

## Article

# INTIME: A Machine Learning-Based Framework for Gathering and Leveraging Web Data to Cyber-Threat Intelligence

Paris Koloveas \*, Thanasis Chantzios, Sofia Alevizopoulou , Spiros Skiadopoulos   
and Christos Tryfonopoulos \*

Department of Informatics & Telecommunications, University of the Peloponnese, GR22131 Tripolis, Greece;  
tchantzios@uop.gr (T.C.); dsc17002@uop.gr (S.A.); spiros@uop.gr (S.S.)

\* Correspondence: pkoloveas@uop.gr (P.K.); trifon@uop.gr (C.T.)

**Abstract:** In today's world, technology has become deep-rooted and more accessible than ever over a plethora of different devices and platforms, ranging from company servers and commodity PCs to mobile phones and wearables, interconnecting a wide range of stakeholders such as households, organizations and critical infrastructures. The sheer volume and variety of the different operating systems, the device particularities, the various usage domains and the accessibility-ready nature of the platforms creates a vast and complex threat landscape that is difficult to contain. Staying on top of these evolving cyber-threats has become an increasingly difficult task that presently relies heavily on collecting and utilising cyber-threat intelligence before an attack (or at least shortly after, to minimize the damage) and entails the collection, analysis, leveraging and sharing of huge volumes of data. In this work, we put forward INTIME, a machine learning-based integrated framework that provides an holistic view in the cyber-threat intelligence process and allows security analysts to easily identify, collect, analyse, extract, integrate, and share cyber-threat intelligence from a wide variety of online sources including clear/deep/dark web sites, forums and marketplaces, popular social networks, trusted structured sources (e.g., known security databases), or other datastore types (e.g., paste bins). INTIME is a zero-administration, open-source, integrated framework that enables security analysts and security stakeholders to (i) easily deploy a wide variety of data acquisition services (such as focused web crawlers, site scrapers, domain downloaders, social media monitors), (ii) automatically rank the collected content according to its potential to contain useful intelligence, (iii) identify and extract cyber-threat intelligence and security artifacts via automated natural language understanding processes, (iv) leverage the identified intelligence to actionable items by semi-automatic entity disambiguation, linkage and correlation, and (v) manage, share or collaborate on the stored intelligence via open standards and intuitive tools. To the best of our knowledge, this is the first solution in the literature to provide an end-to-end cyber-threat intelligence management platform that is able to support the complete threat lifecycle via an integrated, simple-to-use, yet extensible framework.

**Keywords:** IoT; cyber-security; cyber-threat intelligence; crawling architecture; machine learning; language models



**Citation:** Koloveas, P.; Chantzios, T.; Alevizopoulou, S.; Skiadopoulos, S.; Tryfonopoulos, C. INTIME: A Machine Learning-Based Framework for Gathering and Leveraging Web Data to Cyber-Threat Intelligence. *Electronics* **2021**, *10*, 818. <https://doi.org/10.3390/electronics10070818>

Academic Editors: Stavros Shiaeles, Bogdan Ghita and Nicholas Kolokotronis

Received: 20 February 2021

Accepted: 24 March 2021

Published: 30 March 2021

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Over the years, cyber-threats have increased in numbers and sophistication; adversaries now use a vast set of tools and tactics to attack their victims with their motivations ranging from intelligence collection to destruction or financial gain. Thus, organizations worldwide, from governments to public and corporate enterprises, are under constant threat by these evolving cyber-attacks. Lately, the utilisation of Internet of Things (IoT) devices on several applications, ranging from home automation to monitoring of critical infrastructures, has created an even more complicated cyber-defence landscape. The sheer number of IoT devices deployed globally, most of which are readily accessible and easily hacked, allows threat actors to use them as the cyber-weapon delivery system of choice

in many of today's cyber-attacks, ranging from botnet-building for Distributed Denial-of-Service (DDoS) attacks [1–4], to malware spreading and spamming.

Trying to stay on top of these evolving cyber-threats has become an increasingly difficult task, and timeliness in the delivery of relevant cyber-threat related information is essential for appropriate protection and mitigation. Such information is typically leveraged from collected data, and includes zero-day vulnerabilities and exploits, indicators (system artifacts or observables associated with an attack), security alerts, threat intelligence reports, as well as recommended security tool configurations, and is often referred to as *cyber-threat intelligence* (CTI). To this end, with the term CTI we typically refer to any information that may help an organisation identify, assess, monitor, and respond to cyber-threats. In the era of big data, it is important to note that the term intelligence does not typically refer to the data itself, but rather to information that has been *collected, analyzed, leveraged* and *converted* to a series of actions that may be followed upon, i.e., has become *actionable*.

While CTI may be collected by resorting to a variety of means (e.g., monitoring cyber-feeds) and from a variety of sources; in this work we are particularly interested in gathering CTI from the *clear, social, deep* and *dark web* where threat actors collaborate, communicate and plan cyber-attacks. Such an approach allows us to provide visibility to several (structured and unstructured) sources that are of preference to threat-actors or security analysts and identify timely CTI including zero-day vulnerabilities and exploits. To do so, we envisioned and designed INTIME; an *integrated framework for Threat Intelligence Mining and Extraction* that encompasses key technologies for pre-reconnaissance CTI *gathering, analysis, management* and *sharing* through the use of state-of-the-art tools and technologies. In this context, newly discovered data from various sources are inspected for their relevance to the task (*gathering*), the corresponding CTI in the form of vulnerabilities, exploits, threat actors, or cyber-crime tools is explored (*analysis*), discovered CTI is identified and if possible consolidated with existing CTI (*management*), and leveraged information is stored and shared via a vulnerability database (*sharing*).

In this work, we describe in detail the architecture, implementation, and evaluation of the various tools, methods and algorithms utilised, developed, and tested for the task of *gathering, analyzing* and *sharing* of public CTI information from existing security sources, deepnet web forums/marketplaces, and clearnet social platforms. To the best of our knowledge, INTIME is the first approach to an *holistic CTI platform* that is able to support the complete CTI gathering, analysis, management and sharing lifecycle via an integrated, simple-to-use, yet extensible framework. The proposed platform supports the complete CTI lifecycle by:

- Collecting public cyber-threat intelligence data from the social, clear, deep and dark web, including related forums, marketplaces, security-related websites and public databases via specialized ready-to-use machine learning-based tools.
- Ranking crawled content to assess its potential relevance and usefulness to the task at hand; it comes with pre-trained models and utility functions that cover basic needs and is customizable to more specific tasks.
- Extracting CTI from the collected content that was classified as useful, by resorting to state-of-the-art natural language understanding and named entity recognition techniques.
- Managing and sharing collected CTI via a combination of custom-made and widely adopted, state-of-the-art solutions that allow the exploration, consolidation, visualization, and seamless sharing of CTI across different organizations.

The novelty of our approach, INTIME, extends beyond the functional characteristics, by introducing novel views and techniques in the architectural part and in the individual methods that were employed and orchestrated to create the final platform. Specifically:

- CTI data collection goes *beyond any currently available open-source platform*, by providing a *novel family of crawlers, social media monitoring tools, and targeted web scrappers* that specialize in collecting a wide variety of data. The designed and developed family

of crawlers supports a wide variety of crawling options that include *focused/topical crawling* directed by appropriate machine learning methods, *downloading of entire domains* based on powerful, yet easy to set up in-depth crawlers, and *TOR-based dark web spidering*. The social media monitoring tool allows users to monitor popular social media streams for content of interest, by utilising publicly available APIs from social platforms and providing a pre-trained, ready-to-use set of *classification algorithms* to distinguish between relevant and non-relevant content. Finally, the targeted web scrapers provide access to structured data from *reputable sources* that do not provide a data feed capability. This is achieved by developing a ready-to-use toolkit for the most prominent sources and by providing appropriate tools for the extension and incorporation of new sources. All content is stored in scalable NoSQL datastores that provide the necessary flexibility, efficiency and interoperability with the rest of the components.

- Ranking of the collected data employs novel, purpose-specific *statistical language modeling* techniques to represent all information in a latent low-dimensional feature space and a novel *ranking-based approach* to the collected content (i.e., rank it according to its potential to be useful) modeled as vector similarity. These techniques allow us to train our language models to (i) capture and exploit the most salient words for the given task by building upon user conversations, (ii) compute the semantic relatedness between the crawled content and the task of CTI collection by leveraging the identified salient words, and (iii) classify the crawled content according to its relevance/usefulness based on its semantic similarity to CTI gathering.
- The designed CTI extraction solution goes beyond (the typical for the task) rule-based matching by introducing *natural language understanding* and *named entity recognition* to the CTI domain. To do so, it introduces domain-specific entities, employs dependency parsing for more accurate extraction, and provides a set of specialized tools for aiding semi-automated linking to known platform/vulnerability naming schemes.
- The CTI management and sharing is achieved by appropriately extending MISP [5,6]; a state-of-the-art platform for CTI storage and sharing. The integrated technological solution provides enhanced and scalable CTI management functionality, such as CTI storage, consolidation, exploration, inspection and visualization, and supports CTI sharing in both human and machine-based formats to interested stakeholders.

Finally, note that INTIME and all the presented methods and tools therein have been entirely designed on and developed by relying on open-source software including an open-source focused crawler, an open-source implementation of word embeddings for the latent topic modeling, open-source natural language understanding tools, open standards and tools for CTI sharing, and open-source datastores for all storage requirements. INTIME is currently under alpha testing in the context of the CYBER-TRUST project; its deployment for the general public and use of the source code is an issue of project exploitation and will be discussed by the consortium at the end of the project.

The rest of this article is organized as follows. Section 2 discusses the existing work related to each sub-task of INTIME. Section 3 presents a high-level view of the overall architecture, including an overview description for each of the subcomponents developed, while Section 4 provides an indicative case study of INTIME within the CYBER-TRUST [7] project. Section 5 provides evaluation details and operational statistics for the various INTIME components. Finally, Section 6 concludes this article and provides future research directions.

## 2. Related Work

In this section, we review the state-of-the-art in crawling by (i) outlining typical architectural alternatives that fit the crawling task, (ii) categorizing the available crawling solutions according to their policy for traversing the web graph, and (iii) categorizing the different crawler types based on the type of the targeted content. Afterwards, we take a look into Social Media monitoring systems and the several aspects of their usage

for the purposes of data collection. Then, we outline some more specific information extraction approaches centered around the discovery of CTI. Finally, we present several threat intelligence services and CTI sharing platforms and tools.

### 2.1. Crawler Architectures

Conceptually, the typical procedure followed by a web crawler is fundamentally simple: it visits a URL, downloads the webpage associated with it, extracts the URLs therein, compares them with a list of visited URLs and adds the non-visited ones to its frontier list. This procedure is repeated until a (sub-)domain is fully crawled. Harnessing the vast scale of online resources and dealing with varying types of content and information sources requires careful engineering and involves important architectural decisions. Depending on the crawling application, the available hardware, and the desired scalability properties, related literature provides several architectural alternatives [8–10].

**Centralized.** In a centralized architecture [8], the page downloading, URL manipulation, and page storage modules resort in a single machine. This type of architecture is easier to implement, deploy, and administer, but is limited to the capabilities of the hardware and thus cannot scale well.

**Parallel/Distributed/Peer-to-peer.** A parallel crawler [11,12] consists of multiple crawling processes, where each one performs all the basic tasks of a crawler. To benefit from the parallelization of the crawling task, the frontier is split among the different processes, while links and other metadata are communicated between the processes to minimize overlap in the crawled space. When the processes run at geographically distributed locations we call this type of crawler distributed. Peer-to-peer crawlers [13,14] constitute a special form of distributed crawlers that are targeted to be run on machines at the edge of the Internet, as opposed to their distributed counterparts that are designed for clusters and server farms. To this end, peer-to-peer crawlers are lightweight processes that emphasize crawl personalization and demonstrate large-scale collaboration usually by means of an underlying distributed routing infrastructure (e.g., a DHT [15], or a super-peer network [13]).

**Hybrid.** Hybrid crawler architectures [16] are the norm in the architectural typology as they aim for a simple design that involves distributing some of the processes, while keeping others centralized. In such architectures, the page downloading module is distributed, while URL management data structures and modules are maintained at a single machine for consistency. Such designs aim at harnessing the control of a centralized architecture and the scalability of a distributed system.

**Cloud-based.** The cloud-based crawlers [17,18], revived known machinery to a renewed scope, versatility and options. Such architectures use cloud computing features alongside big data solutions such as Map/Reduce and NoSQL databases, to allow for resource adaptable web crawling and serve the modern the Data-as-a-Service (DaaS) concept.

#### 2.1.1. Policy-Based Typology

Since the main task of a crawler is to traverse the webpage graph and collect information, crawlers are also categorized by the way the web graph is traversed.

**Universal (general-purpose).** General-purpose crawlers are mainly meant to support search engines and typically aim at large-scale crawls; the typical universal crawler policy has to take into account trade-offs regarding thematic coverage, page freshness, and content/page bias.

**Incremental.** To minimize the impact of the inconsistencies between the web and the crawled content, URLs should periodically be prioritized and revisited. The simplest way to do this is periodic crawling, but this implies a waste of resources by revisiting pages that have not changed. The informed process of prioritizing and revisiting URLs that are more likely to have changed since the last time they were crawled, is usually referred to as incremental crawling [19] (see also [20] for a comprehensive survey).

**Preferential (focused/topical).** Since most crawlers can download only a small subset of the Web, they need to carefully decide what page to download. By retrieving important or thematically relevant pages early, a focused crawler [21,22] can improve the quality of the downloaded pages. The studies concerning this category explore how a crawler can discover and identify important webpages or pages relevant to a topic [23] or user interest [10,24–27], and propose various algorithms to achieve this goal.

### 2.1.2. Usage Typology

Although the first web crawlers that set the pathway for the spidering technology were developed for the clear (or surface) web, over time specialized solutions aiming at the different facets (social, deep, dark) of the web were gradually introduced. Below we organise crawlers in terms of their intended usage.

**Clear/surface web.** Since the introduction of the first crawler [28] in 1993, the majority of the research work on crawlers has focused on the crawling of the surface web, initially on behalf of search engines, and gradually also for other tasks. There is an abundance of work on Clear Web crawling; some insightful surveys on the topic include [8,20].

**Web 2.0.** The advent of the user-generated content philosophy and the participatory culture that was brought by Web 2.0 sites such as blogs, forums and social media, formed a new generation of specialized crawlers that focused on forum [29–34], blog/microblog [35,36], and social media [37–40] spidering. The need for specialized crawlers for these websites emerged from the quality and creation rate of content usually found in forums/blogs, the well-defined structure that is inherent in forums/blogs that makes it possible to even develop frameworks for creating blog crawlers [41], and the implementation particularities that make other types of crawlers inappropriate or inefficient for the task.

**Deep/Dark/Hidden web.** The amount of information and the inherent interest for data that reside out of reach of major search engines, hidden either behind special access websites (Deep Web) or anonymization networks such as TOR [42] and I2P [43] (Dark Web), gave rise to specialized crawlers that are able to traverse these hidden regions of the web [44]. To this end, over the last ten years, a number of works related to Deep Web crawling have been published [45–49], investigating also different architectural [50–52] and automation [53] options, quality issues such as sampling [54], query-based exploration [55], duplicate elimination [48], and new application domains [56,57].

**Social media.** Recently emerged as one of the most popular CTI sources, social media has lately been utilised to collect knowledge about vulnerabilities, threats and incidents. Several monitoring systems have been used for *CTI detection*, where tweets related to security vulnerabilities were collected and subsequently analyzed by using supervised machine learning techniques. Such systems detect security threats on Twitter stream data and alert the users for their existence by generating appropriate warnings [58,59]. Moreover, Twitter monitoring systems are used to generate a continuously updated summary of the threat landscape [60], to identify the major categories in software vulnerabilities, to understand the factors that impact the retweeting of software vulnerability posts [61], and to identify groups of users whose posts were information-rich regarding vulnerabilities and real-world exploits [62]. Also, the fact that the information shared on Twitter usually appears earlier than any official announcement, can improve the organizations' reaction to newly discovered vulnerabilities [62,63]. Gathered cyber-threat intelligence data from Twitter suggest that cyber-threat relevant tweets do not often include a CVE [64] identifier [65].

The developed INTIME crawlers fall into the hybrid architecture typology, since they allow users to deploy many crawlers in parallel, while crawl management resorts to centralized mechanisms and storage is based on a NoSQL cloud-based solution. Policy-wise, our approach supports both incremental and preferential crawling, while usage-wise we support all the current modalities available in the literature. The above features make our crawler design unique, since there is no other solution in the literature able to support in a uniform and seamless way all the aforementioned typologies. Moreover, our design



extends the current state-of-the-art by introducing a two-stage content ranking mechanism where a selective classification is initially used to prune the crawl frontier, while a more refined approach based on the collected content is used to decide on its relevance to the task.

## 2.2. Information Extraction for CTI

There is a great variety of information extraction approaches to consider, and they all depend on the source and the type of data that a crawler collects. Some indicative approaches are described below.

Husari et al. [66] introduced *TTPDrill*, a crawler that collects CTI reports written in unstructured text and maps them to attack patterns and techniques (TTPs) of a Threat-Action ontology, based on MITRE's CAPEC [67] and ATT&CK [68] threat repository. To filter out which of the crawled pages contained useful information and discard non-relevant pages, an SVM document classifier was built, using features such as, *Number of words*, *SecurityAction-word density*, a metric that computes the percentage of security-related verbs that appear compared to the total number of words, and *SecurityTarget-word density*, a metric that computes the percentage of security-related nouns that appear compared to the total number of words. Apart from the classification method, a set of regular expressions for common objects in the ontology such as IP address, port number, domain, etc. was also used. Also, synonyms such as "logs" and "records", are checked using WordNet, Thesaurus, and Watson Synonym to improve the text understanding. *TTPDrill* then identifies candidate threat actions (grammatical structures of Subject-Verb-Object) from the threat reports and maps them to the ontology by comparing the similarity between the candidate actions and known threat actions in the ontology.

Another project called iACE [69], was tasked with identifying semantic elements (Named Entity Recognition, or NER) and extracting relations between them (Relation Extraction, or RE). The type of data that the authors wanted to collect were *Indicators of Compromise* or IOC. Simple approaches for automatic collection of IOCs, like finding IP, MD5 and other IOC-like strings in an article, do not work well in practice, which easily brings in false positives, mistaking non-IOCs for IOCs. It was observed that the open intelligence sources that produce high-quality IOCs, describe IOCs in a simple and straightforward manner, using a fixed set of context terms which are related to the *iocterms* used in the OpenIOC format to label the types of IOCs [70]. Also, the grammatical connections between such terms and their corresponding IOCs are also quite stable. This allows iACE to leverage relations to identify an IOC and its context tokens, combining the NER and RE steps together. iACE also utilises a set of regular expressions and common context terms extracted from *iocterms* to locate the sentences that contain IOC tokens. Within each of the sentences, the approach attempts to establish a relation between the IOC token and the context term: it converts the sentence into a Dependency Graph to describe its grammatical structure and extracts the shortest paths linking each pair of a context token and the IOC token; over each path, a graph mining technique is applied to analyse the relation between the tokens, for the purpose of determining whether they include an IOC and its context.

INTIME faces tasks similar to the projects outlined in this section in the Data Analysis module (Sections 3.2 and 4.2). Instead of text that follows a specific structure (e.g., CTI reports, IOCs), our approach extends to any unstructured free form text that could potentially contain CTI. We achieve this by utilising word embeddings to assess the similarity of a text to our topic and rank it accordingly and by extending the established NER task with specific rules for higher accuracy when detecting cyber-security-related entities.

## 2.3. CTI Sharing

In this section, we review several solutions related to the discovery and management of CTI, and present their main features and characteristics, with respect to several aspects, such as their architecture, offered services, standards' adoption, and mode of operation. Specifically, in the following, we review *CTI sharing tools and platforms*, *threat intelligence services* and *threat intelligence platforms*. As shown below, INTIME is currently at the frontier

of functionalities, features and supported technologies regarding CTI management and sharing, offering more features than open-source solutions while matching in functionality the most advanced proprietary systems.

### 2.3.1. CTI Sharing Tools and Platforms

In this subsection, we will evaluate five open-source platforms and tools that implement common sharing standards, to facilitate the CTI analysis, sharing and reviewing. Specifically, we consider Open Source Threat Intelligence Platform & Open Standards For Threat Information Sharing (MISP [6]), Open Source Threat Intelligence Gathering and Processing Framework (GOSINT [71]), Your Everyday Threat Intelligence (YETI [72]), a Python implementation of TAXII Services (OpenTAXII [73]) and Collective Intelligence Framework (CIF [74]).

As presented in Table 1, all of the outlined open-source platforms and tools provide CTI in both human and machine readable formats, and they also provide an API. The *Tagging mechanism* refers to the capabilities of the platforms and tools to tag stored artifacts with additional metadata. Next, the *Correlation mechanism* indicates the ability of platforms and tools to intercorrelate the stored CTI. Specifically, MISP provides a correlation engine that is able to automatically intercorrelate CTI, while YETI provides a GUI that enables users to manually create correlations between artifacts. Finally, the *Data formats* illustrates the supported data exchange formats of each platform and tool.

**Table 1.** Cyber-threat intelligence(CTI) sharing tools and platforms.

	MISP	GOSINT	YETI	OpenTAXII	CIF	INTIME
Human & machine readable	✓	✓	✓	✓	✓	✓
API	✓	✓	✓	✓	✓	✓
Tagging mechanism	✓	✓	✓	✗	✓	✓
Correlation mechanism	automatically	✗	manually	✗	✗	automatically, with CPE auto-suggestion
Data formats	CSV, XML, JSON, OpenIOC, STIX/TAXII, and more	STIX/TAXII, VERIS, IODEF, IDMEF	JSON	TAXII	JSON, CSV	CSV, XML, JSON, OpenIOC, STIX/TAXII, and more

### 2.3.2. Threat Intelligence Services

There is a plethora of threat intelligence services that are able to support the CTI sharing life-cycle in various ways. In this subsection, we present several representative threat intelligence services; namely: ZeroFox [75], CTAC by Wapack Labs [76], SearchLight by Digital Shadows [77], Intel 471 [78], Flashpoint Intelligence Platform [79], Security Ratings by BitSight [80], BreachAlert by SKURIO [81] and F5 Labs [82].

As presented in Table 2, all of the presented threat intelligence services provide CTI reports. The *organisation assets' tracking* refers to the capabilities of each threat intelligence service to monitor specific CTI, related to an organisation's assets. Finally, the *Alerting mechanism* indicates whether the threat intelligence service is able to alert its users about events that may concern them.

**Table 2.** Threat intelligence services.

	ZeroFox	CTAC	SearchLight	Intel 471	Security Ratings	Flashpoint	BreachAlert	F5 Labs	INTIME
Vulnerabilities tracking	✓	✓	✓	✓	✓	✓	✗	✓	✓
TTPs tracking	✓	✓	✗	✓	✓	✓	✗	✓	✓
Organisation assets' tracking	✓	✓	✓	✓	✓	✗	✓	✗	✓
Deep/dark web monitoring	✓	✓	✓	✓	✗	✓	✓	✓	✓
Social media monitoring	✓	✓	✓	✗	✓	✗	✓	✗	✓
CTI reports	✓	✓	✓	✓	✓	✓	✓	✓	✓
API	✓	✓	✓	✓	✓	✓	✓	✗	✓
Alerting mechanism	✓	✗	✓	✓	✓	✓	✓	✗	✓

### 2.3.3. Threat Intelligence Platforms

As with the threat intelligence services presented previously, there are numerous threat intelligence platforms, which aim to organise and manage CTI in a centralized manner and also integrate them with other security solutions. In this subsection we present numerous aspects of representative threat intelligence platforms; namely: Helix by FireEye [83], Recorded Future [84], Cyjax [85], EclecticIQ [86], Cyber Advisor by SurfWatch Labs [87], BloxOne by Infoblox [88], ThreatStream by Anomali [89], ThreatQ by ThreatQuotient [90], Soltra by Celerium [91], ThreatConnect [92], VDMR by Qualys [93], MANTIS by SIEMENS [94] and BrightCloud by Webroot [95].

All of the aforementioned threat intelligence platforms provide an API and search capabilities over their CTI. Table 3 presents the aspects of the threat intelligence platforms. The *CTI intercorrelation* describes whether each platform is intercorrelating the stored CTI artifacts. Next, the *Remediation proposals* indicates the ability of the platforms to propose remediation strategies and techniques over vulnerable configurations or breached assets. Finally, *SaaS* and *On-premises* present the type of the platforms, whether they are available as a web-hosted service or executed locally, respectively.

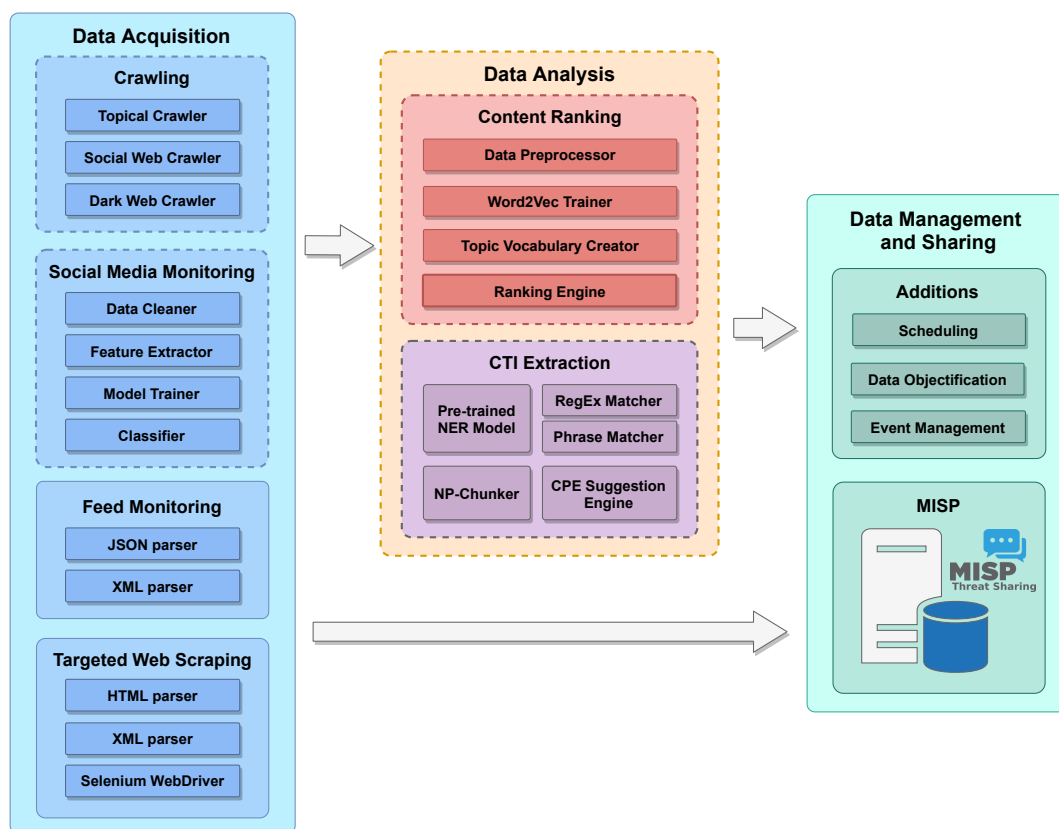


**Table 3.** Threat intelligence platforms.[illegible]

### 3. System Architecture

The architecture of our system, INTIME, consists of three major components. The *Data Acquisition module* is responsible for the monitoring and crawling of various web resources by employing traditional crawling and scraping techniques, along with machine learning-assisted components to direct the crawl to relevant sources. Although this module can easily extract information from specific well-structured sources, further analysis is required when it comes to web content crawled from unstructured/semi-structured sources. To further analyse the gathered content, the *Data Analysis module* hosts two machine learning-based submodules, the *Content Ranking submodule*, which acts as an internal filter that ranks the data according to their relevance to the topic at hand, and the *CTI Extraction submodule*, that employs several information extraction techniques to extract useful information from the webpages that were deemed relevant. The idea behind this two-stage approach stems from the inability of a simple crawler to accurately model the openness of the topic. The difficulty emerges from websites that, although relevant to the topic (e.g., discussing IoT security in general), have no actual information that may be leveraged to actionable intelligence (e.g., do not mention any specific IoT-related vulnerability). After the analysis, the extracted information is passed to the last module, *Data Management and Sharing*, which hosts all the Cyber-Threat Intelligence that the system gathers.

Machine Learning and Deep Learning have a central role in our architecture, as the entire *Data Analysis module* is built upon DL techniques such as Word Embeddings (content ranking) and Named Entity Recognition (CTI extraction). Also, the the *Data Acquisition module* uses traditional ML algorithms to classify gathered content in the *Crawling* and *Social Media Monitoring* submodules. In Figure 1, every module where Machine Learning methods are present, is enclosed in dashed lines.



**Figure 1.** A high-level view of INTIME's architecture (components in dashed lines employ machine/deep learning techniques).

Methodologically we followed a four-stage approach to developing INTIME: (i) in the *analysis phase* we explored the state-of-the-art in cyber-security and CTI management, defined use case scenarios and gathered end-user requirements, (ii) in the *research phase* we designed advanced novel methods that allowed INTIME to achieve its mission regarding proactive technologies and cyber-attack detection, (iii) in the *development phase* we implemented, integrated, and tested the tools, components, and services of the platform, and (iv) in the *validation phase* we developed initial evaluation plans on real-world datasets and scenarios, and designed the more detailed evaluation of the individual components. The first iteration of this architecture was initially developed by Koloveas et al. [96], while specifically focusing on the crawling and ranking tasks. In this section, we outline the architecture of the proposed system, as illustrated in Figure 1.

the input from the end-users is essential for defining the platform's functionalities/services; the developed methods must take into account related international activities (listed below); § the software tools will be open source and must conform to existing or upcoming standards; § the final platform must be assessed against performance and regulatory compliance criteria.

### 3.1. Data Acquisition Module

Currently useful cyber-security-related information that may be leveraged to actionable intelligence may be found in a vast variety of different online sources ranging from technical or security-focused blogs in the clear web, discussions between experts in specialized security forums, social media content, to underground dark web hacker forums and marketplaces selling cybercrime tools and zero-day vulnerabilities and exploits. To cover this widespread need for data acquisition, our architecture provides a flexible yet powerful *Data Acquisition Module* that is conceptually separated into four distinct submodules:

- *Crawling* allows users to easily setup and deploy automated data collection crawlers that are able to navigate the clear, social, and dark web to discover and harvest content of interest. The *Crawling* submodule allows the user to select between a wide variety of options including focused/topical crawling directed by appropriate machine learning methods, downloading of entire domains based on powerful, yet easy to set up in-depth crawlers, TOR-based dark web spidering, and semi-automated handling of authentication methods based on cookie management. After collecting the content of interest, the users may then use rest of the modules provided by our architecture to further process it to extract useful CTI from it. The *Crawling* submodule is discussed in more detail in Section 3.1.1.
- *Social Media Monitoring* allows users to observe popular social media streams for content of interest; to do so it uses publicly available APIs from social platforms and provides a pre-trained, ready-to-use set of classification algorithms that may be used to distinguish between relevant and non-relevant content. The *Social Media Monitoring* submodule is elaborated on in Section 3.1.2.
- *Feed Monitoring* allows the users to record structured JSON or RSS-based data feeds from established credible sources such as NIST, while allowing them to modify several monitoring parameters like the monitoring interval and the type of objects they are interested in (e.g., CVEs, CPEs, or CWEs).
- *Targeted Web Scraping* provides tools to access structured data from reputable sources, like KB-Cert Notes by Carnegie Mellon University [97] or Exploit-DB [98], that do not provide a data feed capability. To support this process, our architecture offers a pre-installed set of tools that may be used to assist the programmer, including standard HTML parsing, XPath querying and Javascript handling tools and libraries.

All the data extracted from the *Crawling* and *Social Media Monitoring* submodules, are stored in an internal NoSQL database (MongoDB), where they are analyzed by the *Content Ranking* and *CTI Extraction* submodules of the *Data Analysis* module. Afterwards, they are sent to the *CTI Management module* as structured CTI. Notice that data collected by the *Feed*

*Monitoring* and *Targeted Web Scraping* submodules are directly stored to the *CTI Management module* since they typically come from structured and trusted data sources.

In the following sections, we elaborate further on the submodules that were outlined above.

### 3.1.1. The Crawling Submodule

The Crawling submodule implements several distinct services that may be invoked by the users to initiate automated data collection on a wide variety of online sources in the clear, social, or dark web; the underlying crawling infrastructure is built on NYU's ACHE crawler [99].

The supported *focused crawling* functionality, uses the *SMILE* [100] page classifier, which is based on a machine learning-based text classifier (SVM, Random Forest), trained by a selection of positive and negative examples of webpages, to direct the crawl toward topically relevant websites (i.e., websites with content relevant to cultural heritage). This functionality can also be assisted by the *SeedFinder* [101] sub-component, which can aid the process of locating initial seeds for the focused crawl; this is achieved by combining the pre-built classification model with a user-provided query relevant to the topic.

*In-depth crawling* is essentially a domain downloading operation based on the ACHE crawler to traverse a specific domain (e.g., a forum or a website) in a Breadth-First Search manner and download all webpages therein. To direct the crawl to specific parts of the website/forum *regex-based filters* are used; these filters provide both blacklisting (resp. whitelisting) functionality to direct the crawler away from (resp. toward) specific sections of the website/forum. In this way the user may instruct the crawler to avoid downloading non-informative pages (e.g., members areas, login pages, help) or to actively direct it to specific discussion threads in a forum.

*Dark web crawling* is also supported by our architecture. This functionality relies on the use of TOR [42] proxies to visit the user-specified onion links. Note that the user is not required to have any experience in this procedure, as all required actions (i.e., joining the TOR network, using the proxy, initializing the crawler) are fired automatically via internal API calls.

Finally, authentication issues that may arise during crawling are resolved via manual user login (the first time the crawler encounters an authentication barrier) and session cookie storage for all subsequent crawler visits.

### 3.1.2. The Social Media Monitoring Submodule

The Social Media Monitoring submodule focuses on real-time event detection from social streams using state-of-the-art tools from the data science domain to automatically classify posts as related or unrelated to a user-defined topic. To gather data from social media streams it uses the provided social platform APIs; the user is able to specify a set of social media accounts and/or a set of keywords that are of interest and the content collection mechanism will retrieve (in a recurring publish/subscribe fashion) all content posted from those accounts or matching the provided keywords. Notice that this mechanism has no control over the fetched data, as this is a functionality provided entirely from the API of the supported social media.

Subsequently, the user is able to classify the retrieved content as related or unrelated to the task by simply selecting among various popular classification algorithms including (multinomial) Naive Bayes, K-Nearest Neighbors, Decision Trees, Random Forests, Logistic Regression, Support Vector Machines, as well as proven deep learning models like Convolutional Neural Networks. All classification and machine learning algorithms come pre-trained on real-world data and with default parameter setups for security classification tasks, but the user may modify both the training data and setup parameters to fit its specific classification needs. The above process is streamlined to be usable out-of-the-box, but the advanced user may also customize all parts of the process, including content acquisition

from social media without an API, introduction of other classification or machine learning algorithms, and task-specific algorithm training.

### 3.1.3. Feed Monitoring and Target Web Scraping Submodules

Apart from the unstructured/semi-structured data that are gathered by the functionalities mentioned above, our system, INTIME, may also be supplemented by structured data from reputable sources of CTI. Such sources are divided in two categories.

The first category contain sources that provide structured data feeds (typically in JSON or XML formats) of their information collections. For this category, we have provided mechanisms that use JSON/XML parsing techniques with variable monitoring periods dependent on the data feed's update frequency.

Sources of the second category do not provide data feeds, but expose the contents of their database on web-based interfaces using a structured manner. For this category, we have implemented several scraping techniques providing a flexible set of tools to account the different types of websites where such information exists. These techniques range from HTML parsing and XPath querying, to sophisticated WebDrivers for automatic form manipulation and dynamic pop-up dismissal.

The data acquired from the above types of sources are inserted into the *Data Management and Sharing* module (Section 3.3) without passing through the *Data Analysis* module (Section 3.2), since they are already structured in the CTI form of INTIME.

### 3.2. Data Analysis Module

Deciding whether a collected website contains useful Cyber-Threat Intelligence is a challenging task given the typically generic nature of many websites that discuss general security issues. To tackle this problem, we created an additional processing layer that ranks the collected content to assess its relevance and usefulness to the task at hand.

Ranking of the collected data employs statistical language modeling techniques to represent all information in a latent low-dimensional feature space and a ranking-based approach to the collected content (i.e., rank it according to its potential to be useful) modeled as vector similarity. These techniques allow us to train our language model to

1. capture and exploit the most salient words for the given task by building upon user conversations,
2. compute the semantic relatedness between the crawled content and the task of CTI collection by leveraging the identified salient words, and
3. classify the crawled content according to its relevance/usefulness based on its semantic similarity to CTI gathering.

The Data Analysis module responsible for the aforementioned mechanism is the *Content Ranking Engine*, which will be further elaborated on in Section 4.2.1.

After the *Content Ranking Engine* decides which of the collected websites are more likely to contain Cyber-Threat Intelligence, the final step in the analysis process is to extract that CTI. To do so, our designed CTI extraction solution extends the Named Entity Recognition task to the CTI domain by introducing domain-specific entities, adapting generic pre-trained language models, introducing dependency parsing, and creating specialized tools for aiding semi-automated linking to known platform/vulnerability naming schemes. The module responsible for this part of the Data Analysis is called *CTI Extraction*, and it will be outlined in Section 4.2.2.

### 3.3. Data Management and Sharing Module

This module provides a variety of tools and functionalities, which are able to support the CTI life-cycle procedure. Through this module, INTIME:

- Identifies the discovered CTI from the Data Acquisition and Data Analysis modules (Sections 3.1 and 3.2) and checks if it is new or it can be used to augment already stored CTI. In the former case discovered CTI is added as new; in the latter case, existing CTI is appropriately extended and updated by merging the discovered CTI.



- Stores CTI in a structured, organized and extensible manner.
- Correlates all CTI stored giving security experts the ability to browse the cyber landscape.
- Provides a UI which implements appropriate tools for the presentation, management, analysis and review of the stored CTI.
- Implements a sharing mechanism, that is able to export CTI in a plethora of CTI sharing standard formats.

The Data Management and Sharing module of INTIME extends MISP [5,6] that takes the lead in the CTI life-cycle support platforms' race [102,103]. MISP models CTI using security *events* that contain *objects* and *attributes*. These artifacts are transparently stored within the internal database of MISP. The stored information is available in both human and machine-readable formats. Users of MISP may access stored information either through a graphical interface or a REST API. Additionally, MISP includes various tools, that enable users to review gathered CTI, comment and categorise it (using proposals, tags and taxonomies), eliminate false positives, comment on the artifacts, and further analyse and enrich CTI through the correlation process. Finally, MISP provides a synchronization protocol that allows information exchange, which supports four main features, namely *pull*, *push*, *cherry-picking* and *feed*.

To accommodate the INTIME architecture requirements, we have appropriately extended MISP to act as an efficient CTI data store that is able to manage the artifacts stemming from the Data Acquisition and Data Analysis modules (Sections 3.1 and 3.2). More specifically, the Data Management and Sharing module is able to not only to store the artifacts in a structured manner through the use of the MISP functionalities, but also enrich, update or modify any existing CTI knowledge. Thus, within INTIME, MISP is able to act as a constantly augmented knowledge database for CTI.

In more detail, the Data Management and Sharing module of INTIME (Figure 1) contains a *Scheduling* submodule that organizes data collection according to the needs of the identified sources' monitoring. For example, we may schedule to read data feeds on specific intervals, or even schedule to scrape online sources that provide a fixed number of entries. Then, the *Data Objectification* submodule considers the gathered CTI and transforms it into objects that are able to be stored in MISP. Finally, the *Event Management* submodule searches through the MISP database for occurrences regarding the structured event, in order to determine if a new event should be created in MISP, or if an existing event should be modified (updated or enriched), enclosing all objects created by the previous submodule into a unified event. All the above submodules are custom-built using Python.

### 3.4. Implementation Aspects

The architecture has been entirely designed on and developed using open-source software. In this section we briefly outline the technology stack of the system, while pointing out noteworthy aspects of our implementation.

The *Data Acquisition* and *Data Analysis* modules are powered by a REST API that is responsible for their previously discussed functionalities. The API has been built on the Flask web application framework [104] and is powered by Swagger, a specification language for describing RESTful APIs in JSON [105]. The API generates Docker configuration files based on user input to activate containers that correspond to each module. The backend of the API uses MongoDB, a NoSQL document-oriented database [106], to store all the crawled documents, and crawler configurations. Internally, our crawlers use the ACHE Focused Crawler [99], with a several customisations to accommodate our needs. For the Content Ranking submodule, we used Gensim's [107] implementation of Word2Vec for our word embeddings and spaCy [108] for the CTI Extraction tasks.

The *Data Management and Sharing* module uses and extends MISP [5,6], with MySQL [109] as the backend. MISP's UI is built with CakePHP [110]. For the implementation of the MISP extension that is presented in Section 3.3, the module makes use of the PyMISP library [111].

#### 4. CYBER-TRUST Case Study

For the CYBER-TRUST [7] case study, we were tasked to use the previously described tools for the purpose of advanced Cyber-Threat Intelligence discovery related to the *Internet of Things* domain. In this section, we share some details regarding the implementation, configurations, datasets and processes that make our system effective in this particular domain.

##### 4.1. Data Acquisition

As described in Section 3.1, the Data Acquisition module consists of the mechanisms responsible for the continuous monitoring and gathering of relevant web content. Those mechanisms are sets of crawlers, social media monitoring systems, structured feed downloaders and custom source-specific web scrapers. In the following paragraphs, we outline the way that these tools are configured for the effective content collection required for our case study.

##### 4.1.1. Crawling

The *Topical*, *Social Web* and *Dark Web* crawlers are made available through a web-based User Interface that exposes several functions of the internal API. Such actions include the following:

- Create new crawlers,
- Start/Stop crawlers, and
- Configure the Seed URLs, Training URLs, Link Filters, SeedFinder queries (depending on the type of crawler).

Figure 2 displays the main configuration pages of the UI, where the user can create Topical (left) and Social/Dark (right) crawlers. A snapshot of the crawler dashboard can be seen in Figure 3. There, the user can monitor the progress of any selected active crawler. Some examples of positive and negative training URLs for the Topical crawler's classification model are listed in Table 4.

Figure 2. Configuration Pages for Focused and Social/Dark crawlers.

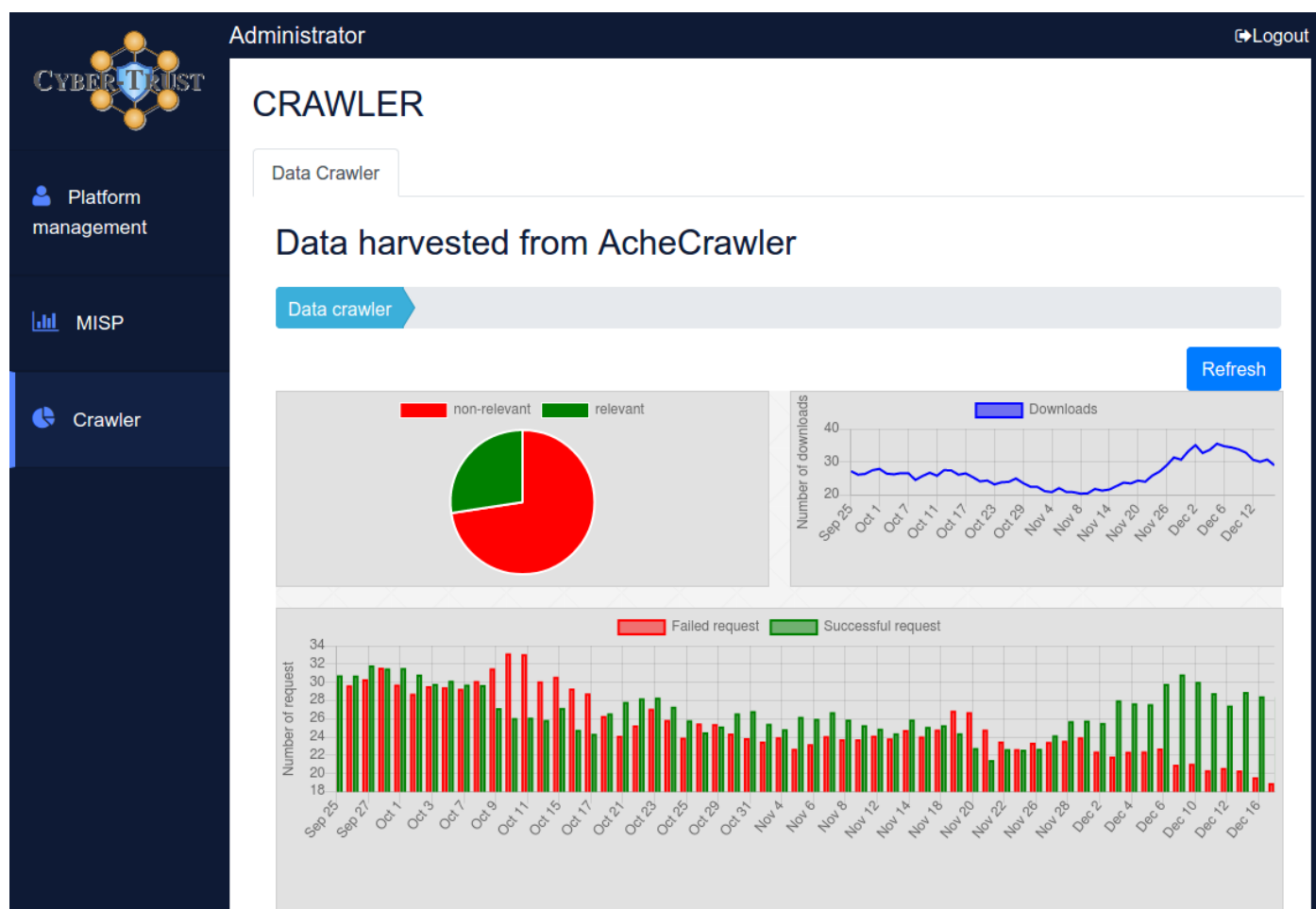


Figure 3. The Crawler Dashboard.

For our case study, we created several *Topical crawlers* focused around “IoT vulnerabilities”, “Android Exploits” and “Exploits available for sale”. We also made several *Social Web crawlers* that monitor famous security forums and blogs. Finally, we created some *Dark Web crawlers* that were geared toward Dark Web Search Engines and Dark Web Marketplaces.

Table 4. Examples of Positive & Negative Training URLs.

Positive	Negative
1. <b>CSO Online</b>   IoT, Cloud, or Mobile: All Ripe for Exploit and Need Security’s Attention.	1. <b>CSO Online</b>   Scammers pose as CNN’s Wolf Blitzer, target security professionals.
2. <b>eSecurity Planet</b>   New IoT Threat Exploits Lack of Encryption in Wireless Keyboards.	2. <b>eSecurity planet</b>   19 top UEBA vendors to protect against insider threats and external attacks.
3. <b>Beyond Security</b>   Security Testing the Internet of Things: Dynamic testing (Fuzzing) for IoT security.	3. <b>SoftLayer</b>   Build your own cloud.

#### 4.1.2. Social Media Monitoring

The social media that we decided to monitor was *Twitter*, as it appears to have a constant stream of relevant, up-to-date information around our topic. Our monitoring scheme consists of two stages: the *data acquisition* and the *classification model* that allows the automatic identification of related and unrelated tweets from Twitter streams. Twitter APIs [112] have been used for data acquisition and machine learning algorithms for the classification task of the collected tweets.

The monitoring system collects live tweets using the Twitter Stream API with keywords related to IoT vulnerabilities such as “*IoTsecurity*”, “*CVE*”, “*exploitIOT*”, “*IOThacking*”, “*attacker*”, etc., as well as, several platform names like “*arduino-uno*”, “*web-server*” etc. as we observed that very often, new exploits were related to these platforms. Each collected tweet is preprocessed and subsequently evaluated by the classification model.

For the training and evaluation process of the classification model, related and unrelated tweets were gathered from specific Twitter users. For the related tweets, we searched Twitter for users with tweets about new or existing IoT vulnerabilities. For instance, some related tweets were from user “*CVEnew*”, which is the official Twitter account maintained by the CVE team to notify the community of new CVE IDs. We also searched for users whose tweets were about IoT devices in general, new trends, updates etc. These tweets did not announce new or existing vulnerabilities, so they were used as the unrelated tweets of the dataset. For example, we collected tweets from users like “*IOTASupport*”, “*iotatoken-news*”, “*MediaTek*”, “*IoTCommunity*”, “*TheIoTWarehouse*”. Apart from the collected tweets, we also used MITRE CVE [64] dictionary as an external database for the training dataset.

On close examination of the training and testing datasets, we realized that all related data contained the *CVE* tag; for that reason, we created two different versions of the datasets, one with the *CVE* tag and one without it. In order to find which classification model and which version of the dataset better suited our requirements, we tested several traditional Machine Learning algorithms, namely Logistic Regression, Multinomial Naive Bayes, Decision Tree Classifier, K-Nearest Neighbors Classifier, Support Vector Machines and Random Forest Classifier, as well as CNN-based models with both datasets. We used Grid Search and 10-fold cross-validation to evaluate the accuracy of the classification models, and we observed that the Random Forest Classifier had the best performance with both datasets. We also noticed that when most classification models were trained with data containing the *CVE* tag, bias was introduced to our models, so we decided to use the datasets that did not contain it.

#### 4.1.3. Feed Monitoring

Apart from the *Crawling* submodule, INTIME develops mechanisms that include information from CTI sources that support information feeds (Section 3.1.3). These sources may be vulnerability and exploit databases that contain analyzed CTI, in the form of vulnerability and exploit reports, available in a feed form. Such reports contain a plethora of useful and actionable intelligence about the vulnerabilities and exploits, such as a description of the vulnerability at hand, an exploit proof-of-concept, a list of the affected products’ configurations (CPEs), metrics that provide an impact factor for the affected product (CVSS), publication and modification dates, references to similar reports, and a unique identifier that has been assigned to the vulnerability at hand (CVE ID).

Examples of such sources include NVD [113] and JVN [114]. Both these sources, provide CTI through structured data feeds (JSON or XML). INTIME is tuned to monitor these sources without encountering any technical difficulties. In more detail:

- *NVD* provides public access to a plethora of data feeds of its contents [115] including vulnerability data, CPE match links, RSS vulnerability information, and official vendor comments on existing vulnerabilities. The vulnerability data feeds contain feeds grouped by the year of the CVE ID (i.e., `nvdCVE-1.1-2020.json`), the most recent entries (`nvdCVE-1.1-recent.json`) and the most recent modifications of past entries (`nvdCVE-1.1-modified.json`). We download the latter two data feeds, which contain the recent and recently modified entries of NVD. Then, through our custom-built Python JSON parser, we extract the CTI of each entry of the JSON files.
- *JVN* provides public access to various data feeds of the JVN iPedia contents [116], that include vulnerability data, CPE match links, and product developer information. As with NVD, the vulnerability data feeds contain feeds grouped by the year of the CVE

ID (i.e., `jvndb_detail_2020.rdf`), the most recent entries (`jvndb_new.rdf`), and the most recent modifications of past entries (`jvndb.rdf`).

We download the latter two data feeds, which contain the recent entries and the modifications of the past entries of JVN. Then, similarly to the NVD Parsing, through our custom-built Python XML parser, we extract the CTI of each entry of the XML files.

#### 4.1.4. Targeted Web Scraping

INTIME is also able to gather information from CTI sources that do not provide information in the form of data feeds (see also Section 3.1.3). To extract the available information we have implemented, tailored HTML crawling and parsing techniques in Python that use XPath queries and the Selenium WebDriver [117]. The latter is used to bypass dynamic pop-ups that confuse parsing.

We have tested our parsers with several certified sources like KB-Cert [97], Exploit-DB [98], VulDB [118], and Oday Today [119]. In all cases, INTIME was able to enrich existing knowledge with CTI artifacts stemming from these sources like for instance (a) temporal and environmental CVSS metrics, (b) exploitation price range estimations and (c) exploit proof-of-concepts.

### 4.2. Data Analysis

As mentioned in Section 3.1, the content from the *Crawling* and *Social Media Monitoring* submodules require further analysis before being sent to the *Data Management and Sharing* module. This is achieved by the *Data Analysis* module, the processing layer responsible for the ranking of the content based on its relevance and the extraction of potentially useful CTI from it.

#### 4.2.1. The Content Ranking Submodule

The idea behind our ranking approach was to represent the topic as a vocabulary distribution by utilising distributional vectors of related words; for example, a topic on IoT security could be captured by related words and phrases like “Mirai botnet”, “IoT”, or “exploit kits”. In this way, we are able to capture semantic dependencies and statistical correlations among words for a given topic and represent them in a low-dimension latent space.

**Language Model.** For the creation of our Language Model, we used Word2Vec [120]; a shallow, two-layer neural network that can be trained to reconstruct linguistic contexts and map semantically similar words close on the *embedding space*. Each word in the embedding space is represented as a *word embedding*. Those word embeddings can capture the relationship between the words in the dataset, making vector arithmetic possible. The above described method, along with a method to map the words to a topic, which will be discussed later, could help us create a *Topic Vocabulary*.

**Dataset.** In order to train the Language Model, we had to create an appropriate dataset for our task. Our dataset had to contain the common vocabulary that is utilised when the topics of *IoT* and *Security* are being discussed. To capture this vocabulary, we resorted to several different discussion forums within the Stack Exchange ecosystem. To this end, we utilised the *Stack Exchange Data Dump* [121] to get access to IoT and information security-related discussion forums including *Internet of Things* [122], *Information Security* [123], *Arduino* [124], and *Raspberry Pi* [125]. The last two were selected because they are the most prominent devices for custom IoT projects with very active communities, so their data would help our model to better incorporate the technical IoT vocabulary. The used data dumps contain user discussions in Q&A form, including the text from *posts*, *comments* and related *tags* in XML format. The files that contain the XML data get periodically downloaded and processed to enrich the dataset and, subsequently, the vocabulary. The *posts* and *comments* were used as input to the *Data Preprocessing* submodule which creates our dataset, seen in Figure 1. The *tags* were used for the *Topic Vocabulary Creator* submodule, which is outlined later in this section.



**Data Preprocessing.** This submodule is responsible for the data normalization process. It performs the typical normalization tasks, namely, sentence tokenization, case folding, symbol removal and contraction removal, as well as, username elimination for data anonymization. Note that we decided to avoid two common normalization actions, namely stemming and lemmatization, as it was evident that by performing such actions we were likely to lose some valuable domain-specific words. For example, the word “wareZ” could be a possible malware name, but if it gets stemmed to “ware” we might lose the chance to discover the related semantic information. Finally, the normalised data get processed by the *MWE Tokenizer*, which, by using NLTK’s [126] *Multi-Word Expression (MWE) Tokenizer*, creates certain multi-word domain-specific terms within the dataset, so Word2Vec would consider them as a single word (e.g., wherever “exploit kits” appears, it becomes “exploit\_kits”).

**Topic Vocabulary.** On several cases, the words of the training dataset can be off-topic, thus, there was the need for a method that removes such words, to create a smaller, more robust, topic-specific vocabulary. To do so, we utilised domain-specific *tags* and augmented them with the set of  $N$  most related terms in the latent space for each tag. Table 5 shows an example of the most relevant terms to the DDoS user tag.

**Table 5.** Most relevant terms for tag DDoS.

Rank	Term	Rank	Term	Rank	Term
#1	volumetric	#6	denial_of_service	#11	slowloris
#2	dos	#7	cloudflare	#12	ip_spoofing
#3	flooding	#8	prolexic	#13	loic
#4	flood	#9	floods	#14	drdos
#5	sloloris	#10	aldos	#15	zombies

**Ranking Engine.** Since useful Cyber-Threat Intelligence manifests itself in the form of cyber-security articles, user posts in security/hacker forums, social media, or advertisement posts in cybercrime marketplaces, it can also be characterized as distributional vectors of words. That way, we can compare the similarity between the distributional vectors of the harvested content and the given topic to assess the relevance and usefulness of the content.

To do so, we employ the *Ranking Engine* submodule. This module first creates the *Topic Vector*, by utilising the generated topic vocabulary and then creates a *Post Vector* for each post entry in the CrawlDB collection.

The *Topic Vector*  $\vec{T}$  is constructed as the sum of the distributional vectors of all the topic terms  $\vec{t}_i$  that exist in the topic vocabulary, i.e.,

$$\vec{T} = \sum_{\forall i} \vec{t}_i \quad (1)$$

Similarly, the *Post Vector*  $\vec{P}$  is constructed as the sum of the distributional vectors of all the post terms  $\vec{w}_j$  that are present in the topic vocabulary. To promote the impact of words related to the topic at hand, we introduce a topic-dependent weighting scheme for post vectors in the spirit of [127]. Namely, for a topic  $T$  and a post containing the set of words  $\{w_1, w_2, \dots\}$ , the post vector is computed as

$$\vec{P} = \sum_{\forall j} \cos(\vec{w}_j, \vec{T}) \cdot \vec{w}_j \quad (2)$$

Finally, after both vectors have been computed, the *Relevance Score*  $r$  between the topic  $T$  and a post  $P$  is computed as the cosine similarity of their respective distributional vectors in the latent space

$$r = \cos(\vec{T}, \vec{P}) \quad (3)$$

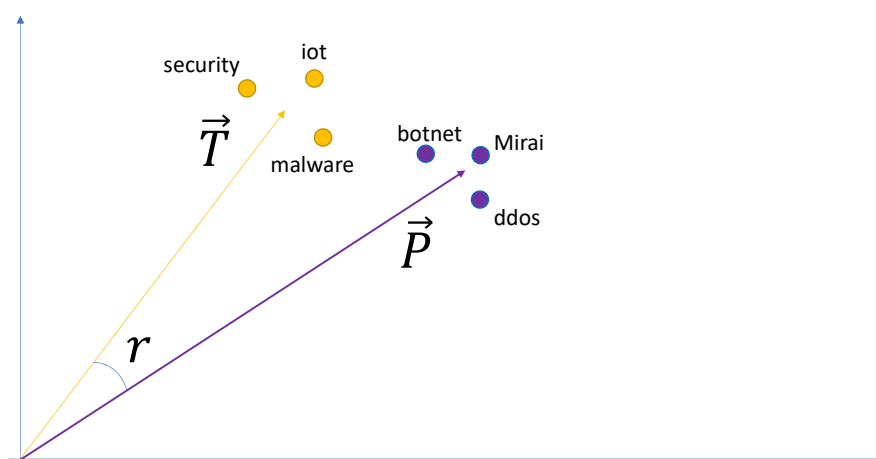
Having computed a relevance score for every crawled post in our datastore, the task of identifying relevant/useful information is trivially reduced to either a thresholding or a

top-k selection operation. The most and least relevant websites can also be used to reinforce the crawler model as input to the *Model Builder* submodule of the *Crawler module*.

Figure 4 shows a theoretical visualization of the ranking process in a 2D vector space for the topic of “IoT security”. Table 6 presents an example of a document that has been processed by the *Ranking Engine*, where the highlighted words have been identified as part of the Topic Vocabulary and the Relevance Score is computed in accordance to the process outlined above.

**Table 6.** Relevance Score computation.

Excerpt from (accessed June 2020):	<a href="http://www.iotforall.com/5-worst-iot-hacking-vulnerabilities">www.iotforall.com/5-worst-iot-hacking-vulnerabilities</a>
The <b>Mirai Botnet</b> (aka Dyn <b>Attack</b> ) Back in October of 2016, the largest <b>DDoS attack</b> ever was launched on service provider Dyn using an <b>IoT botnet</b> . This led to huge portions of the <b>internet</b> going down, including <b>Twitter</b> , the Guardian, <b>Netflix</b> , Reddit, and CNN.	
This <b>IoT botnet</b> was made possible by <b>malware</b> called <b>Mirai</b> . Once <b>infected</b> with <b>Mirai</b> , computers continually search the <b>internet</b> for vulnerable <b>IoT devices</b> and then use known default <b>usernames</b> and <b>passwords</b> to <b>log in</b> , infecting them with <b>malware</b> . These <b>devices</b> were things like digital cameras and <b>DVR players</b> .	
Relevance Score:	0.8563855440900794



**Figure 4.** Theoretical visualization of the ranking process.

#### 4.2.2. The CTI Extraction Submodule

After the collected documents are ranked, the subsequent step of the analysis task is to extract Cyber-Threat Intelligence from them. To this end, INTIME extends the well-known NLP technique of Named Entity Recognition, by introducing domain-specific entities, utilises a dependency parser to perform Noun Phrase Chunking, and with the assistance of a specialized Suggestion Engine, is able to provide information to accommodate the linking of the discovered CTI to existing information of our CTI Sharing solution. In the following paragraphs, we provide some insight to the inner workings of the aforementioned mechanisms.

**Named Entity Recognition.** The primary technique that we decided to use for the task was Named Entity Recognition (NER). When we apply NER on a relevant text, we can identify specific entities that can lead to CTI discovery. To perform the task we used spaCy, a Python library with several ready-to-use tools for NLP.

As the texts that we needed to analyse originated from various sources with not well-defined structures, we concluded that it would not be efficient to train a NER model with entity-annotated data from scratch. Therefore, we decided to use spaCy’s pre-trained model to detect generic entities that were not limited to our topic.

To assist the pre-trained model to find more entities related to the topic, we also utilised spaCy’s “*Phrase Matcher*” functionality, which matches unique multi-word phrases

and maps them to the specified named entites. The phrases that we imported to the model were full names of companies/organizations and products extracted from JVN and were mapped to the *ORG* and *PRODUCT* entities (Table 7).

Finally, apart from the entities that our pre-trained model was able to identify, we needed several domain-specific entities that we had to introduce by ourselves. We achieved that by defining Regular Expressions for our entities, which were added to our NER pipeline via spaCy's "Regex Matcher".

Table 7. Supported Entity Types.

Type	Description	Source
PERSON	People, including fictional.	Pre-trained model
ORG	Companies, agencies, institutions, etc.	Pre-trained model, Phrase-Matcher
PRODUCT	Objects, vehicles, foods, etc.	Pre-trained model, Phrase-Matcher, RegexMatcher
DATE	Absolute or relative dates or periods.	Pre-trained model
TIME	Times smaller than a day.	Pre-trained model
MONEY	Monetary values.	Pre-trained model, Regex-Matcher
CVE	Common Vulnerabilities and Exposures (CVE) identifier.	Regex Matcher
CPE	Common Platform Enumeration (CPE) identifier.	Regex Matcher
CWE	Common Weakness Enumeration (CWE) identifier.	Regex Matcher
CVSS2_VECTOR	Common Vulnerability Scoring System (CVSS) v2.	Regex Matcher
CVSS3_VECTOR	Common Vulnerability Scoring System (CVSS) v3.0-v3.1.	Regex Matcher
IP	IP address.	Regex Matcher
VERSION	Software version.	Regex Matcher
FILE	Filename or file extension.	Regex Matcher
COMMAND/FUNCTION/CONFIG	Shell command, code function or configuration setting.	Regex Matcher

Table 7 shows the entities that our system is able to identify, along with the mechanisms responsible for the identification. Figure 5 shows some identified entities on a sample text.

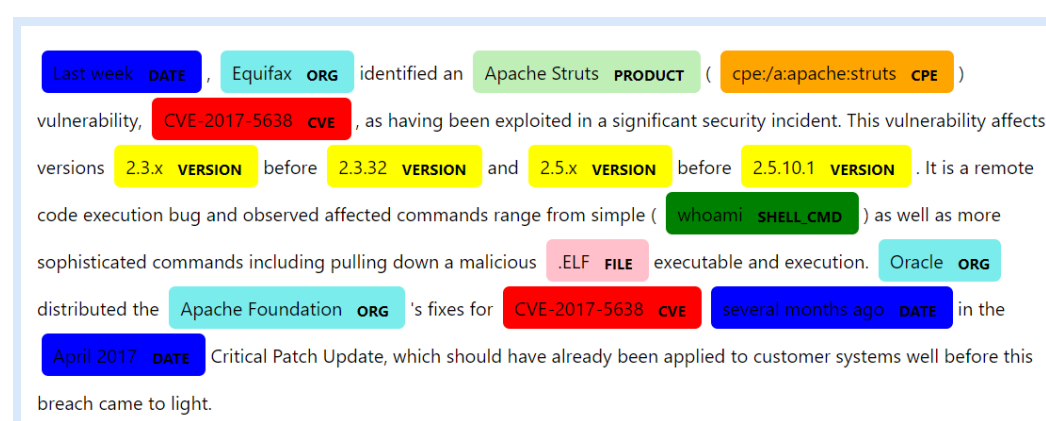


Figure 5. Identified entities.

**CPE Suggestion Engine.** In the previous section we outlined the process of extracting entities from unstructured text documents in an attempt to identify Cyber-Threat Intelligence. While this is an important task on its own, the extracted information is still largely unstructured and as the *Data Management and Sharing module* already contains large amounts of verified and structured CTI (Sections 3.1.3 and 4.3), we decided that it would be beneficial to create a mechanism that helps us combine them. By doing so, we would be able to add new pieces of information to the already existing CTI entries.

The most obvious entities that we could use to map our newly found data to the structured CTI are “CVE” and “CPE”. However, non-technical users do not tend to use these types of identifiers when they converse in the context of web forums, etc, so the likelihood of encountering them in significant enough numbers is low. Because of that, we had to resort to a hybrid solution, the *CPE Suggestion Engine*, which we will describe below.

Although CVE and CPE entities are unique, *Product* entities appear with high frequency on the relevant gathered texts. Since we had already built a MongoDB collection with *Organizations*, *Products* and their *CPEs*, we decided that we could use it to create a recommendation engine, which by utilising text retrieval methods, would suggest the most likely CPEs associated with a particular *Product* entity. Those suggestions get added to the final object that will be sent to the *Data Management and Sharing module*, where a security expert can evaluate the suggested CPEs to see if it actually matches an existing Event, and subsequently, perform the linking of the Objects when necessary.

The reason behind the use of suggestions instead of exact string matching on *Product* entities was mainly due to the fact that on a free-text setting, a user might abbreviate a part of the product, use only the common popular name of it (e.g., “*Struts*” instead of “*Apache Struts*”), or simply make a spelling mistake. To present accurate suggestions, we opted to approach the problem as a Query Engine would (*auto-suggestions/predictive search*), by performing *fuzzy text search*. To that end, the *CPE Suggestion Engine* uses n-grams, a common method for calculating text similarity. Initially, the n-grams for each product in the MongoDB collection get generated and indexed. Then, we perform a query for each discovered *Product* entity, and by using MongoDB’s Text Search Operator, we compare the similarity of the query’s n-grams to the indexed n-grams. Finally, MongoDB returns the results sorted by text match score. Tables 8 and 9 show that even with a partial name of the product, the *CPE Suggestion Engine* performs well, by returning the correct and relevant results at the top two spots.

**Table 8.** Possible Products/CPEs for query “Apache Struts”.

#	Product	CPE	Score
1	Apache Struts	cpe:/a:apache:struts	1160.75
2	Apache Struts Showcase App	cpe:/a:apache:struts2-showcase	1148.02
3	Apache Storm	cpe:/a:apache:storm	748.68
4	Apache Stats	cpe:/a:apache:stats:apache_stats	748.68
5	Apache LDAP Studio	cpe:/a:apache:apache_ldap_studio	734.84
6	Apache Directory Studio	cpe:/a:apache:apache_directory_studio	730.98
7	Apache Standard Taglibs	cpe:/a:apache:standard_taglibs	720.43
8	Apache Solr	cpe:/a:apache:solr	643.65
9	Apache Shindig	cpe:/a:apache:shindig	642.21
10	Apache Syncope	cpe:/a:apache:syncope	642.21

**Table 9.** Possible Products/CPEs for query “Struts”.

#	Product	CPE	Score
1	Apache Struts Showcase App	cpe:/a:apache:struts2-showcase	223.72
2	Apache Struts	cpe:/a:apache:struts	195.83
3	S2Struts	cpe:/a:theseasars_foundation:s2struts	170.00

**NP Chunking.** For the final part of our analysis, we used spaCy’s *Dependency Parser* to perform the task of Noun Phrase Chunking (NP Chunking) [128,129]. NP Chunking the subset of Text Chunking that deals with the task of recognizing non-overlapping text parts that consist of noun phrases (NPs). NP Chunking has several applications such as, resolving of word sense ambiguities, retrieving the data’s from the documents depending on the chunks rather than the words in Information Retrieval systems, and aligning of text in parallel corpora for machine translation.

While most of the CTI that we expect to discover can be effectively modeled to our Named Entity Recognizer presented above, some domain-specific concepts cannot be adequately defined as named entities. Such concepts include types of attacks and system vulnerabilities, exploit names, malware names, etc. We could add them to our Phrase Matcher as terminology lists, but due to the dynamic way that such concepts are described in non-technical texts, the effectiveness of the system would not be satisfactory.

After a thorough observation of our collected data, we discovered a common pattern, that these concepts are innately expressed as Noun Phrase chunks. For example, phrases such as “database injection vulnerability”, “brute-force attack” and “privilege escalation exploit” are all NPs that can be classified as Cyber-Threat Intelligence, and we would not be able to identify them with our pre-existing infrastructure.

To this end, as part of our CTI Extraction module, we have implemented an NP Chunker that detects all the NP chunks found in a document and groups them in an object called “HIGHLIGHTS”.

For instance, on the document presented in Figure 5, the HIGHLIGHTS would be the following:

- “Apache Struts vulnerability”,
- “remote code execution bug”, and
- “April 2017 Critical Patch Update”

This method can greatly assist the security experts to quickly identify whether a document contains actionable CTI or not.

#### 4.3. Data Management and Sharing

In Section 3.3, we have described the components of the Management and Sharing module of INTIME. In this section, we will focus the specifics of the Data Objectification, Event Management and MISP submodules and illustrate how these submodules are used within the CYBER-TRUST case study.

**Data Objectification.** Information stemming from the Data Acquisition and Data Analysis modules (Sections 3.1 and 3.2) needs to be transformed into appropriate objects before it can be stored into MISP. This process is termed *Data Objectification*.

To store CTI we utilise the vulnerability [130] and weakness [131] objects provided by MISP. Additionally, we have created custom-made objects to encapsulate information that is provided by our sources but is not modeled by the vulnerability and weakness objects. Thus, we have introduced extra-vulnerability and expdb-poc objects that extend vulnerability, and exploit-poc [132] objects respectively. Finally, in every object we store a source identifier in the credit attribute, to be able to distinguish the source of the stored CTI.

Data Objectification is performed by various custom-made Python scripts. These scripts depend on the source type and the resulting object. This submodule first parses the provided information and generates the appropriate objects in JSON format. The information consists of triplets of the type <field,value,comment>; the specific values of the triplets are populated from the parsing phase. The comment field is used to enrich the value information. For example, it can be used to declare if the source of a reference, is from the affected vendor, or from another vulnerability notes’ source.

**Event Management.** This submodule is responsible for the consolidation of the gathered information with the information already stored in MISP. Gathered and stored CTI tends to differ in variety, quality and timeliness. Moreover, analyzing CTI at different times, may provide different assessments. This happens due to the dynamics of the available information at the time of the analysis. A security analyst is interested in all available information and assessments. Thus, we gather all available CTI from our sources, we parse them one-by-one, to extract the CTI provided, and then, we merge with existing information and store it in a clustered manner.

To determine whether the CTI which arrived, is cataloged by the system or not, we query MISP. The result of the query can lead to two possible outcomes.



In the case that *the incoming CTI does not exist*, we generate a new event in the MISP instance. Next, we generate the required MISP objects (with regard to the specifications of each monitored source), from the constructed JSON structures of the Data Objectification submodule. Additionally, the generated objects' validity is checked both locally, through the PyMISP library's objects' definitions, and externally, through a PyMISP request of the MISP instance objects' definitions. Both definitions must be the same for this step to succeed, and they are expressed in the form of JSON files, in the PyMISP library's files and the MISP instance's files. Finally, we attach the generated objects to the event that was generated in the first step, on the MISP instance.

In contrast, if *the incoming CTI regards an existing event E* in MISP, we distinguish two cases. In the first case, *the incoming information is not already stored in the event E*. In this case, the incoming information is able to enrich the existing event *E*. Thus, we consider the MISP objects, created by the Data Objectification module and check the `credit` field of each object within the indexed event. If there is no match, the system proceeds to attach the generated MISP objects to the existing event *E*. In the second case, *the incoming information is stored in the event E*. In this case, the incoming information updates the existing event *E*. As with the previous case, we consider the corresponding MISP objects and again check the `credit` field of each object within the stored event *E*. If there is a match, the system proceeds to check the `modified` attribute of the matching object, which regards the modification date of the CTI encompassed. If the modification date of the newly parsed CTI is more recent than the previously stored one, the system deletes the stored object, and proceeds to attach the newly generated object, to the event *E*.

**MISP.** The CTI stored in MISP is available in both human and machine-readable formats and is accessible through a graphical interface and a REST API. The starting page of MISP provides a list through which a user can navigate amongst the stored events. These events are provided along with their metadata such as:

1. publication status (published or not),
2. organisation that produced the event,
3. organisation that owns the generated event
4. event's unique identifier,
5. MISP clusters and tags that may describe the event,
6. number of attributes included in the event,
7. email of the user that produced the event on the instance,
8. event's generation date,
9. event's information field,
10. event's distribution level.

Moreover, each entry of the events' list, is accompanied by a set of actions (publish/edit/delete/view event). Through the events' list page, a MISP user is provided with search capabilities, which regard the aforementioned metadata of the events, or values stored within the events' attributes (as presented in Figure 6). Finally, after each event creation/modification, MISP proceeds to recalculate the correlations, through the MISP Correlation Engine, since there is a possibility that the newly stored CTI may regard the same affected products, as other events. After this process, the event at hand points to all related events (as depicted in the top part of Figure 6).

When a user selects an event from the list, the MISP instance opens the event's view page, which provides all information that concern the selected event. Through this view, a user is not only able to view the event's contents, but is also provided with practical views about the event, such as the event's timeline (an overview of event's changes timeframe) and the event's correlation graph (that illustrates all related events). Finally, a MISP user is able to manage the event in a plethora of manners, such as editing the event's metadata, deleting it, adding objects and attributes, publishing it, merging attributes from another event. This functionality along with the provided search capabilities, allows security experts to further assess and evaluate CTI.

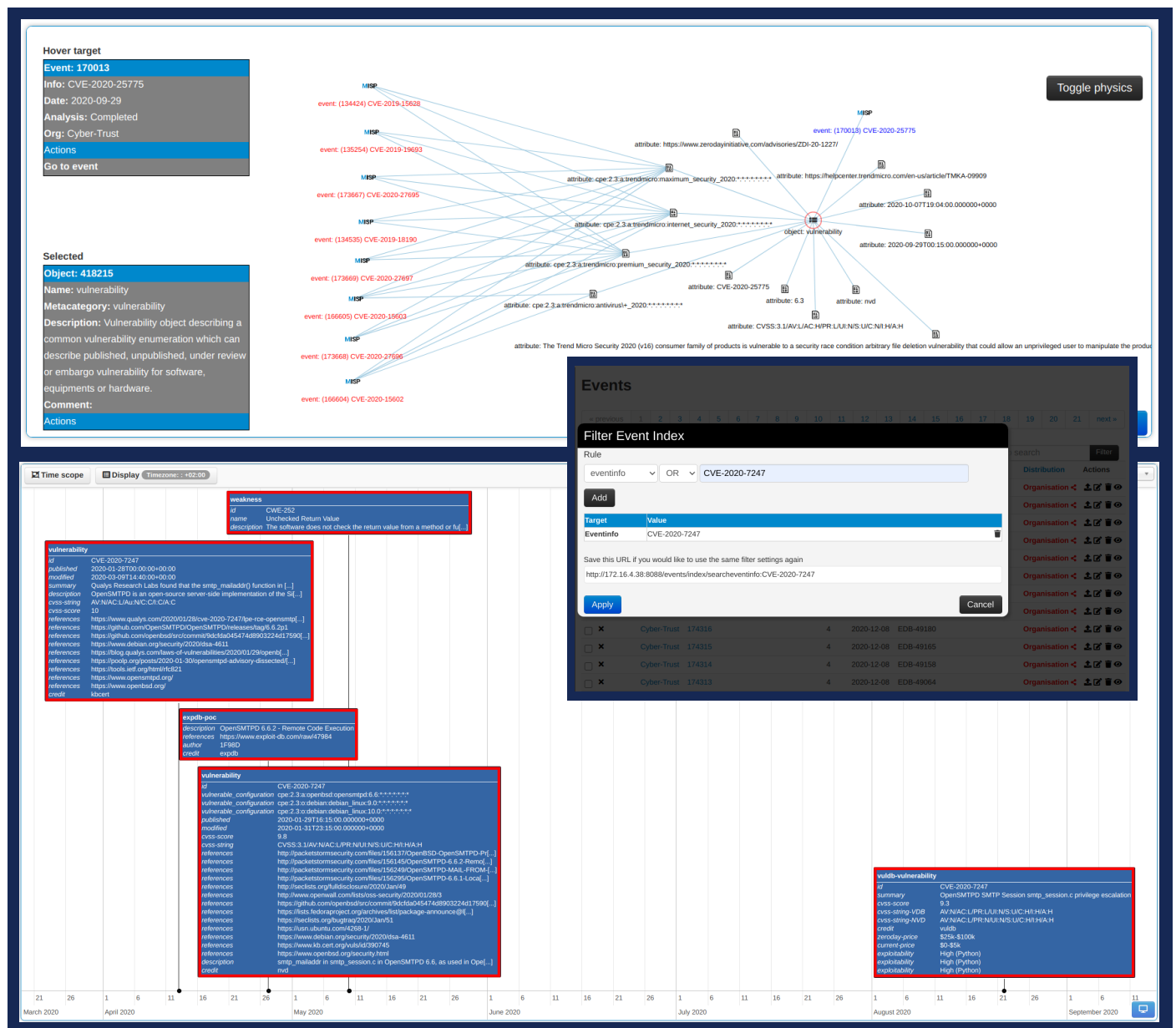


Figure 6. MISP graphical interface.

## 5. Experimental Evaluation

In this section, we provide several proof-of-concept evaluations aiming to showcase the performance and viability of our design; the presented experiments focus on the classification models of the *Data Acquisition* module and provide some insights and statistics on the overall architecture from the perspective of the *Data Management and Sharing* module. The detailed performance assessment of each individual module and comparison against existing solutions is out of the scope of this (architecturally oriented) manuscript and is already planned as future work.

### 5.1. Evaluation of the Topical Crawler's Classification Model

For the evaluation of the SMILE Classifier for the Topical Crawl, we created two models, each using a different approach for the page selection process. In this section, we are going to use a running example and show the reasoning behind the page selections along with some results on specified metrics. For the purposes of the evaluation, we monitored four different metrics for each crawl (as also shown in Table 10):

1. The total number of crawled pages.
2. The number of crawled pages that were considered relevant.
3. The number of crawled pages that were considered irrelevant.
4. The percentage of relevant pages or harvest rate.

**Table 10.** Page Classifier Results.

Metric	NFP Model	WFP Model
Total Pages	392,157	1,020,983
Relevant Pages	37,157	1,012,045
Irrelevant Pages	355,000	8938
Harvest Rate	9.475%	99.125%

**NFP Model.** For the first proposed model, we selected a small sample of relevant and irrelevant pages with the positive-negative ratio being exactly 1:1. We employed a *Narrow False Positives* approach, where pages that we considered relevant were pages that covered the subject of security in IoT. Pages that we considered irrelevant were pages that cover either security or IoT, but not both at the same time.

For the seed pages, we used the pages that we selected as relevant. Although this is not an ideal scenario, as the seeds need to have outlinks relevant to the topic for the crawl to work optimally, we came to this selection as the task of hand-picking another satisfactory set of relevant pages that would also have appropriate outlinks proved to be quite difficult.

As shown in Table 10, the harvest rate of the crawler is low, which indicates that the classification model is strict on what is considered relevant. Notice that this is a good result considering that the provided training set is relatively small. On further inspection, there were a few “false negatives” in the results, i.e. relevant pages that were considered irrelevant by the model, but this was expected from such a low number of input pages and such a strict policy on what is considered relevant and irrelevant. Also, since the crawler was only allowed to follow links of the pages that were selected as relevant, the small set and not ideal quality (not many outlinks relevant to the topic) of initial seeds led to a small number of total crawled pages in general, so we determined that although this configuration had an adequate classification performance, it required more and better quality seeds to be able to function optimally.

**WFP Model.** For the second proposed model, we experimented with a positive-negative ratio of 1:2. This time, the model follows a *Wide False Positives* policy, where pages that we considered relevant were pages that cover the subject of security in IoT, whereas the irrelevant pages consisted of some pages that cover security along with several random tech-related pages. In this case, for the seed pages, we used ACHE’s SeedFinder Tool with the query “iot vulnerabilities”.

On this crawl, the harvest rate was very high, since a significantly small number of pages was considered irrelevant. Because of the random URLs that we added to the negative pages, the model classified every page that was not similar to them as relevant. Even though the seeds for this crawl were better than the previous one, the final result is not optimal, as there are too many false positives, i.e. irrelevant pages that were considered relevant, which ultimately made the classification a failure.

After evaluating the proposed models, we concluded that the best results were obtained when using the URLs of the NFP model for the training, which incorporates a strict policy on irrelevant pages. With the strict policy, there were a few false negatives, but it proved to be much better than the alternative; having most of the crawled pages be false positives (*WFP model*). Also, the seeds generated by the SeedFinder Tool for the WFP model proved more efficient compared to the case where the positive URLs were used as seeds (*NFP model*). Consequently, the final decision was to use the training URLs of the NFP model, along with the seed URLs of the WFP model.

### 5.2. Twitter Classifier Comparison: CNN vs. Random Forest

Apart from the traditional Machine Learning algorithms mentioned in Section 4.1.2, several architectures of CNN models (layers, kernel size, filters, dropout, batch normalization, learning rate, activation function) were tested until we found the one that better fits our problem. On detailed examination of the results of the Random Forest classifier and the selected CNN model, we discovered that their performance is similar, across all metrics, however, the Random Forest classifier has the best performance in every setup that we tested (with or without the CVE tag). Each classification model's performance can be seen in Table 11.

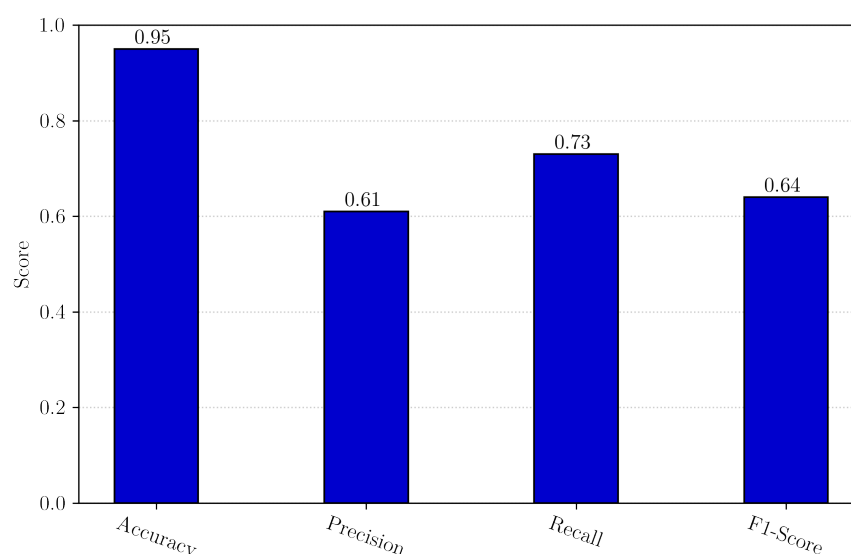
**Table 11.** Performance comparison of best Twitter Classifiers.

Metric	Classifier	Train CVE – Test CVE	Train CVE – Test No CVE	Train No CVE – Test CVE	Train No CVE – Test No CVE
Accuracy	Random Forest	<b>0.8737</b>	<b>0.8744</b>	<b>0.8751</b>	<b>0.8751</b>
	CNN	0.8542	0.8287	0.8426	0.8483
Precision	Random Forest	<b>0.8914</b>	<b>0.8799</b>	<b>0.8925</b>	<b>0.8925</b>
	CNN	0.8491	0.8440	0.8714	0.8595
Recall	Random Forest	0.8484	<b>0.8645</b>	<b>0.8503</b>	<b>0.8503</b>
	CNN	<b>0.8581</b>	0.8025	0.8002	0.8291
F1-score	Random Forest	<b>0.8694</b>	<b>0.8721</b>	<b>0.8709</b>	<b>0.8709</b>
	CNN	0.8536	0.8227	0.8343	0.8441

To verify the performance of our system in a real-world setting, we performed two distinct 30-min runs of a full monitoring cycle (Twitter Stream API + Classification) with different keywords as input for each one.

In the first run we used 400 keywords related to the domain of IoT security, along with various platform names. The monitoring system acquired 46,051 tweets, out of which, 45,825 were classified as not related and 226 as related, meaning that its harvest rate was 0.49%. Because of the small harvest rate, we performed a close examination of the classified tweets, and we observed that most of the non-related ones had completely deviated from our subject. For that reason, although the system was moderately successful at identifying them appropriately, we concluded that this configuration was not computationally or storage efficient and decided to not use it.

In the second run we used a smaller, very specific set of 11 keywords. This time the monitoring system acquired 1677 tweets; 1584 not related and 93 related, thus achieving a harvest rate of 5.54%. This configuration was very successful at classifying the tweets, achieving 95% accuracy. The detailed metrics of this run can be seen in Figure 7.

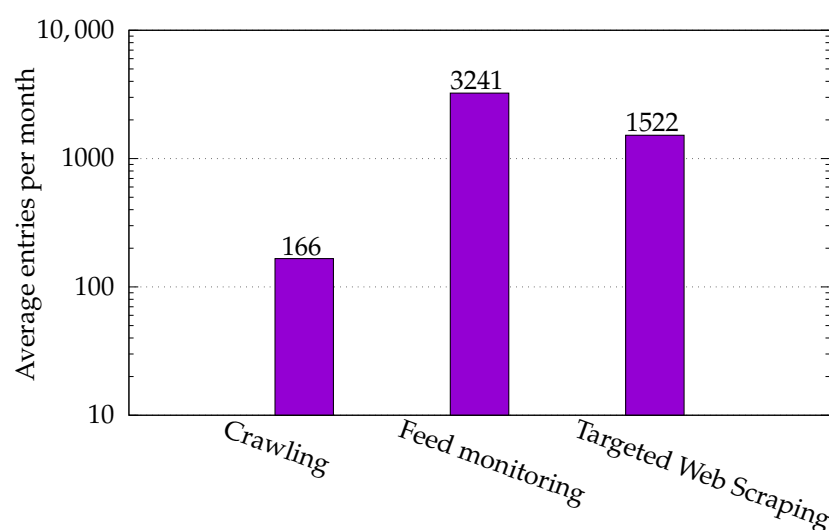


**Figure 7.** Evaluation metrics of live run of the Social Media Monitoring system with 11 keywords.

### 5.3. Data Management and Sharing Insights

During the course of the Data Management and Sharing module's utilisation, our system has stored approximately *181K entries* in INTIME. Table 12 presents the number of artifacts that were stored in MISP, for each Data Acquisition source. Notice that the total entries number is different than the sum of the artifacts of each source, due to the artifacts' merging procedures described in Sections 3.3 and 4.3.

On its first run, the Data Management and Sharing module stored all publicly available CTI of the monitored sources. Then, it synchronized with the sources on a daily basis, to receive any new entries or updates. To provide an overview of the CTI gathering, Figure 8 presents the gathering throughput of each source during the year 2020. Specifically, it presents the average MISP entries' creations and modifications per month, as occurred by each source. By taking under consideration both Table 12 and Figure 8, Feed monitoring sources (namely NVD and VulDB) hold the lion's share in vulnerabilities publishing.



**Figure 8.** The CTI gathering throughput of each source (average MISP entries/month).



**Table 12.** The number of artifacts stored in INTIME.

Source	Number of Artifacts
Crawling	~4000
Feed monitoring	~157,000
Targeted Web Scraping	~115,000
<b>Total Entries</b>	~181,000

## 6. Conclusions and Outlook

We have presented INTIME, the first solution in the literature to provide an end-to-end CTI management platform that is able to support the collection, analysis, leveraging and sharing of cyber-threat intelligence via an integrated, extensible framework. We presented the architectural solutions behind the proposed system, discussed the individual module technologies and provided details on the module orchestration.

We also described several novel services that extend the current state-of-the-art in various ways and include:

- easy-to-deploy data harvesting from the clear, social, deep, and dark web; this approach allows security analysts to define and deploy pre-trained focused crawlers, perform site acquisition via in-depth crawlers, and define targeted social media monitors via an easy-to-use graphical user interface that masks away all relevant technicalities. While specialized CTI crawlers are available in the literature, this is the first system that offers so wide functionality with respect to CTI collection without requiring specialized user intervention.
- automated content ranking according to its potential usefulness with respect to CTI; this approach targets to identify the most promising crawler content and is the first in the literature to view the crawling task as a two-stage process, where a crude classification is initially used to prune the crawl frontier, while a more refined approach based on the collected content is used to decide on its relevance to the task.
- automated CTI identification and extraction via state-of-the-art natural language understanding processes; this is the first work in the literature to go beyond regular expression matching (typically used to identify simple objects such as IPs) used in CTI extraction and defines new entities (such as commands, CVEs, and bitcoin values) that are interlinked and correlated according to context via dependency parsing.
- leveraging of the identified security items to actionable cyber-threat intelligence using automated entity disambiguation, and machine-assisted assisted linkage and correlation; this is the first work in the literature to augment CTI with proposed related identifiers such as CPE in order to assist correlation of previously uncorrelated CTI objects.
- CTI management and sharing via open standards and intuitive tools; this is facilitated by adopting research and industry standards such as STIX/TAXII and by resorting to standardized CTI visualization options such as timelines and correlation graphs.

Finally, we advocated INTIME's appropriateness for generic security-related reconnaissance tasks, instantiated it for the IoT security domain, and showcased its usefulness within the scope of the CYBER-TRUST project and provided evaluation details for the various INTIME components that were developed and operational statistics from the pilot deployment of the platform.

Interesting directions for future research include (i) tighter integration with other security-related software such as intrusion detection/prevention systems (IDS/IPS), (ii) integrating cloud capabilities to provide elasticity on the used resources, (iii) introducing a more sophisticated, zero-administration CTI linking and correlation engine, (iv) performing a large-scale operational evaluation of the platform, including user studies to improve usability and document additional needs.

**Author Contributions:** Conceptualization, P.K., T.C., S.S. and C.T.; Methodology, P.K., T.C., S.A., S.S. and C.T.; Software, P.K., T.C. and S.A.; Validation, P.K., T.C. and S.A.; Writing—Original draft, P.K., T.C., S.A., S.S. and C.T.; Supervision, S.S. and C.T. All authors have read and agreed to the published version of the manuscript.

**Funding:** This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreements no. 786698 and 833673. The work reflects only the authors’ view and the Agency is not responsible for any use that may be made of the information it contains.

**Data Availability Statement:** Some of the experiments described in the manuscript, use the Stack Exchange Data Dump [121] for training the Word2Vec models.

**Acknowledgments:** We would like to thank our colleagues in Mathema S.R.L. for implementing part of the UI of the platform, and especially Emanuele Bellini and Simone Naldini for the technical and Alessandro Bellini for the financial support during the research visit of Paris Koloveas at the company.

**Conflicts of Interest:** The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

## References

1. Cvitić, I.; Peraković, D.; Periša, M.; Botica, M. Novel approach for detection of IoT generated DDoS traffic. *Wirel. Netw.* **2019**, *1*–14. [CrossRef]
2. Cvitić, I.; Peraković, D.; Periša, M.; Husnjak, S. An overview of distributed denial of service traffic detection approaches. *Promet-Traffic Transp.* **2019**, *31*, 453–464. [CrossRef]
3. Bhushan, K.; Gupta, B.B. Distributed denial of service (DDoS) attack mitigation in software defined network (SDN)-based cloud computing environment. *J. Ambient Intell. Humaniz. Comput.* **2019**, *10*, 1985–1997. [CrossRef]
4. Osanaiye, O.; Choo, K.K.R.; Dlodlo, M. Distributed denial of service (DDoS) resilience in cloud: Review and conceptual cloud DDoS mitigation framework. *J. Netw. Comput. Appl.* **2016**, *67*, 147–165. [CrossRef]
5. Wagner, C.; Dulaunoy, A.; Wager, 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, Vienna, Austria, 24–28 October 2016; ACM: New York, NY, USA, pp. 49–56.
6. MISP. Available online: <https://www.misp-project.org/> (accessed on 20 February 2021).
7. Cyber-Trust EU. Available online: <http://cyber-trust.eu/> (accessed on 20 February 2021).
8. Najork, M. Web Crawler Architecture. In *Encyclopedia of Database Systems*; Springer: New York, NY, USA, 2009; pp. 3462–3465. [CrossRef]
9. Hsieh, J.M.; Gribble, S.D.; Levy, H.M. The Architecture and Implementation of an Extensible Web Crawler. In Proceedings of the 7th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2010, San Jose, CA, USA, 28–30 April 2010; pp. 329–344.
10. Harth, A.; Umbrich, J.; Decker, S. MultiCrawler: A Pipelined Architecture for Crawling and Indexing Semantic Web Data. In Proceedings of the Semantic Web—ISWC 2006, 5th International Semantic Web Conference, ISWC 2006, Athens, GA, USA, 5–9 November 2006; pp. 258–271. [CrossRef]
11. Ahmadi-Abkenari, F.; Selamat, A. An architecture for a focused trend parallel Web crawler with the application of clickstream analysis. *Inf. Sci.* **2012**, *184*, 266–281. [CrossRef]
12. Quoc, D.L.; Fetzer, C.; Felber, P.; Rivi re, E.; Schiavoni, V.; Sutra, P. UniCrawl: A Practical Geographically Distributed Web Crawler. In Proceedings of the 8th IEEE International Conference on Cloud Computing, CLOUD 2015, New York, NY, USA, 27 June–2 July 2015; pp. 389–396. [CrossRef]
13. Vikas, O.; Chiluka, N.J.; Ray, P.K.; Meena, G.; Meshram, A.K.; Gupta, A.; Sisodia, A. WebMiner—Anatomy of Super Peer Based Incremental Topic-Specific Web Crawler. In Proceedings of the Sixth International Conference on Networking (ICN 2007), Sainte-Luce, Martinique, France, 22–28 April 2007; p. 32. [CrossRef]
14. Bamba, B.; Liu, L.; Caverlee, J.; Padliya, V.; Srivatsa, M.; Bansal, T.; Palekar, M.; Patrao, J.; Li, S.; Singh, A. DSphere: A Source-Centric Approach to Crawling, Indexing and Searching the World Wide Web. In Proceedings of the 23rd International Conference on Data Engineering, ICDE 2007, Istanbul, Turkey, 15–20 April 2007; pp. 1515–1516. [CrossRef]
15. Stoica, I.; Morris, R.T.; Liben-Nowell, D.; Karger, D.R.; Kaashoek, M.F.; Dabek, F.; Balakrishnan, H. Chord: A scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Trans. Netw.* **2003**, *11*, 17–32. [CrossRef]
16. Shkapenyuk, V.; Suel, T. Design and Implementation of a High-Performance Distributed Web Crawler. In Proceedings of the 18th International Conference on Data Engineering, San Jose, CA, USA, 26 February–1 March 2002; pp. 357–368. [CrossRef]
17. Gupta, K.; Mittal, V.; Bishnoi, B.; Maheshwari, S.; Patel, D. AcT: Accuracy-aware crawling techniques for cloud-crawler. *World Wide Web* **2016**, *19*, 69–88. [CrossRef]

18. Li, Y.; Zhao, L.; Liu, X.; Zhang, P. A Security Framework for Cloud-Based Web Crawling System. In Proceedings of the 11th Web Information System and Application Conference, WISA 2014, Tianjin, China, 12–14 September 2014; pp. 101–104. [\[CrossRef\]](#)
19. Ntoulas, A.; Cho, J.; Olston, C. What's new on the web?: The evolution of the web from a search engine perspective. In Proceedings of the 13th international conference on World Wide Web, WWW 2004, New York, NY, USA, 17–20 May 2004; pp. 1–12. [\[CrossRef\]](#)
20. McCurley, K.S. Incremental Crawling. In *Encyclopedia of Database Systems*; Springer: New York, NY, USA, 2009; pp. 1417–1421. [\[CrossRef\]](#)
21. Sizov, S.; Graupmann, J.; Theobald, M. From Focused Crawling to Expert Information: An Application Framework for Web Exploration and Portal Generation. In Proceedings of the 29th International Conference on Very Large Data Bases, VLDB 2003, Berlin, Germany, 9–12 September 2003; pp. 1105–1108.
22. Chakrabarti, S.; van den Berg, M.; Dom, B. Focused Crawling: A New Approach to Topic-Specific Web Resource Discovery. *Comput. Netw.* **1999**, *31*, 1623–1640. [\[CrossRef\]](#)
23. Zhang, Z.; Nasraoui, O.; van Zwol, R. Exploiting Tags and Social Profiles to Improve Focused Crawling. In Proceedings of the 2009 IEEE/WIC/ACM International Conference on Web Intelligence, WI 2009, Milan, Italy, 15–18 September 2009; pp. 136–139. [\[CrossRef\]](#)
24. Ester, M.; Gro, M.; Kriegel, H.P. Focused Web Crawling: A Generic Framework for Specifying the User Interest and for Adaptive Crawling Strategies. In Proceedings of the 27th International Conference on Very Large Data Bases, Roma, Italy, 11–14 September 2001; Morgan Kaufmann Publishers: Burlington, MA, USA, 2001.
25. Chakrabarti, S.; Punera, K.; Subramanyam, M. Accelerated focused crawling through online relevance feedback. In Proceedings of the Eleventh International World Wide Web Conference, WWW 2002, Honolulu, HI, USA, 7–11 May 2002; pp. 148–159. [\[CrossRef\]](#)
26. Gaur, R.; Sharma, D.K. Focused crawling with ontology using semi-automatic tagging for relevancy. In Proceedings of the Seventh International Conference on Contemporary Computing, IC3 2014, Noida, India, 7–9 August 2014; pp. 501–506. [\[CrossRef\]](#)
27. Pham, K.; Santos, A.S.R.; Freire, J. Learning to Discover Domain-Specific Web Content. In Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, WSDM 2018, Marina Del Rey, CA, USA, 5–9 February 2018; pp. 432–440. [\[CrossRef\]](#)
28. Singh, M.P. *The Practical Handbook of Internet Computing*; CRC Press, Inc.: Boca Raton, FL, USA, 2004.
29. Jiang, J.; Yu, N.; Lin, C. FoCUS: Learning to crawl web forums. In Proceedings of the 21st World Wide Web Conference, WWW 2012, Lyon, France, 16–20 April 2012; Companion Volume, pp. 33–42. [\[CrossRef\]](#)
30. Sachan, A.; Lim, W.; Thing, V.L.L. A Generalized Links and Text Properties Based Forum Crawler. In Proceedings of the 2012 IEEE/WIC/ACM International Conferences on Web Intelligence, WI 2012, Macau, China, 4–7 December 2012; pp. 113–120. [\[CrossRef\]](#)
31. Yang, J.; Cai, R.; Wang, C.; Huang, H.; Zhang, L.; Ma, W. Incorporating site-level knowledge for incremental crawling of web forums: A list-wise strategy. In Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, 28 June–1 July 2009; pp. 1375–1384. [\[CrossRef\]](#)
32. Wang, Y.; Yang, J.; Lai, W.; Cai, R.; Zhang, L.; Ma, W. Exploring traversal strategy for web forum crawling. In Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2008, Singapore, 20–24 July 2008; pp. 459–466. [\[CrossRef\]](#)
33. Cai, R.; Yang, J.; Lai, W.; Wang, Y.; Zhang, L. iRobot: An intelligent crawler for web forums. In Proceedings of the 17th International Conference on World Wide Web, WWW 2008, Beijing, China, 21–25 April 2008; pp. 447–456. [\[CrossRef\]](#)
34. Guo, Y.; Li, K.; Zhang, K.; Zhang, G. Board Forum Crawling: A Web Crawling Method for Web Forum. In Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2006), Hong Kong, China, 18–22 December 2006; pp. 745–748. [\[CrossRef\]](#)
35. Hurst, M.; Maykov, A. Social Streams Blog Crawler. In Proceedings of the 25th International Conference on Data Engineering, ICDE 2009, Shanghai, China, 29 March–2 April 2009; pp. 1615–1618. [\[CrossRef\]](#)
36. Agarwal, S.; Sureka, A. A Topical Crawler for Uncovering Hidden Communities of Extremist Micro-Bloggers on Tumblr. In Proceedings of the the 5th Workshop on Making Sense of Microposts Co-Located with the 24th International World Wide Web Conference (WWW 2015), Florence, Italy, 18 May 2015; pp. 26–27.
37. Chau, D.H.; Pandit, S.; Wang, S.; Faloutsos, C. Parallel crawling for online social networks. In Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, AB, Canada, 8–12 May 2007; pp. 1283–1284. [\[CrossRef\]](#)
38. Zhang, Z.; Nasraoui, O. Profile-Based Focused Crawler for Social Media-Sharing Websites. In Proceedings of the 20th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2008), Dayton, OH, USA, 3–5 November 2008; Volume 1, pp. 317–324. [\[CrossRef\]](#)
39. Buccafurri, F.; Lax, G.; Nocera, A.; Ursino, D. Crawling Social Internetworking Systems. In Proceedings of the International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2012, Istanbul, Turkey, 26–29 August 2012; pp. 506–510. [\[CrossRef\]](#)
40. Khan, A.; Sharma, D.K. Self-Adaptive Ontology based Focused Crawler for Social Bookmarking Sites. *IJIRR* **2017**, *7*, 51–67. [\[CrossRef\]](#)

41. Ferreira, R.; Lima, R.; Melo, J.; Costa, E.; de Freitas, F.L.G.; Luna, H.P.L. RetriBlog: A framework for creating blog crawlers. In Proceedings of the ACM Symposium on Applied Computing, SAC 2012, Trento, Italy, 26–30 March 2012; pp. 696–701. [\[CrossRef\]](#)
42. Tor Project. Available online: <https://www.torproject.org/> (accessed on 20 February 2021).
43. I2P Anonymous Network. Available online: <https://geti2p.net/en/> (accessed on 20 February 2021).
44. Valkanas, G.; Ntoulas, A.; Gunopulos, D. Rank-Aware Crawling of Hidden Web sites. In Proceedings of the 14th International Workshop on the Web and Databases 2011, WebDB 2011, Athens, Greece, 12 June 2011.
45. Wang, Y.; Lu, J.; Chen, J.; Li, Y. Crawling ranked deep Web data sources. *World Wide Web* **2017**, *20*, 89–110. [\[CrossRef\]](#)
46. Zhao, F.; Zhou, J.; Nie, C.; Huang, H.; Jin, H. SmartCrawler: A Two-Stage Crawler for Efficiently Harvesting Deep-Web Interfaces. *IEEE Trans. Serv. Comput.* **2016**, *9*, 608–620. [\[CrossRef\]](#)
47. Zheng, Q.; Wu, Z.; Cheng, X.; Jiang, L.; Liu, J. Learning to crawl deep web. *Inf. Syst.* **2013**, *38*, 801–819. [\[CrossRef\]](#)
48. Jiang, L.; Wu, Z.; Feng, Q.; Liu, J.; Zheng, Q. Efficient Deep Web Crawling Using Reinforcement Learning. In Proceedings of the Advances in Knowledge Discovery and Data Mining, 14th Pacific-Asia Conference, PAKDD 2010, Hyderabad, India, 21–24 June 2010; Part I, pp. 428–439. [\[CrossRef\]](#)
49. Madhavan, J.; Ko, D.; Kot, L.; Ganapathy, V.; Rasmussen, A.; Halevy, A.Y. Google’s Deep Web crawl. *PVLDB* **2008**, *1*, 1241–1252. [\[CrossRef\]](#)
50. Shaila, S.G.; Vadivel, A. Architecture specification of rule-based deep web crawler with indexer. *Int. J. Knowl. Web Intell.* **2013**, *4*, 166–186. [\[CrossRef\]](#)
51. Zhao, J.; Wang, P. Nautilus: A Generic Framework for Crawling Deep Web. In Proceedings of the Data and Knowledge Engineering—Third International Conference, ICDKE 2012, Wuyishan, China, 21–23 November 2012; pp. 141–151. [\[CrossRef\]](#)
52. Li, Y.; Wang, Y.; Tian, E. A New Architecture of an Intelligent Agent-Based Crawler for Domain-Specific Deep Web Databases. In Proceedings of the 2012 IEEE/WIC/ACM International Conferences on Web Intelligence, WI 2012, Macau, China, 4–7 December 2012; pp. 656–663. [\[CrossRef\]](#)
53. Furche, T.; Gottlob, G.; Grasso, G.; Schallhart, C.; Sellers, A.J. OXPath: A language for scalable data extraction, automation, and crawling on the deep web. *Vldb J.* **2013**, *22*, 47–72. [\[CrossRef\]](#)
54. Lu, J.; Wang, Y.; Liang, J.; Chen, J.; Liu, J. An Approach to Deep Web Crawling by Sampling. In Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence, WI 2008, Sydney, NSW, Australia, 9–12 December 2008; pp. 718–724. [\[CrossRef\]](#)
55. Liu, J.; Wu, Z.; Jiang, L.; Zheng, Q.; Liu, X. Crawling Deep Web Content through Query Forms. In Proceedings of the Fifth International Conference on Web Information Systems and Technologies, WEBIST 2009, Lisbon, Portugal, 23–26 March 2009; pp. 634–642.
56. Li, Y.; Wang, Y.; Du, J. E-FFC: An enhanced form-focused crawler for domain-specific deep web databases. *J. Intell. Inf. Syst.* **2013**, *40*, 159–184. [\[CrossRef\]](#)
57. He, Y.; Xin, D.; Ganti, V.; Rajaraman, S.; Shah, N. Crawling deep web entity pages. In Proceedings of the Sixth ACM International Conference on Web Search and Data Mining, WSDM 2013, Rome, Italy, 4–8 February 2013; pp. 355–364. [\[CrossRef\]](#)
58. Sapienza, A.; Bessi, A.; Damodaran, S.; Shakarian, P.; Lerman, K.; Ferrara, E. Early Warnings of Cyber Threats in Online Discussions. In Proceedings of the 2017 IEEE International Conference on Data Mining Workshops, ICDM Workshops 2017, New Orleans, LA, USA, 18–21 November 2017; Gottumukkala, R., Ning, X., Dong, G., Raghavan, V., Aluru, S., Karypis, G., Miele, L., Wu, X., Eds.; IEEE Computer Society: Piscataway, NJ, USA, 2017; pp. 667–674. [\[CrossRef\]](#)
59. Mittal, S.; Das, P.K.; Mulwad, V.; Joshi, A.; Finin, T. CyberTwitter: Using Twitter to generate alerts for cybersecurity threats and vulnerabilities. In Proceedings of the 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2016, San Francisco, CA, USA, 18–21 August 2016; Kumar, R., Caverlee, J., Tong, H., Eds.; IEEE Computer Society: Piscataway, NJ, USA, 2016; pp. 860–867. [\[CrossRef\]](#)
60. Alves, F.; Bettini, A.; Ferreira, P.M.; Bessani, A. Processing tweets for cybersecurity threat awareness. *Inf. Syst.* **2021**, *95*, 101586. [\[CrossRef\]](#)
61. Syed, R.; Rahafrooz, M.; Keisler, J.M. What it takes to get retweeted: An analysis of software vulnerability messages. *Comput. Hum. Behav.* **2018**, *80*, 207–215. [\[CrossRef\]](#)
62. Sabottke, C.; Suciu, O.; Dumitras, T. Vulnerability Disclosure in the Age of Social Media: Exploiting Twitter for Predicting Real-World Exploits. In Proceedings of the 24th USENIX Security Symposium, USENIX Security 15, Washington, DC, USA, 12–14 August 2015; Jung, J., Holz, T., Eds.; USENIX Association: Berkeley, CA, USA, 2015; pp. 1041–1056.
63. Sauerwein, C.; Sillaber, C.; Huber, M.M.; Mussmann, A.; Breu, R. The Tweet Advantage: An Empirical Analysis of 0-Day Vulnerability Information Shared on Twitter. In Proceedings of the ICT Systems Security and Privacy Protection—33rd IFIP TC 11 International Conference, SEC 2018, Held at the 24th IFIP World Computer Congress, WCC 2018, Poznan, Poland, 18–20 September 2018; IFIP Advances in Information and Communication Technology; Janczewski, L.J., Kutylowski, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2018; Volume 529, pp. 201–215. [\[CrossRef\]](#)
64. CVE Dictionary, MITRE. Available online: <https://cve.mitre.org/> (accessed on 20 February 2021).
65. Le, B.D.; Wang, G.; Nasim, M.; Babar, M.A. Gathering Cyber Threat Intelligence from Twitter Using Novelty Classification. *arXiv* **2019**, arXiv:1907.01755.



66. Husari, G.; Al-Shaer, E.; Ahmed, M.; Chu, B.; Niu, X. TTPDrill: Automatic and Accurate Extraction of Threat Actions from Unstructured Text of CTI Sources. In Proceedings of the 33rd Annual Computer Security Applications Conference, ACSAC 2017, Orlando, FL, USA, 4–8 December 2017; ACM: New York, NY, USA, 2017; pp. 103–115. [CrossRef]
67. CAPEC, MITRE. Available online: <https://capec.mitre.org/> (accessed on 20 February 2021).
68. ATTA&CK, MITRE. Available online: <https://attack.mitre.org/> (accessed on 20 February 2021).
69. Liao, X.; Yuan, K.; Wang, X.; Li, Z.; Xing, L.; Beyah, R. Acing the IOC Game: Toward Automatic Discovery and Analysis of Open-Source Cyber Threat Intelligence. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, CCS '16, Vienna, Austria, 24–28 October 2016; ACM: New York, NY, USA, 2016; pp. 755–766. [CrossRef]
70. OpenIOC. Available online: <https://www.fireeye.com/services/freeware.html> (accessed on 20 February 2021).
71. GOSINT. Available online: <https://gosint.readthedocs.io/en/latest/> (accessed on 20 February 2021).
72. YETI—Your Everyday Threat Intelligence. Available online: <https://yeti-platform.github.io/> (accessed on 20 February 2021).
73. OpenTAXII. Available online: <https://opentaxii.readthedocs.io/en/stable/> (accessed on 20 February 2021).
74. CIF—Collective Intelligence Framework. Available online: <https://csirtgadgets.com/collective-intelligence-framework> (accessed on 20 February 2021).
75. ZeroFox. Available online: <https://www.zerofox.com/> (accessed on 20 February 2021).
76. CTAC, Wapack Labs. Available online: <https://www.wapacklabs.com/ctac> (accessed on 20 February 2021).
77. SearchLight, Digital Shadows. Available online: <https://www.digitalsadows.com/searchlight/> (accessed on 20 February 2021).
78. Intel 471. Available online: <https://intel471.com/> (accessed on 20 February 2021).
79. Flashpoint Intelligence Platform. Available online: <https://www.flashpoint-intel.com/platform/> (accessed on 20 February 2021).
80. BitSight Security Ratings. Available online: <https://www.bitsight.com/security-ratings> (accessed on 20 February 2021).
81. BreachAlert, SKURIO. Available online: <https://skurio.com/solutions/breach-alert/> (accessed on 20 February 2021).
82. F5 Labs. Available online: <https://www.f5.com/labs> (accessed on 20 February 2021).
83. Helix Security Platform, FireEye. Available online: <https://www.fireeye.com/products/helix.html> (accessed on 20 February 2021).
84. Recorded Future. Available online: <https://www.recordedfuture.com/> (accessed on 20 February 2021).
85. Cyjax. Available online: <https://www.cyjax.com/cyber-threat-services/> (accessed on 20 February 2021).
86. EclecticIQ. Available online: <https://www.eclecticiq.com/platform> (accessed on 20 February 2021).
87. Cyber Advisor, SurfWatch Labs. Available online: <https://www.surfwatchlabs.com/threat-intelligence-products/cyber-advisor> (accessed on 20 February 2021).
88. Infoblox. BloxOne Threat Defense Advanced: Strengthen and Optimize Your Security Posture from the Foundation. 2020. Available online: <https://www.infoblox.com/wp-content/uploads/infoblox-datasheet-bloxone-threat-defense-advanced.pdf> (accessed on 30 October 2020).
89. ThreatStream, Anomali. Available online: <https://www.anomali.com/products/threatstream> (accessed on 20 February 2021).
90. ThreatQ, ThreatQuotient. Available online: <https://www.threatq.com/> (accessed on 20 February 2021).
91. Soltra, Celerium. Available online: <https://www.celerium.com/automate> (accessed on 20 February 2021).
92. ThreatConnect. Available online: <https://threatconnect.com/> (accessed on 20 February 2021).
93. VDMR, Qualys. Available online: <https://www.qualys.com/apps/vulnerability-management-detection-response/> (accessed on 20 February 2021).
94. The MANTIS Cyber Threat Intelligence Management Framework, SIEMENS. Available online: <https://django-mantis.readthedocs.io/en/latest/readme.html> (accessed on 20 February 2021).
95. BrightCloud, Webroot. Available online: <https://www.brightcloud.com/> (accessed on 20 February 2021).
96. Koloveas, P.; Chantzios, T.; Tryfonopoulos, C.; Skiadopoulos, S. A crawler architecture for harvesting the clear, social, and dark web for IoT-related cyber-threat intelligence. In Proceedings of the 2019 IEEE World Congress on Services (SERVICES), Milan, Italy, 8–13 July 2019; IEEE: Piscataway, NJ, USA, 2019; Volume 2642, pp. 3–8.
97. CERT Vulninterability Notes Database, Carnegie Mellon University. Available online: <https://www.kb.cert.org/vuls/bypublished/desc/> (accessed on 20 February 2021).
98. Exploit Database—ExploitDB. Available online: <https://www.exploit-db.com/> (accessed on 20 February 2021).
99. Ache Crawler, GitHub. Available online: <https://github.com/ViDA-NYU/ache> (accessed on 20 February 2021).
100. Li, H. Smile. 2014. Available online: <https://haifengl.github.io> (accessed on 20 February 2021).
101. Vieira, K.; Barbosa, L.; da Silva, A.S.; Freire, J.; Moura, E. Finding seeds to bootstrap focused crawlers. *World Wide Web* **2016**, *19*, 449–474. [CrossRef]
102. Chantzios, T.; Koloveas, P.; Skiadopoulos, S.; Kolokotronis, N.; Tryfonopoulos, C.; Bilali, V.; Kavallieros, D. The Quest for the Appropriate Cyber-threat Intelligence Sharing Platform. In Proceedings of the 8th International Conference on Data Science, Technology and Applications, DATA 2019, Prague, Czech Republic, 26–28 July 2019; Hammoudi, S., Quix, C., Bernardino, J., Eds.; SciTePress: Setúbal, Portugal, 2019; pp. 369–376. [CrossRef]
103. de Melo e Silva, A.; Gondim, J.J.C.; de Oliveira Albuquerque, R.; García-Villalba, L.J. A Methodology to Evaluate Standards and Platforms within Cyber Threat Intelligence. *Future Internet* **2020**, *12*, 108. [CrossRef]
104. Flask. Available online: <https://flask.palletsprojects.com/en/1.1.x/> (accessed on 20 February 2021).
105. Swagger. Available online: <https://swagger.io/> (accessed on 20 February 2021).
106. MongoDB. Available online: <https://www.mongodb.com/> (accessed on 20 February 2021).

107. Řehůřek, R.; Sojka, P. Software Framework for Topic Modelling with Large Corpora. In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, Valletta, Malta, 22 May 2010; ELRA: Valletta, Malta, 2010; pp. 45–50.
108. Honnibal, M.; Montani, I.; Van Landeghem, S.; Boyd, A. spaCy: Industrial-strength Natural Language Processing in Python. *zenodo* **2020**. [CrossRef]
109. MySQL. Available online: <https://www.mysql.com/> (accessed on 20 February 2021).
110. CakePHP. Available online: <https://cakephp.org/> (accessed on 20 February 2021).
111. PyMISP, GitHub. Available online: <https://github.com/MISP/PyMISP> (accessed on 20 February 2021).
112. Twitter API. Available online: <https://developer.twitter.com/en/docs/twitter-api> (accessed on 20 February 2021).
113. NVD, NIST. Available online: <https://nvd.nist.gov/> (accessed on 20 February 2021).
114. JVN iPedia. Available online: <https://jvndb.jvn.jp/en/> (accessed on 20 February 2021).
115. NVD Data Feeds, NIST. Available online: <https://nvd.nist.gov/vuln/data-feeds> (accessed on 20 February 2021).
116. JVN iPedia Feed. Available online: <https://jvndb.jvn.jp/en/feed/> (accessed on 20 February 2021).
117. Selenium. Available online: <https://www.selenium.dev/> (accessed on 20 February 2021).
118. Vulnerability Database – VulDB. Available online: <https://vuldb.com/> (accessed on 20 February 2021).
119. ODay.today Exploit Database. Available online: <https://oday.today/> (accessed on 20 February 2021).
120. Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.; Dean, J. Distributed Representations of Words and Phrases and Their Compositionality. In Proceedings of the 26th International Conference on Neural Information Processing Systems—Volume 2, NIPS’13, Lake Tahoe, NV, USA, 5–8 December 2013; Curran Associates Inc.: Red Hook, NY, USA, 2013; pp. 3111–3119.
121. Stack Exchange Data Dump, Archive.org. Available online: <https://archive.org/details/stackexchange> (accessed on 20 February 2021).
122. Internet of Things, Stack Exchange. Available online: <https://iot.stackexchange.com/> (accessed on 20 February 2021).
123. Information Security, Stack Exchange. Available online: <https://security.stackexchange.com/> (accessed on 20 February 2021).
124. Arduino, Stack Exchange. Available online: <https://arduino.stackexchange.com/> (accessed on 20 February 2021).
125. Raspberry Pi, Stack Exchange. Available online: <https://raspberrypi.stackexchange.com/> (accessed on 20 February 2021).
126. Bird, S.; Klein, E.; Loper, E. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*; O’Reilly Media, Inc.: Sebastopol, CA, USA, 2009.
127. Biega, J.A.; Gummadi, K.P.; Mele, I.; Milchevski, D.; Tryfonopoulos, C.; Weikum, G. R-Susceptibility: An IR-Centric Approach to Assessing Privacy Risks for Users in Online Communities. In Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval, Pisa, Italy, 17–21 July 2016; ACM: New York, NY, USA, 2016; SIGIR ’16, pp. 365–374. [CrossRef]
128. Sang, E.F.; Buchholz, S. Introduction to the CoNLL-2000 shared task: Chunking. *arXiv* **2000**, arXiv:abs/cs/0009008.
129. Veenstra, J.; Buchholz, S. Fast NP chunking using memory-based learning techniques. In Proceedings of the 8th Belgian—Dutch conference on machine learning, (BENELEARN’98), Wageningen, The Netherlands, 1998; ATO-DLO; pp. 71–78.
130. MISP Objects—Vulnerability. Available online: [https://www.misp-project.org/objects.html#\\_vulnerability](https://www.misp-project.org/objects.html#_vulnerability) (accessed on 20 February 2021).
131. MISP Objects—Weakness. Available online: [https://www.misp-project.org/objects.html#\\_weakness](https://www.misp-project.org/objects.html#_weakness) (accessed on 20 February 2021).
132. MISP Objects—Exploit-poc. Available online: [https://www.misp-project.org/objects.html#\\_exploit\\_poc](https://www.misp-project.org/objects.html#_exploit_poc) (accessed on 20 February 2021).