A crawler architecture for harvesting the clear, social, and dark web for IoT-related cyber-threat intelligence

Paris Koloveas Thanasis Chantzios Christos Tryfonopoulos Spiros Skiadopoulos
University of the Peloponnese, GR22131, Tripolis, Greece
{*pkoloveas, tchantzios, trifon, spiros*}*@uop.gr*

***Abstract*—The clear, social, and dark web have lately been identified as rich sources of valuable cyber-security information that –given the appropriate tools and methods– may be identified, crawled and subsequently leveraged to actionable cyber-threat intelligence. In this work, we focus on the information gathering task, and present a novel crawling architecture for transparently harvesting data from security websites in the clear web, security forums in the social web, and hacker forums/marketplaces in the dark web. The proposed architecture adopts a two-phase approach to data harvesting. Initially a machine learning-based crawler is used to direct the harvesting towards websites of interest, while in the second phase state-of-the-art statistical language modelling techniques are used to represent the harvested information in a latent low-dimensional feature space and rank it based on its potential relevance to the task at hand. The proposed architecture is realised using exclusively open-source tools, and a preliminary evaluation with crowdsourced results demonstrates its effectiveness.**

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

I. 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. Lately, the utilisation of IoT devices on a number of applications, ranging from home automation to monitoring of critical infrastructures, has created an even more complicated cyber-defense 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 today's cyber-attacks, ranging from botnet-building for DDoS attacks, 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 artefacts 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 organization 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, analysed, 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, we are particularly interested in gathering CTI from the clear, social, and dark web where threat actors collaborate, communicate and plan cyber-attacks. Such an approach allows us to provide visibility to a number of sources that are of preference to threat-actors and identify timely CTI including zero-day vulnerabilities and exploits. To do so, we envision an integrated framework that encompasses key technologies for pre-reconnaissance CTI *gathering, analysis* and *sharing* through the use of state-of-the-art tools and technologies. In this context, newly discovered data from various sources will be inspected for their relevance to the task (*gathering*), and discovered CTI in the form of vulnerabilities, exploits, threat actors, or cyber-crime tools will be identified (*analysis*) and stored in a vulnerability database using existing formats like CVE[1] and CPE[1] (*sharing*).

In this work we focus on the *gathering* part of the envisioned framework, and present a novel architecture that is able to transparently provide a crawling infrastructure for a variety of CTI sources in the clear, social, and dark web. Our approach employs a thematically focused crawler for directing the crawl towards websites of interest to the CTI gathering task. This is realised by resorting to a combination of machine learing techniques (for open domain crawl) and regex-based link filtering (for structured domains like forums). The retrieved content is stored in an efficient NoSQL datastore and is retrieved for further inspection in order to decide its usefulness to the task. This is achieved by employing statistical language modelling techniques [1] 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). These techniques allow us to train our language model to (i) capture

This work has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no. 786698. 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.

[1]https://www.mitre.org

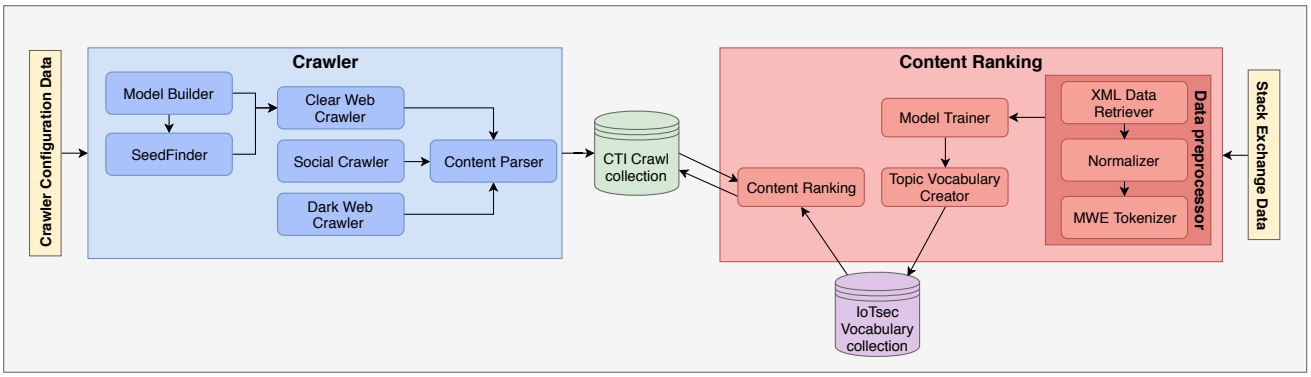

Figure 1. A high-level view of the proposed architecture

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 at hand 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. Notice that the post-mortem inspection of the crawled content is necessary, since the thematically focused crawl is forced to make a crude decision on the link relevance (and if it should be visited or not) since it resorts on a limited feature space (e.g., alt-text of the link, words in the url, or relevance of the parent page).

To the best of our knowledge, this is the first approach to CTI gathering that views 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. This novel approach to CTI gathering is integrated to an infrastructure that is entirely open-source and able to transparently monitor the clear, social, and dark web.

The rest of the paper is organised as follows. In the next section, we present the architecture and provide details on several design and implementation choices, while in Section III we present our evaluation plan and a preliminary effectiveness evaluation using both crowdsourced results and anecdotal examples. Finally, Section IV outlines related work, while Section V discusses future research directions.

II. Architecture

The proposed architecture consists of two major components: the *crawling module* and the *content ranking module*. The idea behind this two-stage approach is the openness of the topic at hand that cannot be accurately modelled by a topical crawler, which mainly focuses on staying within the given 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). To overcome this challenge, a focused crawler that employs machine learning techniques to direct the crawl is aided by advanced language models to decide on the usefulness of the crawled websites by utilising the harvested content. To this end, we designed a novel module that employs a ranking-based approach to the assessment of the usefulness of a website content using language models. To capture the background knowledge regarding vocabulary and term correlations, we resorted to latent topic models [1], while ranking is modelled as vector similarity to the task.

The proposed architecture has been entirely designed on and developed using open-source software; it employs an open-source focused crawler[2], an open source implementation of word embeddings[3] for the latent topic modeling, and an open-source NoSQL database[4] for the storage of the topic models and the crawled content. The front-end modules are based on HTML, CSS and JavaScript. Figure 1 displays the complete architecture with all the sub-components that will be detailed in the following sections.

A. Clear/social/dark web crawling

The crawling module is based upon the open-source implementation of the ACHE Crawler[2] and contains three different sub-components: (i) a focused crawler for the clear web, (ii) an in-depth crawler for the social web (i.e., forums), and (iii) a TOR-based crawler for the dark web. Below, we outline the characteristics of the different sub-components and present the rationale behind our implementation choices.

Clear web crawler. This sub-component is designed to perform *focused crawls* on the clear web with the purpose to discover new resources that may contain cyber-threat intelligence. To direct the crawl towards topically relevant websites we utilise an SVM classifier, which is trained by resorting on an equal number of positive and negative examples of websites that are used as input to the *Model Builder* component of ACHE. Table I contains some example entries that are specific to our task. Subsequently, the *SeedFinder* [2] component is utilised to aid the process of locating initial seeds for the focused crawl on the clear web; this is achieved by combining the classification model built previously with a user-provided query relevant to the topic - in our case we use the query “iot vulnerabilities”.

[2]https://github.com/ViDA-NYU/ache
[3]https://radimrehurek.com/gensim/
[4]https://www.mongodb.com/

Table I
POSITIVE & NEGATIVE WEBPAGE EXAMPLES

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

Social web crawler. This sub-component is used to perform *in-depth crawls* on specific selected forums on the social web, where the topic of discussion matches the given task. To this end, the social web crawler can be provided with links to discussion threads on IoT vulnerabilities and use them to traverse the forum structure and download all relevant discussions on the topic. Notice that contrary to clear web crawling, in the forum crawl all links are considered relevant by default since they correspond to discussions over the given topic, so there is no need for utilising a page classification model. However, to filter out parts of the forum that are irrelevant or non-informative (e.g., user info pages), we employ *regex-based link filters* that may be applied within specific forums or in a cross-forum fashion. Table II contains some example link filters. Notice that this type of crawl is primarily used for thread/forum monitoring and is not targeted in identifying new websites.

Dark web crawler. This sub-component is used to perform *in-depth crawls* on specific websites on the dark web by utilising TOR proxies. To do so, the crawler is provided with a number of onion links that correspond to hacker forums or marketplaces selling cybercrime tools and zero-day vulnerabilities/exploits and monitors the discussions for content of interest. To overcome user authentication mechanisms that are often in place in dark web forums/marketplaces, the dark web crawler requires an initial manual login. After a successful user authentication, the session cookies are stored and are utilised (via HTTP requests) in subsequent visits of the crawler to simulate user login. After each crawl completes a set interval, the crawled HTML pages are parsed by the *content parser* sub-component, which extracts the textual content along with useful metadata (e.g., bitcoin value of sold cybercrime tools or user fame/activity/reputation level).

All content from the different (clear/social/dark) web crawling components is downloaded in its raw HTML format and stored in a NoSQL document store (mongoDB[4]) for further processing as discussed in the next section.

B. Content ranking and classification

Deciding whether a crawled 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 designed and implemented a novel content ranking module that assesses the relevance and usefulness of the crawled content. To do so, we represent the topic as a vocabulary distribution by utilising distributional vectors of related words; for example

Table II
REGEX-BASED LINK FILTERS

Category	Regex	Operation
Whitelist	https:\/\/www\.wilderssecurity\.com\/threads\/.*	Crawl content only from the ***Threads*** section of the domain
Whitelist	https:\/\/blogs\.oracle\.com\/security\/.*	Crawl all articles about ***Security***
Blacklist	https:\/\/www\.wilderssecurity\.com\/members\/.*	Crawl the entire domain apart from the ***Members Area***
Blacklist	https:\/\/www\.securityforum\.org\/events\/.*	Crawl the entire domain apart from ***Events***

a topic on IoT security could be captured by related words and phrases like "Mirai botnet", "IoT", or "exploit kits". Such salient phrases related to the topic may be obtained by un-/semi-supervised training of latent topic models over external datasets such as IoT and security related forums. 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 by state-of-the-art latent topic models [1].

Since useful cyber-threat intelligence manifests itself in the form of cyber-security articles, user posts in security/hacker forums, or advertisement posts in cybercrime marketplaces, it can also be characterised as distributional vectors of salient words. Then, the similarity between the distributional vectors of harvested content and the given topic (i.e., IoT vulnerabilities) may be used to assess the content relevance to the topic.

To better capture the salient vocabulary that is utilised by users for the IoT security domain we resorted to a number of different discussion forums within the Stack Exchange ecosystem to create a training dataset. To this end, we utilised the *Stack Exchange Data Dump*[5] to get access to IoT and information security related discussion forums including *Internet of Things*, *Information Security*, *Arduino*, *Raspberry Pi*, and others. The utilised data dumps contain user discussions in Q&A form, including the text from *posts*, *comments* and related *tags*, and were used as input to the *data preprocessor* sub-component described below.

Data preprocessor. This sub-component is responsible for the data normalisation process that involves a number of steps described in the following. The input data for the sub-component is typically XML-formatted and at first an XML DOM parser is used to parse the data and keep only the useful part that contains user posts, comments, and tags. The parsed data are, subsequently, fed into the *Normalizer* that performs typical normalisation (e.g., case folding, symbol removal) and anonymisation (e.g., username elimination) actions. Finally, the third step in the preprocessing phase (*Multi-Word Expression tokenisation*) includes the identification and characterisation of important multi-word terms (such as "exploit kits" or "Mirai botnet") in order to extend the functionality of the skip-gram model [1] for such terms.

Model trainer. The preprocessed document corpus is subsequently utilised to train the language model [1]; this is

[5]https://archive.org/details/stackexchange

Table III
MOST RELEVANT TERMS FOR TAG DDOS

rank	term	rank	term	rank	term
#1	dos	#6	flood	#11	slowloris
#2	volumetric	#7	aldos	#12	botnet
#3	flooding	#8	floods	#13	drdos
#4	cloudflare	#9	ip spoofing	#14	blackholing
#5	prolexic	#10	radware	#15	amplification

done by using the Gensim[3] open source implementation of word2vec (based on a neural network for word relatedness), setup with a latent space of 150 dimensions, a training window of 5 words, a minimum occurrence of 1 term instance, and 10 parallel threads. The result of the training is a 150-dimensions distributional vector for each term that occurs at least once in the training corpus.

Topic vocabulary creator. To automatically extract the set of salient words that will be used to represent the topic we utilised the extracted user tags, and augmented them with the set of N most related terms in the latent space for each user tag; term relatedness was provided by the trained language models and the corresponding word vectors. Table III shows an example of the most relevant terms to the DDoS user tag, for N =5,10, and 15. The resulting (expanded) vocabulary is stored in a separate NoSQL document store (mongoDB[4]).

Content ranking. To assess the relevance and usefulness of the crawled content we employ the *content ranking* sub-component; this component utilises the expanded vocabulary created in the previous phase to decide how similar a crawled post is to the topic by computing the similarity between the topic and post vectors. This is done as follows.

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.,

$$\vec{T} = \sum\nolimits_{\forall i} \vec{t_i}$$

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 [3]. Namely for a topic T and a post containing the set of words $\{w_1, w_2, \ldots\}$, the post vector is computed as

$$\vec{P} = \sum\nolimits_{\forall j} \cos\left(\vec{w_j}, \vec{T}\right) \cdot \vec{w_j}$$

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

$$r = \cos\left(\vec{T}, \vec{P}\right)$$

Having computed a relevance score for every crawled post in the NoSQL datastore, the classification task of identifying relevant/useful posts is trivially reduced to either a thresholding or a top-k selection operation. Notice that the most/least relevant websites may also be used to reinforce the crawler model (i.e., as input to the *Model Builder* sub-component).

Table IV
RELEVANCE SCORE COMPUTATION

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

Comparing the pros and cons of thresholding and top-k post selection in the context of assessing a continuous crawling task is an ongoing research effort.

III. PRELIMINARY EVALUATION

In this section we present a preliminary evaluation of the architecture and the evaluation plan for a thorough investigation of each system component.

For the purposes of the evaluation, we ran a *focused crawl* with a model consisting of 7 positive and 7 negative URLs, a sample of which may be seen on Table I, and 21 seeds extracted by the *SeedFinder* component for the query "iot vulnerabilities". The crawl harvested around $22K$ websites per hour/per thread on a commodity machine; around 10% of the frontier links were considered *relevant* according to the crawler model and were thus harvested.

In order to determine the quality of information within the harvested URLs that were considered relevant in the *focused crawl*, we developed a web-based *evaluation tool* (a snapshot of which is shown in Figure 2) for collecting crowdsourced data from human experts. Using this tool, human judges are provided with a random harvested website and are asked to assess its relevance on a 4-point scale; these assessments are expected to showcase the necessity of the language models and drive our classification task (i.e., help us identify appropriate thresholding/top-k values). Early results from a limited number of judgements on the clear web crawl show that about 1% of the harvested websites contain actionable cyber-threat intelligence, while (as expected) the percentage is higher for the social and dark web.

Besides the evaluation tool, we have also implemented a visualisation component for the computation of the relevance score for posts. This component highlights vocabulary words on the actual crawled posts, and displays the computed relevance score according to our model. A sample output of the component for a random post is shown in Table IV.

IV. RELATED WORK

Web crawlers, typically also known as robots or spiders, are tightly connected to information gathering from online sources. In this section we review the state-of-the-art in crawling by (i) outlining typical architectural alternatives that fit the crawling task and (ii) categorising the different crawler types based on the type of the targeted content.

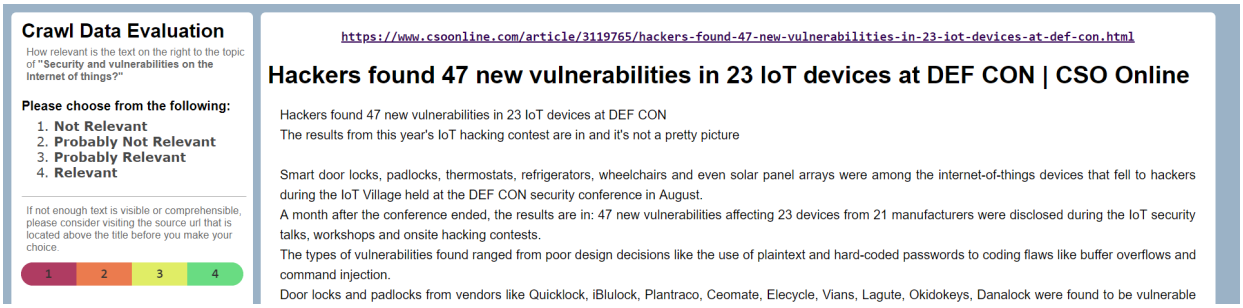

Figure 2. Web-based evaluation tool for collecting crowdsourced data from human experts

A. Architectural typology

Depending on the crawling application, the available hardware, the desired scalability properties and the ability to scale up/out the existing infrastructure, related literature provides a number of architectural alternatives [4]–[6].

Centralized. Typically, special-purpose or small-scale crawlers follow a centralised architecture [4]; the page downloading, the URL manipulation, and the page storage modules, resort in a single machine. This centralised architecture is, naturally, easier to implement, simpler to deploy, and straightforward to administer, but is limited to the capabilities of the hardware and thus cannot scale well. For this reason, the more sophisticated crawler designs put effort in scaling out, i.e., exploiting the inherently distributed nature of the web and adopt some form of decentralisation.

Hybrid. Hybrid crawler architectures (e.g., [7]) are the norm in the architectural typology as they aim for a conceptually simple design that involves distributing some of the processes, while keeping others centralised. In such architectures, the page downloading module is typically 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 centralised architecture and the scalability of a distributed system; however the centralised component usually acts as a bottleneck for the crawling procedure and represents a single point of failure.

Parallel/Distributed. A parallel crawler [8], [9] consists of multiple crawling processes (usually referred to as C-procs in crawler jargon), where each such process performs all the basic tasks of a crawler. To benefit from the parallelisation of the crawling task the frontier is typically split among the different processes, while to minimise overlap in the crawled space, links and other metadata are communicated between the processes. When all processes run on the same LAN then we refer to an intra-site parallel crawler, while when C-proc's run at geographically distributed locations connected by a WAN (or the Internet) we refer to a distributed crawler.

Peer-to-peer. The advent of peer-to-peer computing almost two decades ago introduced peer-to-peer search engines like Minerva [10]; this in turn gave rise to the concept of peer-to-peer crawlers [11], [12]. Peer-to-peer crawlers constitute a special form of distributed crawlers that are typically 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 emphasise crawl personalisation and demonstrate large-scale collaboration usually by means of an underlying distributed routing infrastructure [11].

Cloud-based. Lately, the requirement for more effective use of resources by means of elasticity gave rise to a new crawler paradigm: the cloud-based crawlers [13], [14], which revived known machinery to a renewed scope, versatility and options. Such architectures use cloud computing features alongside big data solutions like Map/Reduce and NoSQL databases, to allow for resource adaptable web crawling and serve the modern the Data-as-a-Service (DaaS) concept.

B. Usage typology

Although the first web crawlers that set the pathway for the spidering technology were developed for the clear (or surface) web, in the course of time specialised 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 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 include [4], [15].

Web 2.0. The advent of the user-generated content philosophy and the participatory culture of Web 2.0 sites like blogs, forums and social media, formed a new generation of specialised crawlers that focused on forum [16]–[19], blog/microblog [20]–[22], and social media [23], [24] spidering. The need for specialised crawlers for these websites emerged from (i) the content quality, (ii) the inherent structure that is present in forums/blogs, and (iii) the implementation particularities (e.g., Javascript-generated URLs) that make other crawler types inapplicable or inefficient.

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 anonymisation networks like Tor[6]

[6]https://www.torproject.org/

and I2P[7] (dark web), gave rise to specialised crawlers [25]. To this end, over the last ten years, a number of works related to deep web crawling have been published [26]–[28], investigating also (i) different architectural [29], [30] and automation [31] options, (ii) quality issues such as sampling [32], query-based exploration [33], duplicate elimination [34], and (iii) new application domains [35], [36].

Cloud. Finally, the elasticity of resources and the popularity of cloud-based services inspired a relatively new line of research focusing on crawler-based service configuration [37] and discovery in cloud environments [38].

V. Outlook

We are currently working on a thorough evaluation of our architecture. Our future research plans involve the extraction of cyber-threat intelligence from the relevant harvested content, by utilising natural language understanding for named entity recognition/disambiguation.

References

[1] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *NIPS*, 2013.

[2] K. Vieira, L. Barbosa, A. S. Silva, J. Freire, and E. Moura, "Finding seeds to bootstrap focused crawlers," *WWW*.

[3] J. Biega, K. Gummadi, I. Mele, D. Milchevski, C. Tryfonopoulos, and G. Weikum, "R-Susceptibility: An IR-Centric Approach to Assessing Privacy Risks for Users in Online Communities," in *ACM SIGIR*, 2016.

[4] M. Najork, "Web crawler architecture," in *Encyclopedia of Database Systems*, 2009.

[5] J. M. Hsieh, S. D. Gribble, and H. M. Levy, "The architecture and implementation of an extensible web crawler," in *USENIX, NSDI*, 2010.

[6] A. Harth, J. Umbrich, and S. Decker, "Multicrawler: A pipelined architecture for crawling and indexing semantic web data," in *ISWC*, 2006.

[7] V. Shkapenyuk and T. Suel, "Design and implementation of a high-performance distributed web crawler," in *ICDE*, 2002.

[8] F. Ahmadi-Abkenari and A. Selamat, "An architecture for a focused trend parallel web crawler with the application of clickstream analysis," *Inf. Sci.*, vol. 184, 2012.

[9] D. L. Quoc, C. Fetzer, P. Felber, E. Rivière, V. Schiavoni, and P. Sutra, "Unicrawl: A practical geographically distributed web crawler," in *IEEE, CLOUD*, 2015.

[10] C. Zimmer, C. Tryfonopoulos, and G. Weikum, "Minervadl: An architecture for information retrieval and filtering in distributed digital libraries," in *ECDL*, 2007.

[11] O. Vikas, N. J. Chiluka, P. K. Ray, G. Meena, A. K. Meshram, A. Gupta, and A. Sisodia, "Webminer–anatomy of super peer based incremental topic-specific web crawler," in *ICN*, 2007.

[12] B. Bamba, L. Liu, J. Caverlee, V. Padliya, M. Srivatsa, T. Bansal, M. Palekar, J. Patrao, S. Li, and A. Singh, "Dsphere: A source-centric approach to crawling, indexing and searching the world wide web," in *ICDE*, 2007.

[13] K. Gupta, V. Mittal, B. Bishnoi, S. Maheshwari, and D. Patel, "AcT: Accuracy-aware crawling techniques for cloud-crawler," *WWW*, 2016.

[14] Y. Li, L. Zhao, X. Liu, and P. Zhang, "A security framework for cloud-based web crawling system," in *WISA*, 2014.

[15] K. S. McCurley, "Incremental crawling," in *Encyclopedia of Database Systems*, 2009.

[16] J. Jiang, N. Yu, and C. Lin, "Focus: learning to crawl web forums," in *WWW*, 2012.

[17] J. Yang, R. Cai, C. Wang, H. Huang, L. Zhang, and W. Ma, "Incorporating site-level knowledge for incremental crawling of web forums: a list-wise strategy," in *ACM, SIGKDD*, 2009.

[18] Y. Wang, J. Yang, W. Lai, R. Cai, L. Zhang, and W. Ma, "Exploring traversal strategy for web forum crawling," in *ACM, SIGIR*, 2008.

[19] R. Cai, J. Yang, W. Lai, Y. Wang, and L. Zhang, "irobot: an intelligent crawler for web forums," in *WWW*, 2008.

[20] M. Hurst and A. Maykov, "Social streams blog crawler," in *ICDE*, 2009.

[21] S. Agarwal and A. Sureka, "A topical crawler for uncovering hidden communities of extremist micro-bloggers on tumblr," in *WWW*, 2015.

[22] R. Ferreira, R. Lima, J. Melo, E. Costa, F. L. G. de Freitas, and H. P. L. Luna, "Retriblog: a framework for creating blog crawlers," in *ACM, SAC*, 2012.

[23] F. Buccafurri, G. Lax, A. Nocera, and D. Ursino, "Crawling social internetworking systems," in *ASONAM*, 2012.

[24] A. Khan and D. K. Sharma, "Self-adaptive ontology based focused crawler for social bookmarking sites," *IJIRR*, 2017.

[25] G. Valkanas, A. Ntoulas, and D. Gunopulos, "Rank-aware crawling of hidden web sites," in *WebDB*, 2011.

[26] Y. Wang, J. Lu, J. Chen, and Y. Li, "Crawling ranked deep web data sources," *WWW*, vol. 20, no. 1, 2017.

[27] F. Zhao, J. Zhou, C. Nie, H. Huang, and H. Jin, "Smartcrawler: A two-stage crawler for efficiently harvesting deep-web interfaces," *IEEE TSC*, 2016.

[28] Q. Zheng, Z. Wu, X. Cheng, L. Jiang, and J. Liu, "Learning to crawl deep web," *Inf. Syst.*, vol. 38, no. 6, 2013.

[29] J. Zhao and P. Wang, "Nautilus: A generic framework for crawling deep web," in *ICDKE*, 2012.

[30] Y. Li, Y. Wang, and E. Tian, "A new architecture of an intelligent agent-based crawler for domain-specific deep web databases," in *IEEE/WIC/ACM, WI*, 2012.

[31] T. Furche, G. Gottlob, G. Grasso, C. Schallhart, and A. J. Sellers, "Oxpath: A language for scalable data extraction, automation, and crawling on the deep web," *VLDB J.*, 2013.

[32] J. Lu, Y. Wang, J. Liang, J. Chen, and J. Liu, "An approach to deep web crawling by sampling," in *IEEE/WIC/ACM, WI*, 2008.

[33] J. Liu, Z. Wu, L. Jiang, Q. Zheng, and X. Liu, "Crawling deep web content through query forms," in *WEBIST*, 2009.

[34] L. Jiang, Z. Wu, Q. Feng, J. Liu, and Q. Zheng, "Efficient deep web crawling using reinforcement learning," in *PAKDD*, 2010.

[35] Y. Li, Y. Wang, and J. Du, "E-FFC: an enhanced form-focused crawler for domain-specific deep web databases," *JIIS*, 2013.

[36] Y. He, D. Xin, V. Ganti, S. Rajaraman, and N. Shah, "Crawling deep web entity pages," in *ACM, WSDM*, 2013.

[37] M. Menzel, M. Klems, H. A. Lê, and S. Tai, "A configuration crawler for virtual appliances in compute clouds," in *IEEE, IC2E*, 2013.

[38] T. H. Noor, Q. Z. Sheng, A. Alfazi, A. H. H. Ngu, and J. Law, "CSCE: A crawler engine for cloud services discovery on the world wide web," in *IEEE, ICWS*, 2013.

[7] https://geti2p.net/