# **Tutorial: Text Analytics for Security**

Tao Xie
Department of Computer Science
University of Illinois at Urbana-Champaign
taoxie@illinois.edu

William Enck
Department of Computer Science
North Carolina State University
enck@cs.ncsu.edu

# **ABSTRACT**

Computing systems that make security decisions often fail to take into account human expectations. This failure occurs because human expectations are typically drawn from in textual sources (e.g., mobile application description and requirements documents) and are hard to extract and codify. Recently, researchers in security and software engineering have begun using text analytics to create initial models of human expectation. In this tutorial, we provide an introduction to popular techniques and tools of natural language processing (NLP) and text mining, and share our experiences in applying text analytics to security problems. We also highlight the current challenges of applying these techniques and tools for addressing security problems. We conclude the tutorial with discussion of future research directions.

# **Categories and Subject Descriptors**

D.4.6 [Operating Systems]: Security and Protection—Invasive software; D.2.1 [Software Engineering]: Requirements/Specifications; I.2.7 [Artificial Intelligence]: Natural Language Processing—Text analysis

# **Keywords**

Security; human expectations; text analytics

#### 1. INTRODUCTION

Context is critical for making security decisions. Security policies often base access decisions on temporal context (e.g., time of day) and environmental context (e.g., geographic location). An OS access control policy frequently considers execution context (e.g., user ID, program arguments, previous inputs). Security analysis for programs often uses contexts of control flow and data flow. Indeed there are many forms of context considered in security.

One form of context is frequently overlooked: human expectations, e.g., did a human expect a certain functionality to occur? This omission may seem odd given that computer

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

HotSoS '16 April 19-21, 2016, Pittsburgh, PA, USA © 2016 Copyright held by the owner/author(s). ACM ISBN 978-1-4503-4277-3/16/04. DOI: http://dx.doi.org/10.1145/2898375.2898397

"A computer is secure if you can depend on it and its software to behave as you expect" [3, p. 5]. The difficulty (and the deficiency in this definition) is that human expectations are often difficult to formally (and even informally) define. Without a concrete definition of "expectation," the security of a system cannot be verified. Humans draw expectations from many sources. One com-

security is sometimes defined with respect to expectation:

Humans draw expectations from many sources. One common source is textual information. For example, developers derive security expectations from API documentation, comments in code, and requirements documents. Users derive security expectations from textual descriptions of program functionality (e.g., mobile application description), as well as text (e.g., UI texts) displayed during runtime. Hence, textual information sources have become inputs from which researchers can derive context.

In this tutorial (which is based on our previous tutorial [2]), we present how text analytics can be applied to security. The tutorial provides an introduction of popular techniques and tools of NLP and text mining such as Word-Net [11], Stanford Parser [8], and Weka [10]. We describe a number of success stores of applying NLP to security. Finally, we discuss the current challenges of applying NLP and text mining techniques and tools for security problems, concluding with future research directions.

# 2. SAMPLE WORK ON TEXT ANALYTICS FOR SECURITY

Our prior work is amongst several that have considered text analytics for security. In our Whyper work [4], we used natural language processing (NLP) to bridge the gap between permission requested by an Android application and the expectations of a user who has read the application description, i.e., textual description of the application in the Google Play Store. The key insight is that existing program analysis tools identify malicious and privacy infringing behavior by comparing program execution to a list of rules created by some expert. However, those rules must be placed within the context of user expectations. For example, if an application is designed to record a user's phone calls, then recording audio in the background during a phone call is expected, and therefore should be allowed. While WHYPER is currently limited to the functional semantics of permissions, the concepts can be extended to other notions of functional semantics (e.g., data flows).

We have also used NLP to automatically extract access control policies (ACPs) from textual requirements documentation [12]. In general, like other types of textual doc-

uments, textual requirements written in English are typically unstructured, ambiguous, and include implicit information, posing challenges for NLP. However, in textual requirements, ACP sentences (i.e., textual requirements sentences for describing ACP rules) tend to follow specific styles such as: [subject] [can/cannot/is allowed to] [action] [resource] for role-based ACPs. To leverage such insight, we developed the Text2Policy approach [12], which includes adapted NLP techniques designed around a model (such as the ACP model) to automatically extract model instances from textual requirements documents. Our Text2Policy approach consists of three main steps: (1) apply linguistic analysis to parse textual requirements documents and annotate words and phrases in sentences from the textual requirements documents with semantic meanings, (2) construct model instances using annotated words and phrases in the sentences, and (3) transform these model instances into formal specifications. More recently, without requiring predefined patterns in the ACP model, Slankas et al. [7] combined techniques from information extraction and machine learning to discover patterns that represent ACPs in sentences, no matter whether or not these ACPs follow one of Text2Policy's predefined patterns in the ACP model.

This work on applying text analytics to security is motivated by text analytics for software engineering (SE). SE data contains a rich amount of natural language text: requirements, code comments, program identifier names, documents, commit messages, release notes, mailing list discussions, etc. The natural language text is essential in the software engineering process to help software developers and software engineering researchers understand and maintain software better. While applying NLP and text mining to SE dates back over a decade [1], it has recently re-emerged as a hot topic [13]. Many recent studies showed that automated analysis of natural language text can improve software reliability, programming productivity, software maintenance, and software quality in general. For example, Shepherd et al. [6] applied NLP techniques such as part-of-speech (POS) tagging to find word paraphrases to expand code search; Tan et al. [9] leveraged NLP techniques such as POS tagging, chunking, and semantic labeling to automatically extract specifications from code comments, and checked source code against these specifications to detect software faults and bad comments; our previous work [14] automatically extracted resource specifications from API documents by leveraging the named entity recognition NLP technique; and our previous work [5] automatically extracted and validated code contracts from API documents by developing new NLP techniques such as noun boosting and equivalence analysis.

# 3. ACKNOWLEDGMENTS

This material is based upon work supported by the Maryland Procurement Office under Contract No. H98230-14-C-0141. This work is also supported in part by NSF grants CNS-1513690, CNS-1222680, CNS-1253346, CNS-1434582, CCF-1349666, CCF-1409423, and CCF-1434596, CNS-1513939, and a Google Faculty Research Award.

# 4. REFERENCES

- [1] V. Ambriola and V. Gervasi. Processing natural language requirements. In *Proceedings of the IEEE* International conference on Automated Software Engineering (ASE), 1997.
- [2] W. Enck and T. Xie. Tutorial: Text analytics for security. In Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security (CCS), 2014.
- [3] S. Garfinkel, G. Spafford, and A. Schwartz. Practical UNIX and Internet Security, 3rd Edition. O'Reilly Media, Feb. 2003.
- [4] R. Pandita, X. Xiao, W. Yang, W. Enck, and T. Xie. WHYPER: Towards automating risk assessment of mobile applications. In *Proceedings of the USENIX Security Symposium (USENIX Security)*, 2013.
- [5] R. Pandita, X. Xiao, H. Zhong, T. Xie, S. Oney, and A. Paradkar. Inferring method specifications from natural language API descriptions. In *Proceedings of* the International Conference on Software Engineering (ICSE), 2012.
- [6] D. Shepherd, Z. P. Fry, E. Hill, L. Pollock, and K. Vijay-Shanker. Using natural language program analysis to locate and understand action-oriented concerns. In *Proceedings of the International* Conference on Aspect-Oriented Software Development (AOSD), 2007.
- [7] J. Slankas, X. Xiao, L. Williams, and T. Xie. Relation extraction for inferring access control rules from natural language artifacts. In Proceedings of Annual Computer Security Applications Conference (ACSAC), 2014.
- [8] Stanford parser. http://nlp.stanford.edu/software/lex-parser.shtml.
- [9] L. Tan, D. Yuan, G. Krishna, and Y. Zhou. /\* iComment: Bugs or bad comments? \*/. In Proceedings of the ACM Symposium on Operating Systems Principles (SOSP), 2007.
- [10] Weka 3: Data Mining Software in Java. http://www.cs.waikato.ac.nz/ml/weka/.
- [11] WordNet. http://wordnet.princeton.edu/.
- [12] X. Xiao, A. Paradkar, S. Thummalapenta, and T. Xie. Automated extraction of security policies from natural-language software documents. In *Proceedings* of the ACM SIGSOFT International Symposium on the Foundations of Software Engineering (FSE), 2012.
- [13] T. Xie. Bibliography on text analytics for software engineering. https://sites.google.com/site/text4se/home/biblio.
- [14] H. Zhong, L. Zhang, T. Xie, and H. Mei. Inferring resource specifications from natural language API documentation. In *Proceedings of the IEEE* International conference on Automated Software Engineering (ASE), 2009.