Pragmatic Characteristics of Security Conversations

An Exploratory Linguistic Analysis

ABSTRACT

Modern software engineers are faced with a tough challenge: collaboratively develop software while keeping it secure. 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* focuses 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 play 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.

ACM Reference Format:

. 2019. Pragmatic Characteristics of Security Conversations: An Exploratory Linguistic Analysis. In *CHASE '19: International Workshop on Cooperative and Human Aspects of Software Engineering, May 25–31, 2019, Montréal, QC, Canada.* ACM, New York, NY, USA, 8 pages.

1 INTRODUCTION

The art of engineering software is an inherently collaborative, and thus a social, practice. As developers engineer software, they use a variety of ways to communicate with one another about design, code, bugs, etc. These communications—particularly those written in email, chat, bug reports, and code reviews—provide a wealth of linguistic data with which one can gain valuable insights into developers' use of natural language. Among these insights are *pragmatics*, which seek to identify the author's style and tone beyond the lexical and semantic content [3, 4]. In the past, researchers have used natural language to characterize various linguistic features of code review feedback, including being acted-upon [7], and usefulness [1, 15].

Security, however, is different. In a world where attackers have lucrative incentives [19] to discover and exploit vulnerabilities in software, developers must balance the needs of delivering their products with the ever-growing need to keep their customers secure. Experts suggest that engineering secure software requires developers to "think like an attacker" [12], and that the mindset must be ingrained in engineering culture. This mindset could be reflected in conversation. From our informal explorations, conversations about security tend to take place in bug reports, and involve some nuanced discourse with varied style and tone (`https://goo.gl/mmPyYY`, for instance). Furthermore, in our prior work, we found several interesting linguistic characteristics of code reviews that have missed a vulnerability [6], some of which were pragmatic in nature.

We want developers to have more conversations that resemble the discussions they tend to have when discussing security bugs. By analyzing linguistic pragmatics, we can dig into not just what people 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 following research questions to accomplish this goal:

RQ 1 Characterizing Security Talk
Do developers use pragmatic characteristics of language differently when discussing security?

RQ 2 Differentiating Security Talk
How well do pragmatic characteristics of language differentiate between security and non-security conversation?

The main contributions of this work are:

- an open data set of bug conversations from the Chromium project with 415,041 bug reports and 152,561,733 tokens (words) for future analysis[1], and
- an initial exploratory analysis into pragmatic features of Chromium bug conversations.

The remainder of this paper is structured as follows: we begin with a brief discussion of our process for collecting and annotating the corpus of bug conversations used in the empirical analysis (Section 2). Then we describe the overall methodology followed in collecting and empirically analyzing the five metrics considered in our study. In Section 3, we use exemplary sentences from actual bugs in our corpus to elaborate and motivate the metrics while addressing our first research question. We address the second research question in Section 4.2 and conclude the paper with a summary of our findings in Section 5 followed by a description of potential limitations in our work in Section 6. Finally, we include an appendix containing additional descriptive metrics and exemplary sentences.

2 METHODOLOGY

With the goal of characterizing the linguistic features of security-oriented conversations in mind, we collected a corpus of bug report conversations from the Chromium project. After collecting the data

[1] Due to the double-blind review process, we cannot include a link to the corpus at this point in time. Should our paper be accepted, we will link to the corpus.

CHASE '19, May 25–31, 2019, Montréal, QC, Canada

we used developer-curated tags to categorize bugs and their comments as *security* or *non-security*. To begin our exploratory analysis we examined the most frequent lemmas (root words) in each population and performed 10x10-fold cross-validation for a simple n-gram model. Next, we used our domain knowledge in the fields of software development, cybersecurity, and natural language processing to refine our research questions and identify pragmatic phenomena that we believe to be prevalent in security conversations: formality, informativeness, implicature, politeness, and uncertainty. Finally, we computed the metrics, ran statistical analyses, and compared the metrics with respect to security and non-security conversations.

We further describe each step of our methodology in the following subsections.

2.1 Data Collection

The corpus of security language presented as part of 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 followed a two step approach to retrieve the comments from bug conversations in the Chromium project. We began by retrieving the identifiers of all bugs reported for Chromium using the web interface to Chromium Monorail. We then used the Monorail RESTful API to programmatically download the bug data as a JSON-formatted document. As of July 11, 2017, we had downloaded 435,822 bugs spanning nine years (2008–2016) and totaling 3.9 GB.

2.2 Data Annotation

We annotated the bug conversations to separate them into two relevant groups: *security* and *non-security*. We used tags that are associated with each bug to achieve the annotation. In total, there are 16,165 unique tags of which 53 are related to security. The three most common of these security-related tags are `Type-Bug-Security`, `Security_Impact-Stable`, and `Security_Severity-High`. We used the security-related tags to identify security bugs and categorized all associated comments as *security*. We categorized comments associated with all other bugs as *non-security*.

The five metrics that we collected are defined at either the sentence- or token-level. We split the bug comments into sentences and the sentences further into tokens (words) all while propagating the appropriate annotation downstream. We persisted all data (bugs, comments, sentences, tokens, and the metrics) to a PostgreSQL database with an aggregate size of 62GB. Shown in Table 1 is a summary of the corpus curated in our study.

Table 1: A summary of the corpus curated 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

2.3 Statistical Analysis

Our main technique for comparing two populations is *association* as defined in the metrics validation literature [13, 16]. We conclude that a metric is associated with security conversations by first computing the medians for both security and non-security conversations and comparing them. We then conduct a one-sided Mann-Whitney-Wilcoxon test based on which population was higher, and compare the p-value against 0.05. We use Mann-Whitney-Wilcoxon in this case so that we do not have an assumption of normality for our metrics. This gives us an initial result of whether or not the metric is associated. We further evaluate the strength of that association with Cliff's δ. For metrics that follow a Bernoulli distribution (boolean-valued), the χ^2 test serves as an analog for Mann-Whitney-Wilcoxon—it does not rely on the assumption of normality and it gives an initial indication of whether or not the metric is independent with respect to security and non-security.

For **RQ 2**, we used a logistic regression model to determine how well the five linguistic metrics, together, explain the differences between security and non-security. This analysis is intended to be a baseline evaluation, with little model performance optimization.

3 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.

3.1 Word Frequencies Are Not Enough

Previous works have utilized word frequencies in their study of developer conversations [1, 15]. Intuitively, the words used to discuss a given subject matter (lexical content) should be representative of the topic. We expect developers to use security-related jargon when talking about security. To examine lexical content we order lemmas (root words) in our corpus based on their frequencies, from most frequent to least. We only consider lemmas that are content words (verbs, nouns, adjectives, adverbs) and are not in NLTK's list [2] of English stop-words. We also disregard lemmas that appear in the top twenty in both security (`S`) and non-security (`NS`) bugs. Below we report the top twenty lemmas, with *(v)* and *(n)* denoting verbs and nouns, respectively.

- (`S`) webcore, v8, security, blink, crash (`v`), void, bool, unsigned, stack *(n)*, fix *(n)*, report *(n)*, system, function *(n)*, char, chrome.dll, size, sure, script, const*, make *(v)*
- (`NS`) windows, expect, google, step, build, tab, happen, work, line, result, behavior, flash, window, information/info, provide, official, mac, able, change *(n)*, change *(n)*

Interpreting these top lemmas from security and non-security sentences, we see that conversations in security bugs often mention very specific technologies such as *webcore* (the subsystem of Chromium containing logic for page layout), *v8* (Chromium's JavaScript engine), and *blink* (Chromium's rendering engine), whereas,

[2] `https://www.nltk.org/nltk_data/`

conversations in non-security bugs tend to contain high-level technologies (*mac*, *google*, *windows*) without discussing specific subsystems or software (with the exception of *flash*). We also observed that security bugs reference specific concepts related to programming (*bool*, *stack*, *void*, *unsigned*, *char*, *const**).

These top lemmas indicate that there is a difference between the language in security and non-security bugs. To understand if this difference is substantial enough to classify bugs as security and non-security, we performed a simple classification experiment using ngrams—unigrams, bigrams, and trigrams (sequences of one, two, and three words, respectively). The performance of the logistic regression model was not particularly impressive, with precision, recall, and f-measure of 90.01%, 14.81%, and 25.41%, respectively. The performance of the model indicates that it successfully classified security conversations (high precision) but only managed to retrieve a small portion of all security conversations (low recall). We suspect the sub par performance of the model to be an artifact of the imbalance in the dataset (few security bugs relative to non-security bugs). While we could have used a resampling technique such as SMOTE to balance the dataset, we chose to continue our exploration of characteristics that can provide more nuanced insights into the conversation than the mere frequency of words that indicate security.

3.2 Formality

When developers have conversations on bug reports, an unspoken etiquette is in place. The environment is a professional one, and developers may be interacting with strangers. In bug reports, participants may not even be other developers but users with varying levels of technical expertise and project familiarity. In the case of Chromium, discovering vulnerabilities are rewarded with monetary bounties [8], so discussion between engineers and researchers is common. In all of these situations, conveying the complex nuance to people of other backgrounds may require higher 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.

With regards to pragmatics, the **formality** of a sentence is a measure of the observance of etiquette in forming a sentence. Our measure of formality is based on Grice's Maxims of Quality and Relevance [9–11]. The sentences below are examples from the bug corpus. Sentences (S_{hi}) and (S_{lo}) come from conversations about security and represent formal and informal language, respectively. The same is true for (NS_{hi}) and (NS_{lo}) except that these sentences are from non-security conversations.

(S_{hi})	0.998	*"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."*
(S_{lo})	6.0e-6	*"I think, perhaps, I've got too much stuff to sync, so it's just stopping at some point."*
(NS_{hi})	0.999	*"However, given the sensitive nature of this detection technology, it may occasionally identify non-malicious, legitimate software programs that also share these behavioral characteristics."*
(NS_{lo})	2.5e-7	*"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."*

Our previous work has shown that formality plays a large role in whether or not a code review comment is acted-upon [7]. 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?

3.3 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 [9], is one measure that captures language that leads to mutual agreement. Informative language is presented clearly, directly, and without ambiguity [10, 11]. The following examples exhibit varying levels of informativeness.

(S_{hi})	1.000	*"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."*
(S_{lo})	5.3e-5	*"Any thoughts on the best way to solve this?"*
(NS_{hi})	0.999	*"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."*
(NS_{lo})	1.4e-5	*"Please help, I don't understand any of this!?!"*

We use a classifier built on the manually annotated corpus from Lahiri [11] to compute informativeness scores (on a continuous scale from 0 to 1) for each sentence in our corpus. Further details on the classifier can be found in our previous work [7]. We define informativeness at the comment-level as the maximum informativeness exhibited by the sentences in the comment.

Thus, our research question for informativeness is:

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

3.4 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 and understood, which is true of both security and non-security conversations. **Implicature**, like formality and informativeness, is also rooted in

Grice's maxims [9], and is a measure of how much context of a sentence is missing. The precise meaning of implicature is disputed among linguists [10, 14], but we used a corpus with manually annotated implicature scores and use the interpretation of implicature in that study [11]. In the examples below, higher implicature means the sentence is missing more context.

(S_{hi})	0.996	*"I don't immediately see how my patch would've caused that use-after-free, but I guess it's possible."*
(S_{lo})	3.8e-5	*"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."*
(NS_{hi})	0.999	*"It seems that something more concise or more obvious would be better."*
(NS_{lo})	3.3e-7	*"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."*

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.

As with formality and informativeness, we trained a classification model on the annotated dataset from Lahiri [11] and used it to compute implicature scores at the sentence-level. We consider the maximum implicature score of the sentences in a comment to be the implicature score of the comment.

Thus, our research question for implicature is:

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

3.5 Politeness

When software engineers are polite to each other, they may be more likely to accept feedback on a bug. 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. **Politeness** is a representation of mutual respect for one another in a conversation. 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 [2].

The following sentences show developers being polite and impolite.

(S_{hi})	0.785	*"I would appreciate if you could try this out and see if you can find a place where this still breaks."*
(S_{lo})	0.155	*"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."*
(NS_{hi})	0.947	*"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?"*
(NS_{lo})	0.088	*"Why don't you simply accept this fact?"*

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.

Politeness is defined at the sentence-level, but we aggregate to the comment-level by taking the sentences with the lowest and highest scores to be the minimum and maximum politeness of a comment, respectively.

Thus, our research question for politeness is:

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

3.6 Uncertainty

As developers collaborate to uncover the source of a software bug, uncertainty in their communication may be a factor that could inhibit the timely resolution of the bug. In particular, uncertainty in developers' discourse may lead them to misinterpret requirements, design decisions, peer feedback, or the likelihood of a security flaw. **Uncertainty**, according to Vincze [18], manifests itself as the lack of information in language that prevents it from being objectively understood by the intended audience. A proposition is considered uncertain if the truth-value (or reliability) of the statement cannot be decided due to information that is missing. Vincze further categorized uncertainty in natural language to be one of four semantic types: epistemic, doxastic, investigative, and conditional. While the concept of uncertainty may be intuitive to understand, there are subtle differences between the four types of semantic uncertainty as can be gleaned from their descriptions that follow.

- 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.

Security

(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?"*

Non-Security

(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.* [5] and a recreation of the classifier from Vincze [18], which utilized characteristics of semantic uncertainty from Szarvas *et al.* [17], to detect uncertainty in bug conversations.

Uncertainty, as identified by the classifier proposed by Vincze [18], is at the level of individual tokens (i.e. words) in a sentence. Vincze inferred a sentence as being uncertain if at least one of its tokens was labeled as uncertain. We followed a similar approach by aggregating the token-level uncertainty labels at the sentence level and then at the comment level. For instance, if a comment has four sentences and one of the sentence has a token labeled as doxastically uncertain, we label that sentence, and the comment itself, as doxastically uncertain. We also labeled a comment as being uncertain, irrespective of the type, if it had at least one uncertain sentence of any type.

Thus, our research question for uncertainty is:

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

Summary

As a summary of the theories we discussed in this section, we present the following formal research questions:

> **RQ 1.1:** Is security conversation more or less **formal** than non-security conversation?
> **RQ 1.2:** Does security conversation exhibit more or less **informativeness** than non-security conversation?
> **RQ 1.3:** Does security conversation exhibit more or less **implicature** than non-security conversation?
> **RQ 1.4:** Does security conversation exhibit more or less **politeness** than non-security conversation?
> **RQ 1.5:** Is security conversation more or less likely to express **uncertainty** than non-security conversation?

4 RESULTS FROM EXPLORATORY ANALYSIS

In the following subsections we provide answers to our research questions.

4.1 RQ 1: Characterizing Security Talk

> **RQ 1:** Do developers use pragmatic characteristics of language differently when discussing security?

To answer **RQ 1** we assessed association using the Mann–Whitney–Wilcoxon test to compare formality, informativeness, implicature, and politeness for security and non-security bug comments. As reported in Table 2, all of our continuous-valued metrics are significant with negligible effect sizes, meaning that while there is a statistically significant difference in the means of formality, informativeness, implicature, and politeness, the magnitude of those differences is minimal.

Table 2: Mann–Whitney–Wilcoxon test results from comparing formality, informativeness, implicature, and politeness for security and non-security bug comments.

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) $\delta < 0.147$

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 3). Thus, all of the metrics are associated with security conversations at a statistically significant level. However, the effect size is considered to be "neglible" according to Cliff's δ, indicating that, when taken individually, the metrics are not *strongly* associated with security conversations.

Shown in Figure 1 is a comparison of the distributions of formality, informativeness, implicature, and politeness collected from security and non-security bug comments. As seen, the distributions are nearly identical for security and non-security conversations, which is in agreement with Cliff's δ. Table 4 in the Appendix contains means (with standard deviations), medians, and first and third quartiles for our linguistic metrics.

The answers to the five questions posed in **RQ 1** are as follows:

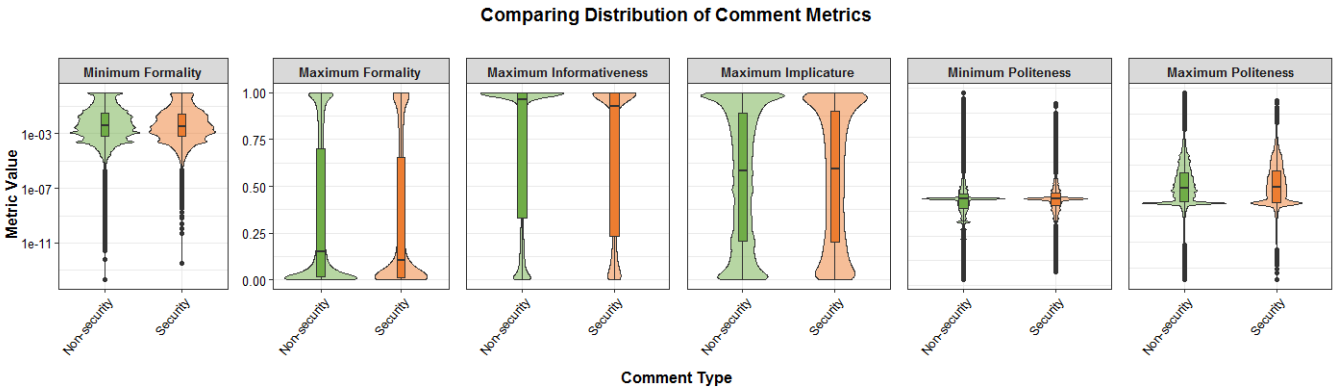

Figure 1: Comparison of distribution of the four linguistic metrics collected from natural language in security and non-security bug comments

Table 3: χ^2 test results when testing the independence of security and non-security comments and the presence of each of the four types of semantic uncertainty and an aggregation of the four to indicate the presence of 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 *** $p < 0.001$ (N) $\varphi < 0.1$

> **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 might suggest that security, being nuanced, is difficult to discuss in a concrete context.

4.2 RQ 2: Differentiating Security Talk

> **RQ 2:** How well do pragmatic characteristics of language differentiate between security and non-security conversation?

In this section, we examine how much different security and non-security conversation is with respect to our five metrics. We do this by examining inter-correlations between the linguistic metrics and by evaluating a predictive model.

We used Spearman's ρ to assess the correlation between our continuous-valued linguistic metrics (i.e. all but the uncertainty). Of note is the $\rho = 0.70$ between maximum formality and maximum informativeness. Both metrics are closely related due to their roots in Grice's maxims [9], which aligns with the interpretations by Lahiri [11]. The high, positive correlation between maximum formality and maximum informativeness is empirical evidence supporting Grice's pragmatic theory. We additionally examined correlation between the metrics within the security and non-security populations, but found no significant deviations between the two.

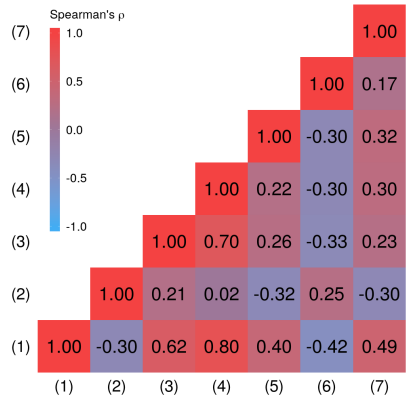

Figure 2: Correlation among continuous-valued metrics: (1) Number of Words, (2) Minimum Formality, (3) Maximum Formality, (4) Maximum Informativeness, (5) Maximum Implicature, (6) Minimum Politeness, (7) Maximum Politeness.

To assess the ability of the pragmatic characteristics to differentiate security and non-security bug comments, we used a logistic regression model evaluated using a stratified random sample of 100,000 comments. We used 10 iterations of 10-fold cross validation

to assess the performance of the model and found its precision, recall, and f-measure to be 95.34%, 6.56%, and 12.27%, respectively. While the model with the pragmatic characteristics improved in precision over the n-gram model from Section 3.1, it degraded in recall. In isolation, the model with the pragmatic characteristics may not be effective, however, in concert with the n-gram model, the overall effectiveness of differentiating security and non-security bug conversations may be improved.

5 SUMMARY & FUTURE WORK

Our goal in this work was to *characterize the pragmatic features of conversations about security so that developers can be more informed about communication strategies regarding security concerns.* We have examined both the content of security conversations and the way in which that content is conveyed. 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 the context of bug reports, exhibit varying degrees of formality, informativeness, implicature, politeness, and uncertainty. The difference between security and non-security conversations is small, but statistically significant. Combining pragmatic metrics into a single model did not produce a robust prediction model, either. However, the field of linguistic analysis offers many more concepts, analyses, and metrics. The results of this study warrant further study into what we believe is an information-rich data set.

In future work, we will examine the syntactic and semantic characteristics of security conversations, as well as discourse-level analyses. We also plan to collect more security conversations spanning other modalities, such as code review, mailing list, and chat. We will explore this data set more closely, making observations about security conversations that tie into linguistic concepts, which would translate to better discriminators, and therefore better insights into effective communication strategies.

6 LIMITATIONS

This study was conducted on a single case study, on the Chromium project. Thus, there are likely idiosyncrasies of it being a browser project, a Google project, an open source project, and a C/C++ project, that may make it unique. This is mitigated by the fact that Chromium is a widely-used product in the World Wide Web today and it is widely-studied project in academia. However, more case studies can help generalize these results across programming languages, domains, and developer communities.

All natural language requires preprocessing, which is an ongoing effort to clean out noise from the data set. Sources of noise in this data include: boilerplate responses, responses from automated bots, source code mentions that are not parsed properly by our lexer. While we worked to clean our data, we cannot guarantee that the data lacks noise. By opening our data set, we hope to see curation from the research community to assist in this process.

Questioning the applicability of the politeness, formality, informativeness, implicature, and uncertainty corpora to security conversations is reasonable. The politeness corpus consists of Wikipedia editor conversations and StackExchange question-and-answer conversations; the associated classification model is based on domain-independent lexical features [2]. The formality, informativeness, and implicature corpus from Lahiri [11] is a generic collection of natural language from blog posts, newspaper articles, and forum threads, which we are comfortable considering domain-agnostic. Finally, the uncertainty corpus [5, 17] used by Vincze [18] is explicitly stated to be domain-independent.

REFERENCES

[1] A Bosu, M Greiler, and C Bird. 2015. Characteristics of Useful Code Reviews: An Empirical Study at Microsoft. In *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*. 146–156. https://doi.org/10.1109/MSR.2015.21

[2] C. Danescu-Niculescu-Mizil, M. Sudhof, D. Jurafsky, J. Leskovec, and C. Potts. 2013. A computational approach to politeness with application to social factors. *Proceedings of the Association for Computational Linguistics (ACL)*.

[3] H.C. Dawson and M. Phelan (Eds.). 2016. *Language Files: Materials for an Introduction to Language and Linguistics*. Ohio State University Press.

[4] K.E. Denham and A.C. Lobeck. 2013. *Linguistics for Everyone: An Introduction*. Cengage Learning.

[5] R. Farkas, V. Vincze, G. Móra, J. Csirik, and G. Szarvas. 2010. The CoNLL-2010 shared task: learning to detect hedges and their scope in natural language text. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning—Shared Task*. Association for Computational Linguistics, 1–12.

[6] Redacted for double-blind review. 2017.

[7] Redacted for double-blind review. 2018.

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

[9] H.P. Grice, P. Cole, J. Morgan, et al. 1975. Logic and conversation. *Syntax and semantics* (1975), 41–58.

[10] F. Heylighen and J. Dewaele. 1999. Formality of language: definition, measurement and behavioral determinants. (1999).

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

[12] G. McGraw. 2006. *Software Security: Building Security in*. Addison-Wesley. https://books.google.com/books?id=HCQdypbpZXgC

[13] Andrew Meneely, Ben H. Smith, and Laurie Williams. 2012. Validating software metrics: A spectrum of philosophies. *ACM Trans. Softw. Eng. Methodol.* 21, 4 (2012), 24:1–24:28. https://doi.org/10.1145/2377656.2377661

[14] C. Potts. 2005. *The logic of conventional implicatures*. Number 7. Oxford University Press on Demand.

[15] M.M. Rahman, Chanchal K. Roy, and R.G. Kula. 2017. Predicting Usefulness of Code Review Comments Using Textual Features and Developer Experience. In *Proceedings of the 14th International Conference on Mining Software Repositories (MSR '17)*. IEEE Press, Piscataway, NJ, USA, 215–226. https://doi.org/10.1109/MSR.2017.17

[16] N. F. Schneidewind. 1992. Methodology for validating software metrics. *IEEE Transactions on Software Engineering* 18, 5 (May 1992), 410–422. https://doi.org/10.1109/32.135774

[17] G. Szarvas, V. Vincze, R. Farkas, G. Móra, and I. Gurevych. 2012. Cross-genre and cross-domain detection of semantic uncertainty. *Computational Linguistics* 38, 2 (2012), 335–367.

[18] V. Vincze. 2014. *Uncertainty Detection in Natural Language Texts*. Ph.D. Dissertation. University of Szeged.

[19] ZERODIUM. [n. d.]. ZERODIUM - How to Sell Your 0day Exploit to ZERODIUM. https://zerodium.com/program.html. [Online] Accessed: 01-28-2019.

APPENDIX

Table 4 contains means (with standard deviations), medians, and first and third quartiles for our linguistic metrics. These are the values that are visually represented in Figure 1.

Table 5 gives more examples of sentences exhibiting formality, informativeness, implicature, politeness, and uncertainty. These are sentences from security comments that we found to be interesting and insightful in addition to the examples included throughout Section 3.

Table 4: Measures of central tendency for continuous-valued metrics.

Metric	Mean		1st Quartile		Median		3rd Quartile	
	sec	nsec	sec	nsec	sec	nsec	sec	nsec
Minimum Formality	0.0640 ± 0.1740	0.0746 ± 0.1919	0.0006	0.0010	0.0035	0.0042	0.0260	0.0320
Maximum Formality	0.3204 ± 0.3712	0.3426 ± 0.3717	0.0147	0.0190	0.1068	0.1491	0.6539	0.7010
Maximum Informativeness	0.6658 ± 0.3955	0.7049 ± 0.3839	0.2352	0.3320	0.9250	0.9633	0.9993	0.9990
Maximum Implicature	0.5500 ± 0.3504	0.5465 ± 0.3473	0.2019	0.2090	0.5933	0.5829	0.8999	0.8910
Minimum Politeness	0.4393 ± 0.0806	0.4310 ± 0.0839	0.4032	0.3900	0.4390	0.4390	0.4697	0.4470
Maximum Politeness	0.5318 ± 0.0952	0.5272 ± 0.0913	0.4436	0.4470	0.5182	0.5124	0.5979	0.5870

Table 5: More examples sentences from bug comments exhibiting linguistic characteristics.

Metric	Example sentences from the Chromium data set
Informativeness	Low: *"Need help with triaging, thanks!"*
	High: *"In order to mitigate what the hidden page might do while the interstitial is showing, we now block all network requests for the hidden page, effectively preventing any redirect on the page."*
Formality	Low: *"If there's a security update going out, maybe we can catch it."*
	High: *"These resources can be viewed by others while in transit, and can be modified by an attacker to change the look of the page."*
Politeness	Low: *"Why not tell people the truth?"*
	High: *"I do think that it may be able to tweak that repro a bit to get the desired values for the integer overflow to happen - I could look into that if you want one?"*
Implicature	Low: *"I disagree though, that cross-window drags indicate a clear intention to perform DnD: The initial PoC opens a new tab as soon as you start dragging, which results in unintentionally switching the window during the drag operation - without any additional user interaction."*
	High: *"Maybe this feature should not get it's information from the omnibox, it feels unessesary."*
Uncertainty	Epistemic: *"Perhaps we have different versions of IcedTea, and perhaps that's important."*
	Doxastic: *"I don't think we can securely do anything about pushState/replaceState on iOS."*
	Investigative: *"In the long run, it should be considered, whether there should be a confirmation dialog for dangerous actions, where the OK-button is blanked out for at least one second."*
	Conditional: *"Alternatively, if you have a diagram that illustrates data flows through the components, servers, etc of your feature, that would be sufficient."*