Pragmatic Characteristics of Security Conversations
An Exploratory Linguistic Analysis

Benjamin S. Meyers, Nuthan Munaiah, Andrew Meneely
Department of Software Engineering
Rochester Institute of Technology
Rochester, NY
{bsm9339,nm6061,axmvse}@rit.edu

Emily Prud'hommeaux
Department of Computer Science
Boston College
Chestnut Hill, MA
prudhome@bc.edu

***Abstract*—Experts suggest that engineering secure software requires a defensive mindset to be ingrained in developer culture, which could be reflected in conversation. But what does a conversation about software security in a real project look like? Linguists analyze a wide array of characteristics: lexical, syntactic, semantic, and pragmatic. *Pragmatics* focus on identifying the style and tone of the author's language. If security requires a different mindset, then perhaps this would be reflected in the conversations' pragmatics. Our goal is to characterize the pragmatic features of conversations about security so that developers can be more informed about communication strategies regarding security concerns. We collected and annotated a corpus of conversations from 415,041 bug reports in the Chromium project. We examined five linguistic metrics related to pragmatics: formality, informativeness, implicature, politeness, and uncertainty. Our initial exploration into these data show that pragmatics plays a role, however small, in security conversations. These results indicate that the area of linguistic analysis shows promise in automatically identifying effective security communication strategies.**

***Index Terms*—software engineering, security, discourse, natural language processing**

I. Introduction

As developers engineer software, they communicate and collaborate with one another in a variety of ways to discuss design, code, and bugs. These communications, particularly those in emails, chat conversations, bug reports, and code reviews, provide a wealth of linguistic data with which one can gain valuable insights into developers' use of natural language. In the past, researchers have used natural language metrics to characterize various linguistic features of code review feedback, including being acted-upon [1] and usefulness [2], [3]. However, the *pragmatics* (i.e. style and tone of natural language [4], [5]) of developers' use of natural language has largely been unexplored.

By analyzing linguistic pragmatics, we can understand not just what developers said, but how they conveyed it. Our goal in this work is to *characterize the pragmatic features of conversations about security so that developers can be more informed about communication strategies regarding security concerns.* We address the research question:

RQ 1: Linguistic Pragmatics of Security
Does developers' language exhibit different pragmatic characteristics when discussing security?

II. Methodology

We collected a corpus of security-related conversations from bug report discussions about the Chromium project. We then used tags associated with bug reports to categorize bugs, and their comments, as *security* and *non-security*. We extracted metrics to quantify some pragmatic features and used statistical methods to determine whether the two classes of bug comments differ in their values of these metrics. We describe our methodology in greater detail in the following paragraphs.

Data Collection: The corpus of natural language used in this work was curated through the retrieval and annotation of comments posted to bug conversations in the Chromium project. The Chromium project uses Monorail, an open-source issue tracking system built by Google, to manage bugs. We used the Monorail RESTful API to programmatically download bug reports as JSON-formatted documents. As of July 11, 2017, we had downloaded 435,822 bugs spanning nine years (2008–2016) and totaling 3.9 GB.

Data Annotation: In the Chromium project, bugs are associated with one or more tags. We used these tags to automatically categorize the bugs into two relevant groups: *security* and *non-security*. In our corpus, there were 16,165 unique tags, of which 53 were determined to be related to security. The three most frequent security-related tags were `Type-Bug-Security`, `Security_Impact-Stable`, and `Security_Severity-High`. The comments associated with any bug tagged with one or more of these tags was marked as *security*. All other bugs were considered to be *non-security*.

Feature Extraction: We collected five metrics from the natural language in our corpus of bug comments. We introduce these metrics in Section III. The metrics are defined at either the sentence- or token-level. Therefore, we split the bug comments into sentences and the sentences into tokens (words) all while retaining the annotation. We persisted all data (bugs, comments, sentences, tokens, and the metrics) to a PostgreSQL database with an aggregate size of 62GB. Shown in Table I is a summary of the corpus curated in our study.

Statistical Analysis: We use association analysis to assess if the pragmatic features, quantified by their respective metrics, of natural language in bug comments are, in fact, associated with *security* or *non-security*. For numerical-valued metrics, we used the Mann-Whitney-Wilcoxon test to assess if the distribution of a metric is similar across security and non-security language. We regard the outcome from Mann-Whitney-Wilcoxon test to be statistically significant if p-value < 0.05. We further used Cliff's δ to assess the strength of association (if any). For categorical metrics, we used the χ^2 test to assess if a metric is independent of bug type. We regard the outcome from χ^2 test to be statistically significant if p-value < 0.05. We further used Pearson's φ to assess the strength of dependence (if any).

Summary: A summary of the corpus used in our study is shown in Table I. We persisted all data (bugs, comments, sentences, tokens, and the metrics) to a PostgreSQL database with an aggregate size of 62GB. We have released this corpus [6] to encourage further exploration of pragmatic characteristics of security conversations, as well as lexical, syntactic, and semantic characteristics.

TABLE I: Summary of the corpus used in our study.

Entity	# Security (%) / Total
Bugs	11,236 (2.7%) / 415,041
Comments	88,750 (4.2%) / 2,135,767
Sentences	309,564 (4.1%) / 7,574,215
Tokens	7,431,996 (4.9%) / 152,561,733

III. Linguistic Exploration

In this section, we discuss how developers talk about security in terms of the pragmatic linguistic characteristics considered in our study: formality, informativeness, implicature, politeness, and uncertainty. We provide a motivation and a refined research question from **RQ 1** for each metric and its analysis. Formality, informativeness, implicature, and politeness all have continuous scale from 0 to 1, whereas uncertainty has a discrete scale with true and false being the permitted values. In motivating each pragmatic feature, we provide examples of security-related ((S_{hi}) and (S_{lo})) and non-security-related ((NS_{hi}) and (NS_{lo})) with high and low values for the corresponding feature, respectively.

Previous work has applied sentiment analysis to developer conversations [1]–[3], [7]–[9], however, sentiment is a semantic characteristic of natural language; we limit our discussion to pragmatic characteristics of language.

A. Formality

For Chromium, discovery of vulnerabilities is rewarded with monetary bounties [10], so discussion between engineers and researchers is common. Conveying the complex nuance of security concerns to people of other backgrounds may require more conversational etiquette. Alternatively, perhaps the urgency with which security conversations need to be held (e.g. in a zero-day situation) might result in forgoing typical conversational etiquette. The **formality** of a sentence, as discussed in Lahiri [11] and Heylighten *et al.* [12], is the observance of etiquette in forming a sentence.

(S_{hi})	*"The hack attacks firmware, it subverts the TPM by persistent callbacks to 0x0 address, and it is therefore persistent under the guest and other user accounts."* (0.998)
(S_{lo})	*"I think, perhaps, I've got too much stuff to sync, so it's just stopping at some point."* (6.0e-6)
(NS_{hi})	*"However, given the sensitive nature of this detection technology, it may occasionally identify non-malicious, legitimate software programs that also share these behavioral characteristics."* (0.999)
(NS_{lo})	*"If they remove every little feature we like about Chrome (*cough* single-click selects all *cough*) and just say, 'Use an extension', I might as well go back to using Firefox, or hell, even IE9, as it's looking pretty promising."* (2.5e-7)

Our previous work has shown that formality plays a large role in whether or not a code review comment is acted-upon [1]. We use the classification method from that study to measure formality.

Thus, our research question for formality is:

RQ 1.1: Is security conversation more or less **formal** than non-security conversation?

B. Informativeness

When developers communicate over bug reports, they are examining highly detailed technical problems. A key step in these conversations is the ability to arrive at a mutual agreement. When security is involved, the stakes are high and the possible scenarios become even more complex. The metric **informativeness**, rooted in the cooperative principles of Grice's pragmatic theory [13], is one measure which captures language that leads to mutual agreement. Informative language is presented clearly, directly, and without ambiguity [11], [12]. The following examples exhibit varying levels of informativeness.

(S_{hi})	*"Alternatively attackers could impersonate the 802.1x networks a user's device is configured to authenticate against automatically, in order to observe responses and subsequently perform offline brute-force attacks on the user's password, or to pass values through authenticating the attacker to the 802.1x protected network."* (1.000)
(S_{lo})	*"Any thoughts on the best way to solve this?"* (5.3e-5)
(NS_{hi})	*"The binary doesn't actually work right (the bundle isn't completely right), but dropping the executable itself into a GYP build gets reasonably close to working (some tests pass, some crash) on the simulator."* (0.999)
(NS_{lo})	*"Please help, I don't understand any of this!?!"* (1.4e-5)

We use a classifier built on the manually annotated corpus from Lahiri [11] to compute informativeness scores. Further details on the classifier can be found in our previous work [1].

Thus, our research question for informativeness is:

RQ 1.2: Does security conversation exhibit more or less **informativeness** than non-security conversation?

C. Implicature

Conveying context is a significant part of any conversation. In a bug report, developers must consider all kinds of context, such as the system design, legacy systems, and the development process. Discussion that lacks proper context will be less clear, leading to misunderstandings, which is true of both security and non-security conversations. **Implicature**, like informativeness, is also rooted in Grice's maxims [13], and is a measure of how much context of a sentence is missing [12], [14]. We used a corpus with manually annotated implicature scores [11] to build a classifier that computes implicature, where scores are continuous from 0–1, with 1 meaning the sentence is missing context and 0 meaning the sentence is not missing context.

(S_{hi}) *"I don't immediately see how my patch would've caused that use-after-free, but I guess it's possible."* (0.996)

(S_{lo}) *"The first incognito tab opened, for each Chrome profile, creates a temporary in-memory profile and every subsequent tab opened will share the same cookie jar - so the behavior you describe in the second paragraph of #3 is working as intended."* (3.8e-5)

(NS_{hi}) *"It seems that something more concise or more obvious would be better."* (0.999)

(NS_{lo}) *"Another issue I'm seeing is that with Chrome when I load a new page or switch tabs the content doesn't display inside the view port until something dynamic happens, like the page scrolls, I move my mouse over something, the content moves, etc."* (3.3e-7)

In (S_{hi}), the author references a patch that they submitted, but little else; this sentence is missing a large portion of context to understand exactly what is being said. Conversely, in (S_{lo}) the author gives a step-by-step description of the sequence of events leading to the behavior in question and further pinpoints exactly where that behavior is described; there is minimal extra context required to understand the comment.

Thus, our research question for implicature is:

RQ 1.3: Does security conversation exhibit more or less **implicature** than non-security conversation?

D. Politeness

The urgency of zero-day security bugs, or the stakes being high, may lead to politeness being ignored. Alternatively, politeness may be needed in this professional environment to convince developers to take security seriously. We utilized a corpus of over 10,000 sentences which were manually annotated for **politeness** and the corresponding classification model to measure politeness within our bug report corpus [15]. The following sentences show developers being polite and impolite.

(S_{hi}) *"I would appreciate if you could try this out and see if you can find a place where this still breaks."* (0.785)

(S_{lo}) *"A bug for sure but probably not worth the effort to fix, given that it's not a security issue, and that it's not something that well-intentioned PDFs will accidentally hit, so not really a stability issue either."* (0.155)

(NS_{hi}) *"Hi, thanks for your report, I think this is working as intended and Firefox 3.5, IE7, IE8 behaves the same as Chrome, Jungshik, could you please confirm?"* (0.947)

(NS_{lo}) *"Why don't you simply accept this fact?"* (0.088)

Consider sentences (S_{hi}) and (S_{lo}) above. In the former, the writer is polite when asking their peer(s) to look into a piece of buggy code. In the latter, the writer is not being polite; they are simply disputing any potential concerns brought up earlier in the conversation.

Thus, our research question for politeness is:

RQ 1.4: Does security conversation exhibit more or less **politeness** than non-security conversation?

E. Uncertainty

Uncertainty, according to Vincze [16], manifests itself as the lack of information required to confirm whether a proposition is true. Vincze further categorized uncertainty as:

- Epistemic: The proposition is possible, based on our existing knowledge of the universe, but its truth-value cannot be determined.
- Doxastic: The proposition is assumed to be true or false, but its truth-value cannot be determined.
- Investigative: The proposition is in the process of having its truth-value determined.
- Conditional: The proposition is true or false based on the truth-value of another proposition.

Consider the examples of sentences from security (S) and non-security (NS) bugs shown below to understand the subtle differences between these four types of semantic uncertainty. The subscript indicates one of the aforementioned types of semantic uncertainty.

(S_E) *"I expect that the advertiser account which owns these Floodlights belongs to someone at Google, but it's still quite troubling."*

(S_D) *"If we fix it to only work over HTTPS and against whitelisted hosts for PPAPI (for ChromeOS) would that alleviate the need to fix all permutations of crash-from-bad-scripts as demonstrated here?"*

(S_I) *"Well, I haven't installed Java to test, but I know our plugin loader can never call CreateProcess for a third-party plugin (and as I explained LoadLibrary cannot exhibit this behavior)."*

(S_C) *"If we fix it to only work over HTTPS and against whitelisted hosts for PPAPI (for ChromeOS) would that alleviate the need to fix all permutations of crash-from-bad-scripts as demonstrated here?"*

(NS_E) *"That's really awkward...and seems like it should be fixed before we declare Yosemite layout tests to be fixed."*

(NS_D) *"I think we've just had this wrong for years."*

(NS_I) *"The trace already shows whether hitTest was called from a mouse event or javascript."*

(NS_C) *"Presently if the screen is locked, we allow it to dim, screen off, and suspend, even if the screensaver is a motion video."*

We used the corpus from Farkas *et al.* [17] and a re-implementation of the classifier from Vincze [16], which utilized characteristics of semantic uncertainty from Szarvas *et al.* [18], to detect uncertainty in bug conversations. Further implementation details can be found in our previous work [1].

Thus, our research question for uncertainty is:

RQ 1.5: Is security conversation more or less likely to express **uncertainty** than non-security conversation?

IV. Results from Exploratory Analysis

To answer **RQ 1** we assessed association using the Mann–Whitney–Wilcoxon (MWW) test to compare formality, informativeness, implicature, and politeness for security and non-security bug comments. As reported in Table II, all of our continuous-valued metrics are statistically significant with negligible effect sizes, meaning that while there is a statistically significant difference in the distribution of formality, informativeness, implicature, and politeness between security and non-security bugs, the magnitude of difference is negligible. A visual comparison of the distributions of these metrics showed

TABLE II: MWW test results for continuous metrics

Metric	**Outcome**
Minimum Formality	S < NS*** (N)
Maximum Formality	S < NS*** (N)
Maximum Informativeness	S < NS*** (N)
Maximum Implicature	S > NS*** (N)
Minimum Politeness	S > NS*** (N)
Maximum Politeness	S > NS*** (N)

Legend *** $p < 0.001$; (N) $\delta < 0.147$

TABLE III: χ^2 test results for uncertainty

Uncertainty	**% Comments**	
	Security	**Non-Security**
Epistemic***	0.87%	18.22%
Doxastic***	0.45%	8.99%
Investigative***	0.12%	2.53%
Conditional***	0.82%	18.55%
Uncertain***	1.58%	35.56%

Legend *** $p < 0.001$

that the distributions are nearly identical for security and non-security conversations, which is in agreement with Cliff's δ.

We also examined the independence of uncertainty types for security and non-security bug comments using the χ^2 test. We found that all four types of uncertainty—as well as the aggregation of the four, indicating the presence of any uncertainty—were statistically significant and not independent (i.e. associated) (see Table III).

The MWW and the χ^2 tests indicate that all of the metrics are associated with security conversations at a statistically significant level. However, the negligible effect sizes reveal that when considered individually, the metrics are not *strongly* associated with security conversations.

In summary, our results indicate:

RQ 1.1: *Developers are* ***less formal*** *in security conversations than in non-security conversations.*
RQ 1.2: *Security conversations have a* ***lower maximum informativeness*** *than non-security conversations.*
RQ 1.3: *Security conversations are* ***more implicative*** *than non-security conversations.*
RQ 1.4: *Developers are* ***more polite*** *in security conversations than in non-security conversations.*
RQ 1.5: *Security conversations are* ***less likely to exhibit uncertainty*** *than non-security conversations, regardless of the type of uncertainty.*

We found that, while discussing security, developers tend to be less formal and less informative, while being certain, polite, and implicative. We see the less formal and more polite nature of security conversations to encourage developers with less experience to express their thoughts. High implicature (*i.e.* lack of sufficient context) might suggest that security, being nuanced and multifaceted, is difficult to discuss with enough relevant detail to ensure mutual understanding between developers.

V. Summary & Future Work

By understanding the pragmatic characteristics of security conversations we hope to provide developers with insights and tactics to pull security to the forefront of their conversations. We found that both security and non-security conversations in bug reports exhibit varying degrees of formality, informativeness, implicature, politeness, and uncertainty. The differences were small, but statistically significant. While the pragmatic characteristics explored in this work are not significant discriminators between security and non-security conversations, the results of this study warrant further examination into what we believe is an information-rich corpus. We have released the corpus used in this work to encourage further linguistic analysis.

References

[1] B. S. Meyers, N. Munaiah, E. Prud'hommeaux, A. Meneely, C. Alm, J. Wolff, and P. K. Murukannaiah, "A dataset for identifying actionable feedback in collaborative software development," in *Meeting for the Assn. for Computational Linguistics*, Melbourne, Australia, 2018.

[2] M. Rahman, C. K. Roy, and R. Kula, "Predicting Usefulness of Code Review Comments Using Textual Features and Developer Experience," in *Int'l Conf. on Mining Software Repositories*, ser. MSR '17. Piscataway, NJ, USA: IEEE Press, 2017, pp. 215–226. [Online]. Available: https://doi.org/10.1109/MSR.2017.17

[3] A. Bosu, M. Greiler, and C. Bird, "Characteristics of Useful Code Reviews: An Empirical Study at Microsoft," in *Int'l Conf. on Mining Software Repositories*, may 2015, pp. 146–156.

[4] H. Dawson and M. Phelan, Eds., *Language Files: Materials for an Introduction to Language and Linguistics*. Ohio State U. Press, 2016.

[5] K. Denham and A. Lobeck, *Linguistics for Everyone: An Introduction*. Cengage Learning, 2013.

[6] B. S. Meyers, N. Munaiah, A. Meneely, and E. Prud'hommeaux, "Security bug conversations," March 2019. [Online]. Available: https://doi.org/10.5281/zenodo.2595071

[7] E. Guzman, D. Azócar, and Y. Li, "Sentiment Analysis of Commit Comments in GitHub: An Empirical Study," in *Int'l Conf. on Mining Software Repositories*. New York, NY: ACM, 2014, pp. 352–355.

[8] D. Pletea, B. Vasilescu, and A. Serebrenik, "Security and Emotion: Sentiment Analysis of Security Discussions on GitHub," in *Int'l Conf. on Mining Software Repositories*. NY, NY: ACM, 2014, pp. 348–351.

[9] N. Munaiah, B. S. Meyers, C. Alm, A. Meneely, P. K. Murukannaiah, E. Prud'hommeaux, J. Wolff, and Y. Yu, "Natural language insights from code reviews that missed a vulnerability," in *Int'l Symposium on Engineering Secure Software and Systems*. Bonn, Germany: Springer, August 2017, pp. 70–86.

[10] Google, "Chrome Rewards - Application Security - Google," https://www.google.com/about/appsecurity/chrome-rewards/index.html, [Online] Accessed: 02-07-2019.

[11] S. Lahiri, "SQUINKY! A Corpus of Sentence-level Formality, Informativeness, and Implicature," *CoRR*, vol. abs/1506.02306, 2015.

[12] F. Heylighen and J. Dewaele, "Formality of language: definition, measurement and behavioral determinants," 1999.

[13] H. Grice, P. Cole, J. Morgan *et al.*, "Logic and conversation," *Syntax and semantics*, pp. 41–58, 1975.

[14] C. Potts, *The logic of conventional implicatures*. Oxford University Press on Demand, 2005, no. 7.

[15] C. Danescu-Niculescu-Mizil, M. Sudhof, D. Jurafsky, J. Leskovec, and C. Potts, "A computational approach to politeness with application to social factors," 2013.

[16] V. Vincze, "Uncertainty Detection in Natural Language Texts," Ph.D. dissertation, University of Szeged, 2014.

[17] R. Farkas, V. Vincze, G. Móra, J. Csirik, and G. Szarvas, "The conll-2010 shared task: learning to detect hedges and their scope in natural language text," in *Conf. on Computational Natural Language Learning—Shared Task*. Assn. for Computational Linguistics, 2010, pp. 1–12.

[18] G. Szarvas, V. Vincze, R. Farkas, G. Móra, and I. Gurevych, "Cross-genre and cross-domain detection of semantic uncertainty," *Computational Linguistics*, vol. 38, no. 2, pp. 335–367, 2012.