Tutorial: Text Analytics for Security

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

Tao Xie
Department of Computer Science
University of Illinois at Urbana-Champaign
taoxie@illinois.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 will 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 will also highlight the current challenges of applying these techniques and tools for addressing security problems. We conclude 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. TEXT ANALYTICS FOR SECURITY

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 security is sometimes defined with respect to expectation: "A computer is secure if you can depend on it and its software to behave as you expect" [2, 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 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.

Our prior work is amongst several that have considered text analytics for security. In our WHYPER work [3], 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 [11]. In general, like other types of textual documents, 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, 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

CCS'14, November 3–7, 2014, Scottsdale, Arizona, USA.
ACM 978-1-4503-2957-6/14/11.
http://dx.doi.org/10.1145/2660267.2660576.

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. [6] 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 [12]. 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. [5] applied NLP techniques such as part-of-speech (POS) tagging to find word paraphrases to expand code search; Tan et al. [8] 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 [13] automatically extracted resource specifications from API documents by leveraging the named entity recognition NLP technique; and our previous work [4] automatically extracted and validated code contracts from API documents by developing new NLP techniques such as noun boosting and equivalence analysis.

In this tutorial, we will use our combined expertise in security and software engineering to present how text analytics can be applied to security. The tutorial will provide an introduction of popular techniques and tools of NLP and text mining such as WordNet [10], Stanford Parser [7], and Weka [9]. It will describe several success stores of applying NLP to security. Finally, we will discuss the current challenges of applying NLP and text mining techniques and tools for security problems, concluding with future research directions.

2. BIOGRAPHIES OF THE PRESENTERS

William Enck is an Assistant Professor in the Department of Computer Science at North Carolina State University. William's research efforts centrally focus on systems security, addressing challenges in smartphones and mobile applications, operating systems, cloud services, telecommunications, and hardware architectures. In particular, his work in mobile application security has led to significant consumer awareness and changes within the space. He earned his Ph.D., M.S., and B.S in Computer Science and Engineering from the Pennsylvania State University in 2011, 2006, and 2004, respectively.

Tao Xie is an Associate Professor in the Department of Computer Science at University of Illinois at Urbana-Champaign, USA. His research interests are in software engineering, focusing on software security, software testing, program analysis, and software analytics. He leads the Automated Software Engineering Research Group at Illinois, and is a member of the Programming Languages, Formal Methods, and Software Engineering (PL-FM-SE) area at Illinois. He received his Ph.D. in Computer Science from the University of Washington at Seattle in 2005. He co-presented a number of tutorials on mining software engineering data and software testing at major software engineering venues.

3. REFERENCES

[1] Ambriola, V., and Gervasi, V. Processing Natural Language Requirements. In *Proceedings of the IEEE International conference on Automated Software Engineering (ASE)* (1997).

[2] Garfinkel, S., Spafford, G., and Schwartz, A. *Practical UNIX and Internet Security, 3rd Edition*. O'Reilly Media, Feb. 2003.

[3] Pandita, R., Xiao, X., Yang, W., Enck, W., and Xie, T. WHYPER: Towards Automating Risk Assessment of Mobile Applications. In *Proceedings of the USENIX Security Symposium* (Aug. 2013).

[4] Pandita, R., Xiao, X., Zhong, H., Xie, T., Oney, S., and Paradkar, A. Inferring method specifications from natural language API descriptions. In *Proceedings of the International Conference on Software Engineering (ICSE)* (2012).

[5] Shepherd, D., Fry, Z. P., Hill, E., Pollock, L., and Vijay-Shanker, K. 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).

[6] Slankas, J., Xiao, X., Williams, L., and Xie, T. Relation extraction for inferring access control rules from natural language artifacts. In *Proceedings of Annual Computer Security Applications Conference (ACSAC)* (2014).

[7] Stanford parser. `http://nlp.stanford.edu/software/lex-parser.shtml`.

[8] Tan, L., Yuan, D., Krishna, G., and Zhou, Y. /* iComment: Bugs or bad comments? */. In *Proceedings of the ACM Symposium on Operating Systems Principles (SOSP)* (2007).

[9] Weka 3: Data Mining Software in Java. `http://www.cs.waikato.ac.nz/ml/weka/`.

[10] Wordnet. `http://wordnet.princeton.edu/`.

[11] Xiao, X., Paradkar, A., Thummalapenta, S., and Xie, T. 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).

[12] Xie, T. Bibliography on text analytics for software engineering. `https://sites.google.com/site/text4se/home/biblio`.

[13] Zhong, H., Zhang, L., Xie, T., and Mei, H. Inferring Resource Specifications from Natural Language API Documentation. In *Proceedings of the IEEE International conference on Automated Software Engineering (ASE)* (2009).