
      "entry": [
        {
          "value": "unauthorized_access",
          "expanded": "Unauthorized access",
          "description": "Access to ML model was granted to unauthorized individual"
        },
        {
          "value": "shadow",
          "expanded": "Shadow model",
          "description": "Imitate existing ML model to successfully transfer adversarial examples"
        },
        {
          "value": "poison",
          "expanded": "Model poisoning",
          "description": "Poison the data set the ML model uses to identify and classify threats"
        },
        {
          "value": "evasion",
          "expanded": "Model evasion",
          "description": "Craft an adversarial example to evade online or offline detection"
        }
      ]
    }
  ],
  "version": 3
}
```

3.5. Protecting Access to ML Models with CP-ABE

Our solution shares ML models and data as MISP events and attributes. These events and attributes are shared within and across communities according to the distribution mechanisms that MISP provides out of the box to limit the sharing of threat intelligence to a limited audience.

As these ML models are subject to attacks, we implemented additional access control mechanisms to ensure that unauthorized individuals cannot get access to the model. We built on top of our previous work TATIS [54] to encrypt the data in the MISP database so that only authorized individuals can decrypt the ciphertext, even after sharing across communities. TATIS is a reverse proxy for MISP's REST APIs, and encrypts data with the Ciphertext-Policy Attribute-Based Encryption (CP-ABE) [55] scheme.

First the attributes in the MISP event (i.e., in our case the ML models and artifacts) are encrypted with a random AES key resulting in a threat event ciphertext CT_{AES}. The AES key itself is then encrypted with CP-ABE using an attribute policy provided by the MISP event or attribute producer,

which results in a ciphertext CT_{ABE} that protects the AES key. These attribute policies are a boolean composition of attribute conditions in disjunctive normal form:

$$\text{role:analyst} \vee (\text{subject:charlie} \wedge \text{organization:company123}) \quad (1)$$

A MISP event consumer first obtains a private CP-ABE decryption key that is derived from user attributes. This private key is then used to decrypt the ciphertext CT_{ABE} to obtain the AES key with which each ML model and associated artifact is individually encrypted. The ciphertext CT_{ABE} can only be decrypted if the user attributes matches the policy specifying the conditions under which the ciphertext can be decrypted. In the above examples, only an analyst or Charlie at Company123 can decrypt CT_{ABE} to retrieve the AES key. The user attributes themselves are maintained in an identity and access management system against which the user needs to authenticate to make use of TATIS. The overall approach of our solution is depicted in Figure 10. For more details on the CP-ABE scheme itself, we refer to our previous work [56].

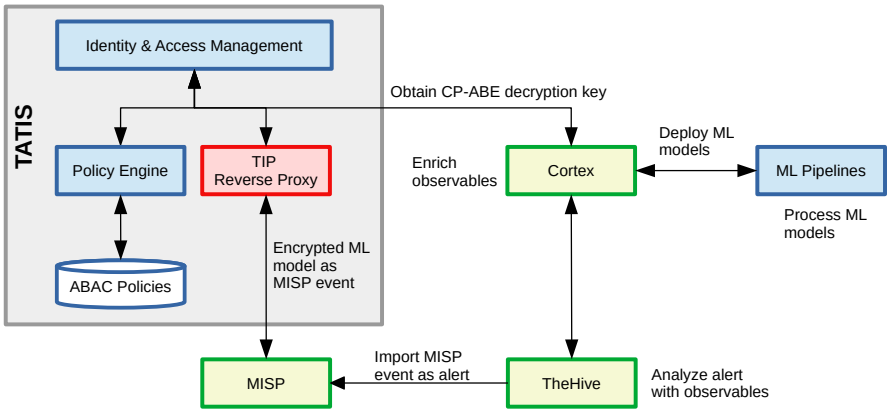


Figure 10. Deploying and evaluating encrypted ML models through TheHive and a Cortex analyzer.

Our encrypted ML models and data are ingested by TheHive (cfr. https://thehive-project.org, accessed on 24 February 2021), a scalable and open Security Incident Response Platform (SIRP) that can receive alerts from different sources via its REST API. It can be tightly integrated with MISP to start the investigation of cases from MISP events as depicted above. The goal of TheHive to simplify the collaboration among security professionals—such as SOC and CERT security analysts—and the process of investigating and acting upon threat intelligence information in a timely manner. TheHive platform is typically complemented with the Cortex engine (cfr. https://github.com/TheHive-Project/Cortex, accessed on 24 February 2021) to analyze observables and events. Analysts can also automate the response by initiating one or more responders.

Figure 11 depicts a subset of the attributes of the MISP event (i.e., only the file attachments) after being imported into TheHive. We extended Cortex with new capabilities by writing a new analyzer and responder in Python that interacts with TATIS to decrypt the ML models and data, to analyze the tags, and process the ML models. For example, our analyzer can not only check the accuracy of a model against a local data set, but also its robustness against adversarial examples.

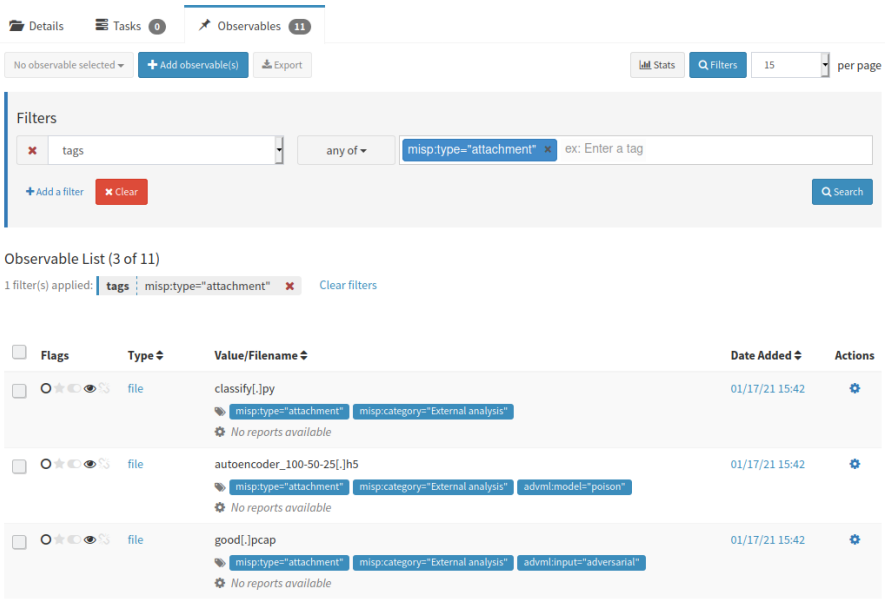


Figure 11. MISP event with ML artifacts imported into TheHive as an alert with observables.

4. Evaluation

In this section, we briefly highlight the potential risks and threats of our design to share encrypted ML models and artifacts as cyber threat intelligence. We already demonstrated the practical feasibility for our DDoS network security use case with several ML and DL models, in the previous section, and will now assess the threat model and added value of applying CP-ABE cryptography to protect these assets.

4.1. Threat Model and Analysis

In practice, encrypted ML models and artifacts are synchronized across different MISP and TheHive instances. Even if a software vulnerability is found in either software platforms, the data is not stored in plaintext in the underlying database (i.e., MariaDB, Cassandra, and ElasticSearch), but encrypted at rest. Only with a matching CP-ABE decryption key can one obtain the AES key with which the ML model and artifacts are encrypted. Hence, a malicious adversary could execute a credential stuffing attack to obtain the login credentials of a targeted subject in the IAM, and subsequently obtain and share the subject's CP-ABE decryption key. Under these circumstances and with the same capabilities, the adversary could instead leak the AES keys or the decrypted data itself. By leaking AES keys, it is less obvious whose login credentials or CP-ABE decryption key have been compromised.

The ML models are decrypted in our TATIS-enabled Cortex analyzer and responder, after which it is deployed in an ML pipeline either for further threat analysis or for a roll-out in a production environment (e.g., a ML enabled IDS solution). In the former case, the analysis itself may lead to new insights into vulnerabilities of the ML model which are reported in the form of tag annotations on existing MISP objects or new MISP objects describing adversarial examples. These insights need to be CP-ABE encrypted as well with the same or a more strict attribute policy. In the latter case, the production environment may be subject to the adversarial ML threats as discussed before, such as model stealing attacks and the construction of shadow models.

4.2. Performance Impact Analysis

For our experiments, we deployed MISP 2.4.136, TheHive 4.0.3, Cortex 3.1.0, as well as our TATIS framework in a VirtualBox environment with 8GB of memory and 2 virtual processors running Ubuntu 20.04. The experimental setup also consists of an AI system for traditional machine learning

(i.e., Scikit-Learn 0.23.2) and deep learning (i.e., Tensorflow 2.4.1) that is equipped with an NVIDIA Titan V CUDA card for accelerated deep learning.

4.2.1. Impact of Storing Encrypted ML Model Files in MISP and TheHive

As databases of CTI platforms can grow fairly big when collecting intelligence about a large amount of cyber threats, we measured the impact on the size of ML models when encrypting and storing them in MISP and TheHive. Each of the individual autoencoders has been serialized into the HDF5 file format, and for this particular type of ML model, the size of each file greatly depends on the number of weights in the neural network. At first, we did not compress the original HDF5 files. The results of the model storage impact are depicted in Figure 12 for 3 autoencoders (including the best one with the highest AUC value).

The impact for MISP and TheHive is the same as the latter pulls each event from MISP as an alert with observables into TheHive. We observe on average a size increase of about 35%. This can be explained by the fact that the models are first encrypted with the AES scheme. This encryption scheme does not significantly increase the size of the ciphertext relatively to the plaintext. However, the ML models (as well as the PCAP files) are stored in MISP as file attachments, which means that they are Base64 encoded. As a result, every three 8-bit bytes are represented as four 6-bit bytes, which explains a size increase of about 33%. The AES key itself is also encrypted with the CP-ABE scheme, which creates an additional overhead of a few kilobytes. Note that if the HDF5 file is stored in plaintext, the overall impact would also be 33% due to Base64 encoding. This means that the impact on the storage size due to the AES and CP-ABE encryption is limited to about 2–3%. Furthermore, we were able to compress the original autoencoder HDF5 files with 28% up to 37% using the gzip utility. This would reduce the overall amount of storage required, but the relative size impact due to the Base64 encoding would remain.

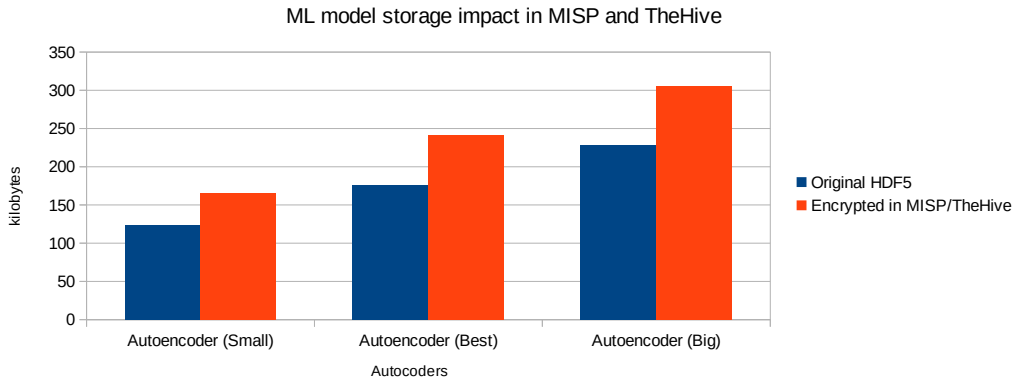


Figure 12. Impact of storing HDF5 autoencoder files in MISP and TheHive.

To avoid storing the model twice in different CTI platforms, the alternative is not to store the models themselves but rather a url to the model stored online and a SHA256 cryptographic hash of the (encrypted) file to ensure its integrity. However, the web server hosting the ML model may then leak the IP address and possibly the organization of the security analyst downloading the ML model.

4.2.2. Performance Impact of Decrypting and Deploying ML Models

MISP events are pulled into TheHive at a regular interval (e.g., every hour) after which they appear as an alert into the dashboard of the security analyst. At that point, they can import the alert as a new case with observables (cfr. Figure 11), and invoke our Cortex analyzers and responders. At this point, the ML model and associated artifacts are decrypted, deployed in the ML pipeline, and processed as illustrated in Figure 10.

We measured the relative computational overhead of decrypting and deploying the autoencoder models, and these performance numbers are depicted in Figure 13. The CP-ABE and AES decryption

takes overall less than 50 ms, which is relatively low compared to the time required to deploy the ML model and artifacts over the network on the GPU accelerated deep learning system. The main reason why the processing of ML models takes this long (i.e., up to multiple seconds) is that this step in some cases also includes the pre-processing of relatively small PCAP files through Zeek or CICFlowMeter to produce the corresponding log files which are then used as input samples for the ML and DL models to classify. To illustrate, CICFlowMeter pre-processed each adversarial PCAP file with the modified HTTP response in about 695 ms, and Zeek did the same job in about 296 ms.

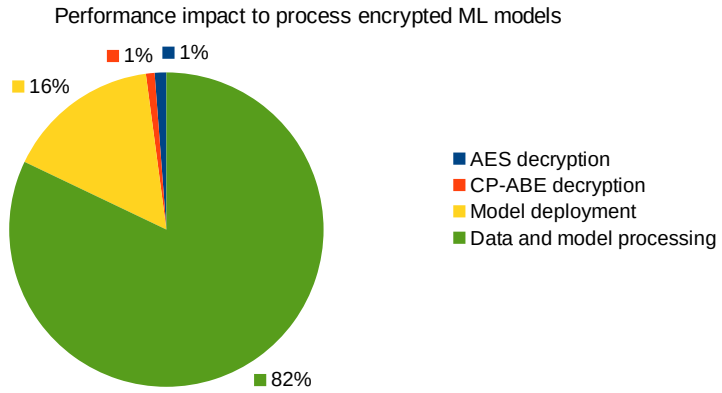


Figure 13. Performance impact of decrypting, deploying, and processing ML models.

To summarize, the computational overhead to cryptographically protect the ML models and associated artifact is relatively small compared to the actual time to pre-process the data, deploy, and evaluate the ML models, for example, to find false positives, false negatives, or identify adversarial ML threats as outlined in our MISP taxonomy of Figure 9.

5. Conclusions

In this work, we proposed a solution to complement the sharing of attribute-based IoC with machine learning based threat detection, as the former have been shown to lose value over time. When using AI to strengthen cyber security applications, and threat detection frameworks in particular, we must account for the risks and vulnerabilities that machine learning models may impose. We demonstrated this for a DDoS network security use case, comparing different ML and DL models, and their robustness against adversarial examples that aim to evade detection.

To address these challenges, our solution proposed a new object template to simplify the sharing of ML models and associated artifacts across different instances of the well-established MISP threat intelligence sharing platform. We additionally defined a new MISP taxonomy to tag this threat intelligence with adversarial ML threats. Last but not least, to prevent unauthorized access to potentially sensitive ML models and input samples, we leveraged our previous work TATIS to encrypt this threat intelligence with CP-ABE so that only authorized members of a community can decrypt this information. To assist security analysts, we implemented these encryption, decryption, and processing capabilities as extensions for the TheHive and Cortex security incident response frameworks. We carried out a performance analysis, indicating that the proposed solution is practically feasible. As future work, we will investigate to what extent threats against shared ML models can be automatically annotated, also for cyber security use cases beyond IDS, and how explainable AI frameworks could be plugged into our solution to help enrich shared ML models.

Author Contributions: Conceptualization, methodology, software, and validation, D.P.; writing—original draft preparation, D.P.; writing—review and editing, D.P.; supervision, D.P. and W.J.; project administration, D.P.; funding acquisition, W.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research is partially funded by the Research Fund KU Leuven, by the Flemish Research Programme Cybersecurity, and by the European H2020-SU-ICT-03-2018 Project No. 830929 CyberSec4Europe (https://cybersec4europe.eu accessed on 24 February 2021).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The CSE-CIC-IDS2018 dataset analysed during this study is available at https://www.unb.ca/cic/datasets/ids-2018.html, accessed on 24 February 2021.

Acknowledgments: We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan V GPU used for this research.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Gschwandtner, M.; Demetz, L.; Gander, M.; Maier, R. Integrating Threat Intelligence to Enhance an Organization's Information Security Management. In Proceedings of the 13th International Conference on Availability, Reliability and Security, ARES 2018, Hamburg, Germany, 27–30 August 2018. [CrossRef]
2. Johnson, C.; Badger, M.; Waltermire, D.; Snyder, J.; Skorupka, C. *Guide to Cyber Threat Information Sharing*; Technical Report; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2016.
3. Tounsi, W.; Rais, H. A survey on technical threat intelligence in the age of sophisticated cyber attacks. *Comput. Secur.* **2018**, *72*, 212–233. [CrossRef]
4. Wagner, T.D.; Palomar, E.; Mahbub, K.; Abdallah, A.E. A Novel Trust Taxonomy for Shared Cyber Threat Intelligence. *Secur. Commun. Netw.* **2018**, *2018*. [CrossRef]
5. Hassan, W.U.; Guo, S.; Li, D.; Chen, Z.; Jee, K.; Li, Z.; Bates, A. NoDoze: Combatting Threat Alert Fatigue with Automated Provenance Triage. In Proceedings of the 26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, CA, USA, 24–27 February 2019.
6. Aminanto, M.E.; Zhu, L.; Ban, T.; Isawa, R.; Takahashi, T.; Inoue, D. Automated Threat-Alert Screening for Battling Alert Fatigue with Temporal Isolation Forest. In Proceedings of the 2019 17th International Conference on Privacy, Security and Trust (PST), Fredericton, NB, Canada, 26–28 August 2019; pp. 1–3. [CrossRef]
7. Li, V.G.; Dunn, M.; Pearce, P.; McCoy, D.; Voelker, G.M.; Savage, S.; Levchenko, K. Reading the Tea Leaves: A Comparative Analysis of Threat Intelligence. In Proceedings of the 28th USENIX Conference on Security Symposium (SEC'19), Santa Clara, CA, USA, 14–16 August 2019; pp. 851–867.
8. Bouwman, X.; Griffioen, H.; Egbers, J.; Doerr, C.; Klievink, B.; van Eeten, M. A different cup of TI? The added value of commercial threat intelligence. In Proceedings of the 29th USENIX Security Symposium, San Diego, CA, USA, 12–14 August 2020; Capkun, S., Roesner, F., Eds.; USENIX Association: Berkeley, CA, USA, 2020; pp. 433–450.
9. Adadi, A.; Berrada, M. Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI). *IEEE Access* **2018**, *6*, 52138–52160. [CrossRef]
10. Athalye, A.; Carlini, N.; Wagner, D. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv* **2018**, arXiv:1802.00420.
11. Jordan, B.; Piazza, R.; Darley, T. STIX Version 2.1. OASIS Committee Specification Draft 01/Public Review Draft 01. 2019. Available online: https://docs.oasis-open.org/cti/stix/v2.1/stix-v2.1.html (accessed on 24 February 2021).
12. Jordan, B.; Varner, D. TAXII Version 2.1. OASIS Committee Specification Draft 04/Public Review Draft 03. 2019. Available online: https://docs.oasis-open.org/cti/taxii/v2.1/taxii-v2.1.html (accessed on 24 February 2021).
13. Darley, T.; Kirillov, I.; Piazza, R.; .; Beck, D. CybOX Version 2.1.1. Part 01: Overview. OASIS Committee Specification Draft 01/Public Review Draft 01. 2016. Available online: http://docs.oasis-open.org/cti/cybox/v2.1.1/cybox-v2.1.1-part01-overview.html (accessed on 24 February 2021).

14. Ramsdale, A.; Shiaeles, S.; Kolokotronis, N. A Comparative Analysis of Cyber-Threat Intelligence Sources, Formats and Languages. *Electronics* **2020**, *9*, 824. [CrossRef]
15. Wagner, C.; Dulaunoy, A.; Wagener, G.; Iklody, A. 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 (WISCS '16), Vienna, Austria, 24 October 2016; pp. 49–56. [CrossRef]
16. Roesch, M. Snort—Lightweight Intrusion Detection for Networks. In Proceedings of the 13th USENIX Conference on System Administration (LISA '99), Seattle, WA, USA, 7–12 November 1999; pp. 229–238.
17. Park, W.; Ahn, S. Performance Comparison and Detection Analysis in Snort and Suricata Environment. *Wirel. Pers. Commun.* **2017**, *94*, 241–252. [CrossRef]
18. Paxson, V.; Campbell, S.; Lee, J. *Bro Intrusion Detection System*; Technical Report; Lawrence Berkeley National Laboratory: Berkeley, CA, USA, 2006.
19. Iklody, A.; Wagener, G.; Dulaunoy, A.; Mokaddem, S.; Wagner, C. Decaying Indicators of Compromise. *arXiv* **2018**, arXiv:1803.11052.
20. Mokaddem, S.; Wagener, G.; Dulaunoy, A.; Iklody, A. Taxonomy driven indicator scoring in MISP threat intelligence platforms. *arXiv* **2019**, arXiv:1902.03914.
21. Sarker, I.H.; Kayes, A.; Badsha, S.; Alqahtani, H.; Watters, P.; Ng, A. Cybersecurity data science: An overview from machine learning perspective. *J. Big Data* **2020**, *7*, 41. [CrossRef]
22. Kumar, M.; Hanumanthappa, M.; Kumar, T.V.S. Intrusion Detection System using decision tree algorithm. In Proceedings of the 2012 IEEE 14th International Conference on Communication Technology, Chengdu, China, 9–11 November 2012; pp. 629–634. [CrossRef]
23. Li, Y.; Xia, J.; Zhang, S.; Yan, J.; Ai, X.; Dai, K. An efficient intrusion detection system based on support vector machines and gradually feature removal method. *Expert Syst. Appl.* **2012**, *39*, 424–430. doi:10.1016/j.eswa.2011.07.032. [CrossRef]
24. Lin, W.C.; Ke, S.W.; Tsai, C.F. CANN: An intrusion detection system based on combining cluster centers and nearest neighbors. *Knowl.-Based Syst.* **2015**, *78*, 13–21. doi:10.1016/j.knosys.2015.01.009. [CrossRef]
25. Aburomman, A.A.; Ibne Reaz, M.B. A novel SVM-kNN-PSO ensemble method for intrusion detection system. *Appl. Soft Comput.* **2016**, *38*, 360–372. doi:10.1016/j.asoc.2015.10.011. [CrossRef]
26. Yin, C.; Zhu, Y.; Fei, J.; He, X. A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks. *IEEE Access* **2017**, *5*, 21954–21961. [CrossRef]
27. Buczak, A.L.; Guven, E. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Commun. Surv. Tutor.* **2015**, *18*, 1153–1176. [CrossRef]
28. Liu, H.; Lang, B. Machine learning and deep learning methods for intrusion detection systems: A survey. *Appl. Sci.* **2019**, *9*, 4396. [CrossRef]
29. Gamage, S.; Samarabandu, J. Deep learning methods in network intrusion detection: A survey and an objective comparison. *J. Netw. Comput. Appl.* **2020**, *169*, 102767. doi:10.1016/j.jnca.2020.102767. [CrossRef]
30. Le, Q.; Boydell, O.; Mac Namee, B.; Scanlon, M. Deep learning at the shallow end: Malware classification for non-domain experts. *Digit. Investig.* **2018**, *26*, S118–S126. doi:10.1016/j.diin.2018.04.024. [CrossRef]
31. Vinayakumar, R.; Alazab, M.; Soman, K.; Poornachandran, P.; Venkatraman, S. Robust intelligent malware detection using deep learning. *IEEE Access* **2019**, *7*, 46717–46738. [CrossRef]
32. Gibert, D.; Mateu, C.; Planes, J. The rise of machine learning for detection and classification of malware: Research developments, trends and challenges. *J. Netw. Comput. Appl.* **2020**, *153*, 102526. doi:10.1016/j.jnca.2019.102526. [CrossRef]
33. Alzaylaee, M.K.; Yerima, S.Y.; Sezer, S. DL-Droid: Deep learning based android malware detection using real devices. *Comput. Secur.* **2020**, *89*, 101663. [CrossRef]
34. Preuveneers, D.; Manco, G.; Guarascio, M.; Zarzosa, S.G.; Martins, R.; Atzeni, A.; Bernabe, J.B.; Soares, J.; Meng, W.; Corin, R.D.; Nieto, A.; Hita, A.; Pashchenko, I.; Tizio, G.D.; Canavese, D. D3.3: Research Challenges and Requirements to Manage Digital Evidence. H2020 CyberSec4Europe Deliverables. 2020. Available online: https://cybersec4europe.eu/publications/deliverables/ (accessed on 24 February 2021).
35. Arp, D.; Quiring, E.; Pendlebury, F.; Warnecke, A.; Pierazzi, F.; Wressnegger, C.; Cavallaro, L.; Rieck, K. Dos and Don'ts of Machine Learning in Computer Security. *arXiv* **2020**, arXiv:2010.09470.
36. Barreno, M.; Nelson, B.; Joseph, A.D.; Tygar, J.D. The security of machine learning. *Mach. Learn.* **2010**, *81*, 121–148. [CrossRef]

37. Rubinstein, B.I.; Nelson, B.; Huang, L.; Joseph, A.D.; Lau, S.h.; Rao, S.; Taft, N.; Tygar, J.D. ANTIDOTE: Understanding and Defending against Poisoning of Anomaly Detectors. In Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement (IMC '09), Chicago, IL, USA, 4–6 November 2009; pp. 1–14. [CrossRef]
38. Biggio, B.; Nelson, B.; Laskov, P. Poisoning Attacks against Support Vector Machines. In Proceedings of the 29th International Coference on International Conference on Machine Learning (ICML'12), Edinburgh, UK, 26 June–1 July 2012; pp. 1467–1474.
39. Chen, S.; Xue, M.; Fan, L.; Hao, S.; Xu, L.; Zhu, H.; Li, B. Automated poisoning attacks and defenses in malware detection systems: An adversarial machine learning approach. *Comput. Secur.* **2018**, *73*, 326–344. doi:10.1016/j.cose.2017.11.007. [CrossRef]
40. Preuveneers, D.; Rimmer, V.; Tsingenopoulos, I.; Spooren, J.; Joosen, W.; Ilie-Zudor, E. Chained Anomaly Detection Models for Federated Learning: An Intrusion Detection Case Study. *Appl. Sci.* **2018**, *8*, 2663. [CrossRef]
41. Tramèr, F.; Zhang, F.; Juels, A.; Reiter, M.K.; Ristenpart, T. Stealing Machine Learning Models via Prediction APIs. In Proceedings of the 25th USENIX Conference on Security Symposium (SEC'16), Austin, TX, USA, 10–12 August 2016; pp. 601–618.
42. Juuti, M.; Szyller, S.; Marchal, S.; Asokan, N. PRADA: Protecting Against DNN Model Stealing Attacks. In Proceedings of the 2019 IEEE European Symposium on Security and Privacy (EuroS P), Stockholm, Sweden, 17–19 June 2019; pp. 512–527. [CrossRef]
43. Kesarwani, M.; Mukhoty, B.; Arya, V.; Mehta, S. Model Extraction Warning in MLaaS Paradigm. In Proceedings of the 34th Annual Computer Security Applications Conference (ACSAC '18), San Juan, PR, USA, 3–7 December 2018; pp. 371–380. [CrossRef]
44. Shokri, R.; Stronati, M.; Song, C.; Shmatikov, V. Membership Inference Attacks Against Machine Learning Models. In Proceedings of the 2017 IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, 22–26 May 2017; pp. 3–18. [CrossRef]
45. Nasr, M.; Shokri, R.; Houmansadr, A. Machine Learning with Membership Privacy Using Adversarial Regularization. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS '18), Toronto, ON, Canada, 15–19 October 2018; pp. 634–646. [CrossRef]
46. Fredrikson, M.; Jha, S.; Ristenpart, T. Model Inversion Attacks That Exploit Confidence Information and Basic Countermeasures. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS '15), Denver, CO, USA, 12–16 October 2015; pp. 1322–1333. [CrossRef]
47. Zhou, C.; Paffenroth, R.C. Anomaly detection with robust deep autoencoders. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, 13–17 August 2017; pp. 665–674.
48. Kurakin, A.; Goodfellow, I.; Bengio, S. Adversarial machine learning at scale. *arXiv* **2016**, arXiv:1611.01236.
49. Tramèr, F.; Kurakin, A.; Papernot, N.; Goodfellow, I.; Boneh, D.; McDaniel, P. Ensemble adversarial training: Attacks and defenses. *arXiv* **2017**, arXiv:1705.07204.
50. Wong, E.; Rice, L.; Kolter, J.Z. Fast is better than free: Revisiting adversarial training. *arXiv* **2020**, arXiv:2001.03994.
51. Sharafaldin, I.; Lashkari, A.H.; Ghorbani, A.A. 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), Funchal, Madeira, Portugal, 22–24 January 2018; pp. 108–116.
52. Yuan, X.; Li, C.; Li, X. DeepDefense: Identifying DDoS attack via deep learning. In Proceedings of the 2017 IEEE International Conference on Smart Computing (SMARTCOMP), Hong Kong, China, 29–31 May 2017; pp. 1–8.
53. Doriguzzi-Corin, R.; Millar, S.; Scott-Hayward, S.; Martínez-del-Rincón, J.; Siracusa, D. Lucid: A Practical, Lightweight Deep Learning Solution for DDoS Attack Detection. *IEEE Trans. Netw. Serv. Manag.* **2020**, *17*, 876–889. [CrossRef]
54. Preuveneers, D.; Joosen, W. TATIS: Trustworthy APIs for Threat Intelligence Sharing with UMA and CP-ABE. In Proceedings of the 12th International Symposium, FPS 2019, Toulouse, France, 5–7 November 2019.
55. Bethencourt, J.; Sahai, A.; Waters, B. Ciphertext-policy attribute-based encryption. In Proceedings of the 2007 IEEE Symposium on Security and Privacy (SP'07), Berkeley, CA, USA, 20–23 May 2007; pp. 321–334.

56. Preuveneers, D.; Joosen, W.; Bernal Bernabe, J.; Skarmeta, A. Distributed Security Framework for Reliable Threat Intelligence Sharing. *Secur. Commun. Netw.* **2020**, *2020*, 8833765. [CrossRef]