Semi-Automated Information Extraction from Unstructured Threat Advisories

Roshni R Ramnani
Accenture Technology Labs
roshni.r.ramnani
@accenture.com

Karthik Shivaram
Illinois Institute of Technology,
Chicago
kshivara@hawk.iit.edu

Shubhashis Sengupta
Accenture Technology Labs
shubhashis.sengupta
@accenture.com

Annervaz K M
Accenture Technology Labs
annervaz.k.m
@accenture.com

ABSTRACT

One of the fundamental challenges for information officers of most organizations today is the growing number of cyber security threats. This has led to an emerging field of *Cyber Threat Intelligence*, which is a mechanism to acquire, categorize and prioritize information regarding impending security threats from disparate online sources, enabling organizations to take the necessary steps to avoid compromising client data and protecting their hardware and software resources. Such information is published as formal security advisories which are largely in the form of unstructured or semi structured data. In this work we describe an approach to read large volume of such unstructured data and automatically extract useful nuggets of information like the exploit targets, techniques for the exploitation and recommended prevention guidelines. We use natural language processing techniques and a pattern identification framework to extract these information nuggets. We present some early results and observations.

CCS Concepts

•**Security and privacy** → *Intrusion/anomaly detection and malware mitigation ;*

Keywords

Cyber Security, Cyber Threat Intelligence, Natural Language Processing, Pattern Detection, Semi-supervised Learning

1. INTRODUCTION

Cyber security is a complex and multifaceted problem domain, as the online threat environment gets dynamic and daunting. The first known cyber-attack started in 1988 with Morris worm compromising UNIX systems [34]. In the recent years, the attacks have become more frequent, damaging and difficult to prevent. For example, in 2013, the Russian firm Kaspersky discovered attack *Red October* on Microsoft Office applications stealing data from embassies, energy providers, nuclear and critical infrastructure since 2007 [1]. As economic costs of these attacks increase; it has, therefore, been imperative that we pro-actively defend against these attacks through superior intelligence gathering and anticipation, and take effective counter measures.

One of the commonly used measure of prevention and intrusion detection against direct attacks on cyber infrastructure like Denial of Service (DoS) or BotNets have been signature analysis [23] or attack behavior analysis [5]. There have also been works on detecting potential Social Engineering attacks like phishing and fake sites [3]. Another popular counter measure has been social data and network analysis (such as reputation) within online Hacker communities. Projects such as Dark Web [9] tried to crawl through hacker blogs or IRC chatrooms to detect impending threat intentions.

In this work, we look at security advisories generated by various software or hardware organizations (like Microsoft or Cisco) as well as those issued by Governmental agencies or watchdogs (organizations such as Financial Services ISAC or Crowd Strike) that contain potential vulnerabilities or threat information. The growing number and complexity of systems deployed in our environment, potential vulnerabilities in them and increasing count of threat warnings issued, make it difficult to manually identify and analyze this information and take appropriate counter measures on time. We use natural language processing, semi-supervised pattern identification and matching techniques to extract key information present in these security advisories in a semi-automated manner from natural language documents. The information extracted is modeled on the basis of a well-known security model called Structured Threat Security Expression (STIX) [2]. We present the approach, experimentation and results in detail.

The rest of the paper is organized as follows: Section 2 explains the Cyber Security Domain Model by providing precise definitions for all Domain Model Elements. Section 3 presents a small introduction to Cyber Threat Analysis, and introduces a custom tool developed for it. In Section 4, we

ISEC '17, February 05-07, 2017, Jaipur, India
© 2017 ACM. ISBN 978-1-4503-4856-0/17/02...$15.00
DOI: http://dx.doi.org/10.1145/3021460.3021482

explain the overall approach taken for extracting such domain elements from unstructured text. Section 5 explains in detail the prototype we developed, which was experimented on publicly available documents. The results of the experiments along with a brief discussion is presented in section 6. We cover some related work in section 7. Section 8 concludes the paper and examines some future work possibilities.

2. CYBER SECURITY DOMAIN MODEL

Due to the challenges and multitude of applications revolving around Security, there is a definite need to define, communicate and collaborate around security concepts . In the past, ontologies or detailed taxonomies have been suggested as mechanisms to capture knowledge [15] [4] for Security. However, recently with the proliferation of Cyber Threats and the need for a dedicated mechanism to combat it, an entire ecosystem involving sources of vunerability information like Threat Advisories, publicly accessible security databases and a number of standards have evolved for information exchange. These standards provide a mechanism of information sharing across organizations [8]. Standards include OpenIOC [16], STIX [2], and Incident Object Description and Exchange Format (IODEF) [13]. As mentioned earlier, for our approach, we use a refined version of the STIX model.

2.1 Vocabulary

The precise definition of the Domain Elements used by us is given below:

1. *IOC (Indicators of Compromise)* : The term may be used to refer to specific artifacts left by an intrusion, or greater sets of information that allow for the detection of intrusions or other activities conducted by attackers. This may involve mentions of attacks like DOS, DDOS or specific CVE identification numbers.

2. *Threat Actors* : These may be specific individuals or groups responsible for the security attack. They tend to include organized cyber criminals and hacktivists.

3. *Campaigns* : These are specific organizations, organization types and countries which are the target of the attack.

4. *TTP (Tactics, Techniques and Procedures)* : This details the mechanism in which the attack could be performed including specific attack patterns, tools, infrastructure used.

5. *COA (Course of Action)* : This details recommendations or guidelines for preventing the attack. They could be as simple as following of recommended guidelines for update of software to detailed actions that must be taken to prevent the attack.

6. *Exploit Target* : This is the software or hardware device which could be affected in the attack.

2.2 Domain Elements

The elements that we have defined map to either Domain Entities like *IOC*, *Threat Actors* or Domain Descriptions like *COA*, *TTP*. Domain Entities refer to noun phrases or specific keywords. Some Entities like *IOC* correspond to well-defined terms which can be stored in public databases like National Vulnerability Database[1]. The presence of the term itself identifies it as a Domain Entity. Domain Entities like Campaigns and Targets correspond to known software/hardware components and/or organizations. Generic Named Entity Recognition(NER) [32] components can be used to identify these modules, however specific patterns will still need to be used, to identify these as relevant Cyber Security Domain Elements. Domain Descriptions correspond to sentences or paragraphs, which may also have different relevant domain entities within them.

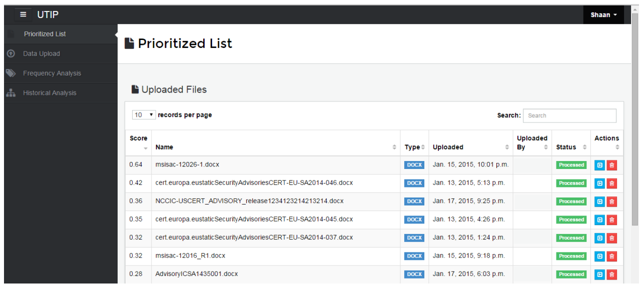
3. CYBER SECURITY ANALYSIS

Performing Cyber Security Analysis involves three main phases. First, is the integration with well known security advisory sources in the forms of RSS feeds, cyber security blogs, hacker forums etc. Second, is the extraction of security concepts from different sources as well as their categorization, together with cross referencing these concepts across sources. The third phase involves detailed analysis in the form of trend analysis, time-series analysis and filtering of organizational relevant data, along with detailed visualizations of the same. This data is used by Security Analysts to quickly assess the security vulnerability of the organization and perform the necessary steps for protection. Huge challenges lie due to the presence of large amounts data of being generated at a fast rate, leading to a need of performing automated analysis. Among these three phases, the second is the most challenging and unaddressed, since most of the sources are largely unstructured. We develop a custom approach and prototype to handle the second phase. The modules developed for the second phase are then integrated into a custom tool *UTIP* to perform Cyber Security Analysis, which is explained next.

3.1 UTIP

The modules relevant to the *Matching Phase* of the prototype (described in the next section), is integrated into a tool for performing Cyber Threat Analysis called Unstructured Threat Intelligence Processor(UTIP). The main purpose of the tool is to enable analysts to prioritize the threat advisories they receive regularly to determine which contain relevant information to their organization, as well as to perform high order analysis on the various threat data collected. On login, the security analyst is taken to the dashboard shown in figure 1. Here the dashboard shows all the uploaded documents ordered by their score, indicating their relevance. On clicking the play button, the user is taken to the screen shown in figure 2. Here the user can view all the Domain Entities extracted together with the sentence they originate from. The score on this screen is calculated as described in section 4.2. Analysts can also record feedback here in case the element extracted is wrong. The tool also provides provision for the analyst to perform multiple types of high-level analysis on the extracted data. One such analysis is the Frequency Analysis represented in the form of a heat map as shown in figure 3. Here, the analyst can view all common threat occurrences across all threat advisories in the system. Clicking on any colored STIX construct will expand on the information related to that Domain Element. The larger the number of occurrences of an element, the larger the size of the associated color in the diagram. Another type of anal-

[1]https://nvd.nist.gov/



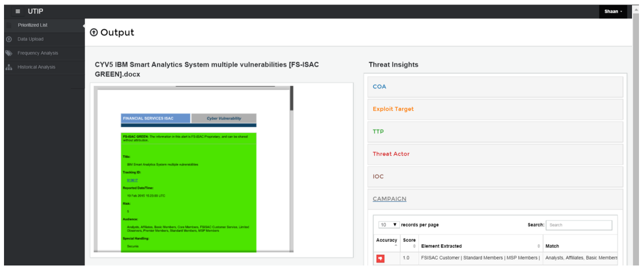
Prioritized List

Uploaded Files

10 records per page Search: Search

Score	Name	Type	Uploaded	Uploaded By	Status	Actions
0.64	msisac-12026-1.docx	DOCX	Jan. 15, 2015, 10:01 p.m.		Processed	
0.42	cert.europa.eustaticSecurityAdvisoriesCERT-EU-SA2014-046.docx	DOCX	Jan. 13, 2015, 5:13 p.m.		Processed	
0.36	NCCIC-USCERT_ADVISORY_release1234123214213214.docx	DOCX	Jan. 17, 2015, 9:25 p.m.		Processed	
0.35	cert.europa.eustaticSecurityAdvisoriesCERT-EU-SA2014-045.docx	DOCX	Jan. 13, 2015, 4:26 p.m.		Processed	
0.32	cert.europa.eustaticSecurityAdvisoriesCERT-EU-SA2014-037.docx	DOCX	Jan. 13, 2015, 1:24 p.m.		Processed	
0.32	msisac-12016_R1.docx	DOCX	Jan. 15, 2015, 9:18 p.m.		Processed	
0.28	AdvisoryICSA1435001.docx	DOCX	Jan. 17, 2015, 6:03 p.m.		Processed	

Figure 1: Uploaded Documents ordered by Priority



Output

CYV5 IBM Smart Analytics System multiple vulnerabilities [FS-ISAC GREEN].docx

Threat Insights

COA

Exploit Target

TTP

Threat Actor

IOC

CAMPAIGN

10 records per page Search: Search

Accuracy	Score	Element Extracted	Match
	1.0	FSISAC Customer | Standard Members | MSP Members |	Analysts, Affiliates, Basic Members

Figure 2: Domain Elements for a sample document

ysis is the Historical Analysis displayed in figure 4. Here, the analyst can view relationships among various threat advisories. Advisories which have common elements between one another will be shown as connected. The tool is highly configurable for doing all these analysis in detail.

4. OUR APPROACH

Our approach for gathering information nuggets from threat advisories involves mainly two phases. In the first phase, we employed an semi-automated method of pattern detection from the training document set. A pattern is defined as a sequence of tokens, which helps to identify and extract a domain element. Let us consider the example *Vulnerability in <ET>* as a pattern. The form *vulnerability in* acts as an anchor for the specification of Exploit Target(ET). A pattern typically consists of one more words which can be used to identify the sentence or explicitly identify the noun phrase related to the model elements. We further refined these patterns(sometime manually) and integrated additional techniques to develop a comprehensive scoring mechanism for overall pattern identification leading to more efficient Domain Element extraction. This is detailed out in the following subsections. In the next phase, we used

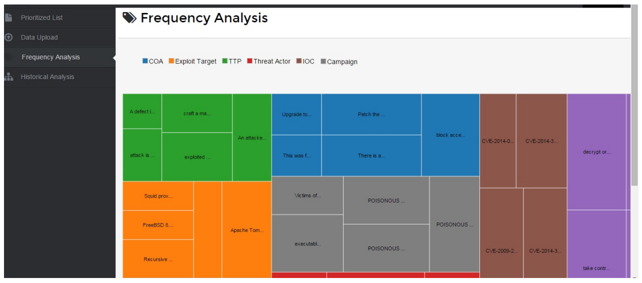


Figure 3: Frequency Analysis per Domain Element

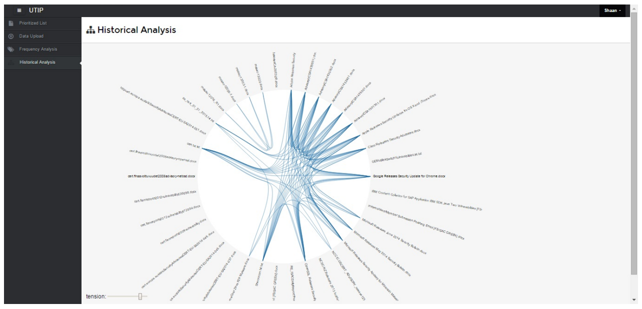


Figure 4: Historical Analysis across Documents

the extracted patterns and developed a matching logic by extracting security domain elements from a test set of documents. We formed a set of documents from publicly available documents [26] [37] [31] and then split the set to form the training and test set of documents.

4.1 Semi-Automated Pattern Extraction

For pattern identification we used a semi supervised approach, weaving together different techniques from literature. The first, involves the identification of nouns (keywords) that represent the various domain elements. Towards this purpose, we adopted and used a unsupervised bootstrapping technique Basilisk [36]. Basilisk works by identifying given pattern contexts surrounding a set of seed words and inducing more keywords that occur in the similar manner or a similar context as the seed words. The pattern context was defined by the pattern templates used in AutoSlog-TS [29]. Few examples of templates defined are: <subj> passive-verb, <subj> verb-infin., <subj> aux-noun, active-verb <dobj>, verb infin. <dobj>, gerund <dobj>, noun aux. <dobj>

When compared to other finite size context window based approaches, one of the advantages here is the use of dependency based information together with POS based information for formation of rigorous patterns. Using POS based information, adjectives or adverbs which could occur between *subj* and *passive-verb* can be disregarded. Using dependency information, pattern extraction for sentences with multiple verbs becomes more accurate. In our implementation, the pattern templates were reduced by merging different verb forms (including gerunds).

The Basilisk algorithm requires as input a set of patterns for each of the nouns in the text. The patterns that relate to a domain element are chosen based on the seed words given as input. In order to populate the seed words for each of the domain elements, we first found the frequencies of each of the words in the corpus. From the list we identified a set of high frequency seed words relevant to each Domain Element. Using this set of seed words, Basilisk algorithm was run iteratively collecting more nouns for each of the domain elements. The initial execution yielded mixed results. The samples are given in table 1. The first example for *Exploit Target* shows a correct extraction *Cisco ASA*. In the second example, the algorithm picks up *BGP Sessions* incorrectly. Examination of the mapping between patterns found for each domain element, and the new potential domain elements extracted per pattern per iteration highlighted a problem known as semantic drift [12]. Due to semantic drift, some erroneous or extremely general patterns in intermediate iterations generated wrong keywords, which further lead to erroneous patterns. Also, close examination revealed that most relevant patterns were generated in the first few iterations, which made it necessary to limit the number of iterations. Hence the approach was altered to include a manual step in which patterns were cleaned up between iterations and and the number of iterations was limited to 10. Additionally, since the AutoSlog pattern list incorporates both shorter and longer patterns, where shorter patterns could be subsets of longer patterns, a few redundant patterns could be removed. The patterns used for extracting these domain elements was recorded and served as input to our pattern repository. A few generated patterns are displayed in table 2.

4.2 Pattern Scoring

After identification of set of key terms and patterns for each domain element to be extracted, we followed a cumulative similarity scoring mechanism to identify and extract a domain element from each sentence of the new documents. This was added to further enhance and complement the already existing patterns that were generated to handle new kinds of documents. Also, automatic patterns generated focussed on Domain Entities. Additional contextual methods were needed for handling Domain Descriptions.

The cumulative score was aggregated from the lexical and semantic similarity between the sentence and the patterns. Also the lexical and semantic similarity was computed for the context as well as the candidate model. For each sentence the context is all text tokens apart from the candidate model elements in that sentence. For example in the sentence, *The windows versions 2.0.4 had fault in RMS module*, one candidate domain element is *RMS module* and other is *windows version 2.0.4*. For the candidate domain element, *RMS module*, the context is *had fault in*. We had seen a sample pattern for exploit target extraction, *Vulnerability in <ET>*. We compute a lexical score between the *had fault in* and *Vulnerability in*, defined as the Jaccard similarity [14] of the two sets of tokens, after doing basic stemming of the tokens. We compute also semantic score based on WordNet [25], for the unmatched tokens from lexical match. In our example we compute a match between *fault* and *vulnerability* using LIN measure [20]. This helps us to match the sentence even if *fault* was not appearing in the patterns for ET. The lexical and sematic similarity computation is done for the *RMS module* and the key terms of the exploit target as well. Then the score is aggregated. If the aggregate score is above a certain minimum threshold, a match is declared.

5. PROTOTYPE IMPLEMENTATION

A prototype, a high level overview diagram of which is depicted in figure 5, has been implemented in Java to automatically identify the Domain Elements from unstructured text. The prototype covers two phases of usage, one the Learning Phase (using the *Pattern Learner*) in which patterns were extracted from a training set, and Matching Phase (using *Pattern Matcher*) in which the patterns were matched against a test set. Information from unstructured data is extracted by using Apache Tika[2] which integrates multiple libraries to read and extract text from different types of documents like Word, PDF, txt, RSS feeds etc. The main

[2]https://tika.apache.org/

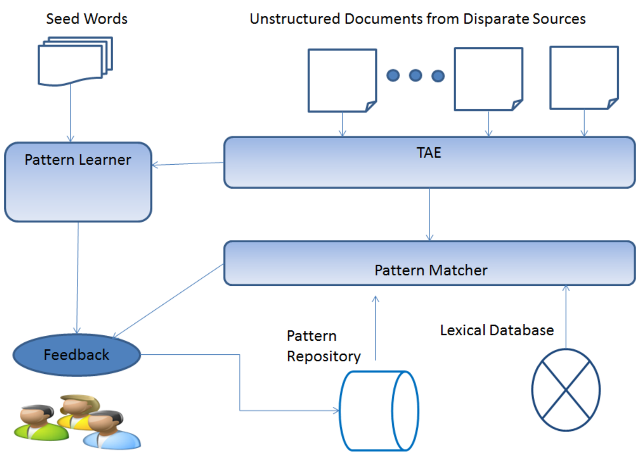


Figure 5: A schematic representation of Pattern Learning and Pattern Matching

components of the prototypes are explained briefly in the following subsections.

Text Analysis Engine(TAE).

The text content extracted from individual documents is passed to the Text Analysis Engine (TAE) for further processing. The TAE is architected in the form of layers. The first layer is the syntactic NLP component. The first step is dividing the text into sentences. The sentence detection is done using OpenNLP[3]. The next steps involve tokenizing each sentence, performing POS tagging and executing the dependency parser. All these operations are done using an open source software called ClearNLP[4] [10]. The next layer involves the interpretation of the parser output in identifying verb, nouns boundaries and their associated dependencies. The voice of the sentence is also identified. This plays a key role in the pattern matching process, such that even if the voice of the sentence and pattern are different, the match happens if the intent is the same. TAE also include basic NLP modules like stemmer namely the Porter stemming algorithm [28] for finding the base form of the words. The Stemmer is key in the pattern learning and matching phases as the stemmer ensures that all words are treated same regardless of the form in which they appear in the sentence. Similarly articles like ”the”, ”a” and modal verbs like ”shall” etc are ignored in the formation of patterns. We used Named Entity Recognition(NER) [32] to identify certain generic named entities like *country,organizations* etc. These are usually the exploit targets and form additional input for pattern matching. We used OpenNLP NER in the prototype.

Pattern Learner.

Pattern Learner module runs two key processes, *Pattern Extraction* and *Pattern Selection*. Pattern Extraction consists of custom code which extracts patterns as per the predefined AutoSlog-TS templates. In this step the entire dependency parse output is saved in the form of an xml document. The pattern templates are also defined in xml. For

[3]https://opennlp.apache.org/
[4]https://github.com/clir

Table 1: Samples of keywords added by Basilisk

Domain Element	Seed Words	Identified Nouns
Exploit Target	WebLogic Server, WebLogic Portal	Cisco ASA
Threat Actor	Unauthenticated attacker, Remote attacker	BGP sessions

Table 2: Sample extracted patterns

Domain Element	Identified Patterns
Threat Actor	<TA>cause
IOC	result in <IOC>
Exploit Target	Vulnerability in <ET>

extraction of patterns, XSLT is used to extract the patterns based on the pattern definition. This can allow pattern templates to be dynamically specified. These patterns extracted, together with the manually identified seed words are passed to the *Pattern Selection* module. The pattern selection module used an open source implementation of the Basilisk algorithm [35].

Pattern Repository.

The patterns generated during the learning phase are manually reviewed, refined and stored in the pattern repository category-wise, along with their associated weights. During the testing phase, user feedback regarding the positive and negative occurrences are included. The patterns stored are used to identify the different elements from the security domain by the pattern matcher.

Pattern Matcher.

This module takes the normalized sentences and computes the scores as described in section 4.2. We integrated with WordNet [25] using Java Wordnet Library(JWNL)[5] and used WS4J[6] for computing the semantic similarity measures.

Feedback.

The elements extracted as well as the patterns used for the extraction are displayed to the user. The user can then provide feedback as to the correctness of the output. During the matching phase this feedback is accumulated and stored in the pattern repository. Each positive feedback increases and each negative feedback decreases the weight of the patterns in pattern scoring for the subsequent extractions.

6. RESULTS

As an initial analysis , we ran the prototype on an test set of 18 threat advisories across sources. We plan to perform this evaluation on a larger test set using the deployed tool. The figure 6 shows the document-wise comparison of precision and recall on our test set. We can see that the overall recall for the documents is pretty high (above 80 percent). Also, recall do not vary much across source document types. However, there is a huge variation in the precision across documents. One issue affecting the precision is the extraction of sentences and sentence detection especially in the formation of complex sentence structures like bullets. More tailored preprocessing will be needed to handle these. Another is instances where words with different meanings having the same stem. For example, the words *attacker* and the verb *attack*. Attacker is actually a keyword for threat actor, whereas *attack* is one of the contextual words used for Exploit Target. Hence some sentences could be wrongly picked up as Exploit Target. We can build in heuristics based on POS tags to over come such issues. Another problem is the extraction of section headers matching the pattern. We could potentially extract meta data from the document and disregard such elements. In relation to this, certain Domain Descriptions like *COA* occur within a particular section of the document. Having a more complete understanding of the document structure would lead to better precision.

We saw high recall and moderate precision in the experiments. We saw some of the reasons for low precision. However, in these kind of extraction tasks, Recall is probably more important than precision. The reason being: if the recall is high most of the required information is extracted and presented to the user. The user can disregard the wrong results with very less effort. The motivation of the work being the reduction of human effort to identify the information from large documents, our work fits the bill. A measure called summarization [6] defined as the ratio between the size of the input document and size of the output has been defined in these contexts, to measure the reduction of human effort achieved. In our case the summarization value is good, indicating the tool is helping to reduce the human effort.

Figure 7 shows the distribution of extractions across categories. It can be seen from the distribution of the extractions across Domain Elements like *IOC* (due to multiple values like CVE numbers) and *TTP* (which extends to an entire paragraph covering multiple sentences), some domain elements have more occurrences than others.

7. RELATED WORK

In this section we will cover, the relevant works, which we couldn't refer to otherwise in the paper.

The identification of information from the unstructured text is a challenging task. Multiple approaches using the context where a word appears have been proposed in literature for this. The definition of context ranges from considering the immediate context surrounding a word within a window size to considering the entire text associated within the document [11, 24, 38] for a multitude of applications

[5]http://sourceforge.net/projects/jwordnet/
[6]https://code.google.com/p/ws4j/

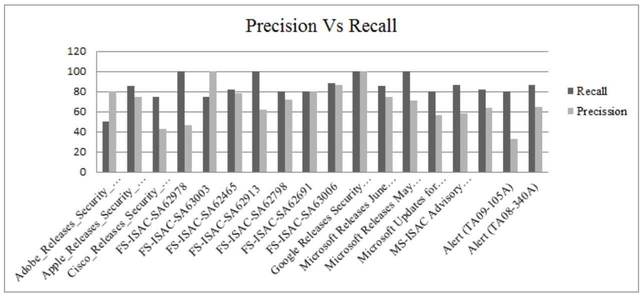


Figure 6: A schematic representation of the Precision vs Recall for the test set

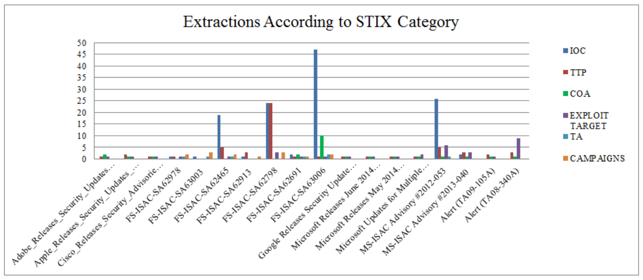


Figure 7: A schematic representation of the distributions of extractions across categories

like Named Entity Recognition, Word Sense Disambiguation, Word co-occurrences and Sentence Extraction. Pattern templates were pioneered by [17] for synonym extraction. In the IR domain T-Expressions [19] were used for automatic knowledge extraction. A multitude of pattern based and distributional similarity techniques have been studied for Domain Entity Extraction [33].In addition, unsupervised bootstrapping techniques using pattern templates have been successfully used in multiple domains [30, 21, 7]. Specific to the Cyber Security Domain, security concept extraction from unstructured text has been undertaken [18, 27]. In [18], a CRF based approach was used for cybersecurity entity extraction and concept spotting, which was then linked to DBpedia resources. In [27], after running an SVM classifier to identify relevant text, Wikitology and a computer security exploits ontology were used to identify concepts, entities, relations and events. A custom Bootstrapping approach was used by [22], using a definition of context based on surrounding words.

8. CONCLUSION AND FUTURE WORK

We presented an approach for semi-automated information extraction from threat advisories. We used a templates based pattern extraction technique together with a bootstrapping algorithm Basilisk to create a pattern repository from a set of training examples and then matched the patterns to information extraction from new documents. We presented the results from our limited experiments and illustrated the effectiveness of the approach and the tool.

We have not yet evaluated a fully supervised machine learned classification approach in the problem context. The non availability of the annotated corpora being the main reason. A supervised classification approach may not be directly applicable for domain element extraction. But filtering the input text relevant for categories before feeding to the pattern extraction module, will help in reducing the noise in the output patterns. The annotated corpora being unavailable our approach will be to first try semi-supervised approaches to build text classifiers. We also like our work to integrate with well known security databases and ontologies for detection of certain Domain Elements like *IOC* and *Exploit Target*. Our further attempt is to improve the recall close to 100 percent, even at the cost of decreasing the precision, but keeping the summarization value good enough, so that there is considerable reduction in human effort by using the tool in the overall process of identifying information from threat advisories.

Acknowledgment

We would like to thank members of the Accenture Technology Labs Security Team including Shimon Modi, Elvis Hovor, Shaan Mulchandani and Mohamed El-Sharkawi and all the members of Accenture Technology Labs Text Analyzer Team.

9. REFERENCES

[1] Red october : Diplomatic cyber attacks investigation. 2013. https://securelist.com/analysis/publications/36740/red-october-diplomatic-cyber-attacks-investigation/.

[2] Sean barnum : Standardizing cyber threat intelligence information with the structured threat information expression. http://stixproject.github.io/getting-started/whitepaper/, 2014.

[3] A. Abbasi and H. Chen. A comparison of tools for detecting fake websites. *IEEE Computer*, 42(10):78–86, 2009.

[4] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *Dependable and Secure Computing, IEEE Transactions on*, 1(1):11–33, 2004.

[5] V. Benjamin and H. Chen. Securing cyberspace: Identifying key actors in hacker communities. In *Intelligence and Security Informatics (ISI), 2012 IEEE International Conference on*, pages 24–29. IEEE, 2012.

[6] D. M. Berry, R. Gacitua, P. Sawyer, and S. F. Tjong. The case for dumb requirements engineering tools. In *Requirements Engineering: Foundation for Software Quality - 18th International Working Conference, REFSQ 2012, Essen, Germany, March 19-22, 2012. Proceedings*, pages 211–217, 2012.

[7] S. Brin. Extracting patterns and relations from the world wide web. In *International Workshop on The World Wide Web and Databases*, pages 172–183. Springer, 1998.

[8] E. W. Burger, M. D. Goodman, P. Kampanakis, and K. A. Zhu. Taxonomy model for cyber threat intelligence information exchange technologies. In *Proceedings of the 2014 ACM Workshop on Information Sharing &#38; Collaborative Security*, WISCS '14, pages 51–60, New York, NY, USA, 2014. ACM.

[9] H. Chen. Exploring extremism and terrorism on the

web: the dark web project. In *Intelligence and Security Informatics*, pages 1–20. Springer, 2007.

[10] J. D. Choi. *Optimization of Natural Language Processing Components for Robustness and Scalability.* PhD thesis, Boulder, CO, USA, 2012. AAI3549172.

[11] K. W. Church and P. Hanks. Word association norms, mutual information, and lexicography. *Computational linguistics*, 16(1):22–29, 1990.

[12] J. R. Curran, T. Murphy, and B. Scholz. Minimising semantic drift with mutual exclusion bootstrapping. In *Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics*, volume 6, 2007.

[13] R. Danyliw, J. Meijer, and Y. Demchenko. The incident object description exchange format (iodef). *Internet Engineering Task Force (IETF), RFC-5070*, 2007.

[14] M. M. Deza and E. Deza. *Encyclopedia of distances.* Springer, 2009.

[15] S. Fenz and A. Ekelhart. Formalizing information security knowledge. In *Proceedings of the 4th international Symposium on information, Computer, and Communications Security*, pages 183–194. ACM, 2009.

[16] O. Framework. An open framework for sharing threat intellegence. http://www.openioc.org/.

[17] M. A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th Conference on Computational Linguistics - Volume 2*, COLING '92, pages 539–545, Stroudsburg, PA, USA, 1992. Association for Computational Linguistics.

[18] A. Joshi, R. Lal, and T. Finin. Extracting cybersecurity related linked data from text. In *Semantic Computing (ICSC), 2013 IEEE Seventh International Conference on*, pages 252–259. IEEE, 2013.

[19] B. Katz. Using english for indexing and retrieving. Technical report, DTIC Document, 1988.

[20] D. Lin. An information-theoretic definition of similarity. In *ICML*, volume 98, pages 296–304, 1998.

[21] T. McIntosh and J. R. Curran. Weighted mutual exclusion bootstrapping for domain independent lexicon and template acquisition. In *Proceedings of the Australasian Language Technology Association Workshop*, volume 2008, 2008.

[22] N. McNeil, R. A. Bridges, M. D. Iannacone, B. Czejdo, N. Perez, and J. R. Goodall. Pace: Pattern accurate computationally efficient bootstrapping for timely discovery of cyber-security concepts. In *Machine Learning and Applications (ICMLA), 2013 12th International Conference on*, volume 2, pages 60–65. IEEE, 2013.

[23] L. Mendonça and H. Santos. Botnets: a heuristic-based detection framework. In *Proceedings of the Fifth International Conference on Security of Information and Networks*, pages 33–40. ACM, 2012.

[24] R. Mihalcea and P. Tarau. Textrank: Bringing order into texts. Association for Computational Linguistics, 2004.

[25] G. A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41, Nov. 1995.

[26] M.S-ISAC. Multi-state information sharing centre. http://msisac.cisecurity.org/.

[27] V. Mulwad, W. Li, A. Joshi, T. Finin, and K. Viswanathan. Extracting information about security vulnerabilities from web text. In *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2011 IEEE/WIC/ACM International Conference on*, volume 3, pages 257–260. IEEE, 2011.

[28] M. Porter. An algorithm for suffix stripping. http://tartarus.org/martin/PorterStemmer/def.txt.

[29] E. Riloff. Automatically generating extraction patterns from untagged text. In *Proceedings of the national conference on artificial intelligence*, pages 1044–1049, 1996.

[30] E. Riloff, R. Jones, et al. Learning dictionaries for information extraction by multi-level bootstrapping. In *AAAI/IAAI*, pages 474–479, 1999.

[31] F. Services. Information sharing and analysis center. https://www.fsisac.com/.

[32] R. Sharnagat. Named entity recognition: A literature survey, 2014.

[33] S. Shi, H. Zhang, X. Yuan, and J.-R. Wen. Corpus-based semantic class mining: distributional vs. pattern-based approaches. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 993–1001. Association for Computational Linguistics, 2010.

[34] E. H. Spafford. The internet worm program: An analysis. *ACM SIGCOMM Computer Communication Review*, 19(1):17–57, 1989.

[35] J. Tablet. Basilisk independent study. http://www.cs.utah.edu/~jtabet/basilisk, 2009.

[36] M. Thelen and E. Riloff. A bootstrapping method for learning semantic lexicons using extraction pattern contexts. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 214–221. Association for Computational Linguistics, 2002.

[37] USCERT. United states computer emergency readiness team. https://www.us-cert.gov/.

[38] D. Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pages 189–196. Association for Computational Linguistics, 1995.