Programmer's Perception of Sensitive Information in Code

Xinyao Ma, Ambarish Gurjar, Anesu Chaora, L. Jean Camp
Luddy School of Informatics, Computing, and Engineering,
Indiana University Bloomington
Indiana, USA
maxiny@iu.edu, agurjar@iu.edu, achaora@iu.edu, ljcamp@indiana.edu

***Abstract*—This study delves into the crucial role of developers in identifying privacy sensitive information in code. The context informs the research of diverse global data protection regulations, such as the General Data Protection Regulation (GDPR) and the California Consumer Privacy Act (CCPA). It specifically investigates programmers' ability to discern the sensitivity level of data processing in code, a task of growing importance given the increasing legislative demands for data privacy.**

We conducted an online card-sorting experiment to explore how the participating programmers across a range of expertise perceive the sensitivity of variable names in code snippets. Our study evaluates the accuracy, feasibility, and reliability of our participating programmers in determining what constitutes a 'sensitive' variable. We further evaluate if there is a consensus among programmers, how their level of security knowledge influences any consensus, and whether any consensus or impact of expertise is consistent across different categories of variables. Our findings reveal a lack of consistency among participants regarding the sensitivity of processing different types of data, as indicated by snippets of code with distinct variable names. There remains a significant divergence in opinions, particularly among those with more technical expertise. As technical expertise increases, consensus decreases across the various categories of sensitive data. This study not only sheds light on the current state of programmers' privacy awareness but also motivates the need for developing better industry practices and tools for automatically identifying sensitive data in code.

I. Introduction

There are myriad requirements for privacy and data protection. Advances in data protection legislation implement different requirements in specific domains, such as in the manipulation of general health data or DNA. There are varying global requirements for privacy, beginning with the General Data Protection Regulation. American states have unique requirements, from the list of variables in the California Consumer Privacy Act to the limits on biometrics in Illinois, Texas, and Washington. There are also combinations of these; for example, New York banned all biometrics in schools for three years and then extended that ban to facial recognition [1]. As a result, developers may reasonably be expected to be aware of the potential for privacy issues in code. This study aims to shed light on these perceptions and how they correlate with expertise.

Specifically, we explore the degree to which identification of privacy sensitivity in code is a cognitively feasible task. We instrument this by asking participants to identify code with named variables as processing data defined as sensitive by the GDPR and CCPA. Building on previous usability research, which has focused on developers as users as well as creators of technology, we examine the degree to which developers of differing levels of expertise can identify the sensitivity of variables according to the CCPA and the GDPR. We selected variables and code snippets that process data that has been found to be personally identifiable information (PII) or privacy sensitive. We examined how programmers perceive the sensitivity information that is embedded or conveyed in code. Additionally, we analyzed their awareness of the sensitivity level for the ten categories of personal information as defined by the GDPR and CCPA. We conducted an online card-sorting experiment to assess programmers' perceptions of variable name sensitivity under specific conditions. The study aimed to answer the following questions:

1) Is there a consensus among programmers about what constitutes 'sensitive' information?
2) Does security knowledge level affect the awareness of privacy issues?
3) Is any consensus or the impact of expertise consistent across different categories of PII?
4) How do human classification patterns differ from that of Large Language Models?

The ability of participants to correctly and consistently identify code snippets with privacy-sensitive variables is the measure we use to address the more specific research questions below, and inform the questions above. There are four specific hypotheses the experiment reported here was designed to address.

H1: Is there agreement among programmers across a wide range of technical expertise as to which code snippets and variables have indicators that they are processing sensitive data?

We report on the accuracy of classification and agreement among participants based on their identification of code snippets which process clearly named sensitive variables. We found that participant consensus varies for different kinds of information.

H2: Are perceptions of sensitivity of data, as measured by perceptions of variables shown in the context of code snippets,

Symposium on Usable Security and Privacy (USEC) 2024
26 February 2024, San Diego, CA, USA
ISBN 979-8-9894372-5-2
https://dx.doi.org/10.14722/usec.2024.23075
www.ndss-symposium.org, https://www.usablesecurity.net/USEC/

influenced by the technical expertise of the developer?

We compare the agreement and accuracy of participants sorted by expertise. Developers were evaluated based on their knowledge. Those who answered all the knowledge questions incorrectly also fail to identify most PII categories, but they put more emphasis on Account/Identifier Identifiers. Expertise is measured using the questions developed by Kelley et al. [2], [3].

H3: Is the difference in the classification of code more a function of expertise than an individual variable?

Our results identify the most and least correlated variables. Accuracy and agreement are reported for the ten data categories examined in the experiment. We illustrate that these differences are categorical. Thus, the differences we see can not reasonably be argued to be a result of the difficulty of reading the name or identifying the category of a variable, as these are internally consistent. However, the contrast between groups with different levels of expertise suggests that these groups of participants have very different overall perceptions of sensitivity.

H4: Is the categorization of code snippets similar between LLMs and human participants?

As Large Language Models have been trained upon human text and human decision, it is reasonable to assume that their decisions would align with those of human participants. Should LLMs accurately identify sensitive code, then the diffusion of such models has the potential to resolve the cognitive challenges we have identified here.

Our results indicate that programmers don't often agree on the sensitivity of the data being processed, as indicated by variables embedded in code snippets. We identify here expertise of knowledge, specifically on participants' ability to answer four security questions selected from previous related literature. Higher levels of expertise decrease, rather than increase, levels of agreement about the privacy sensitivity of data as indicated by code snippets. We report these patterns in terms of expertise and category of information.

While most developers understand that code can hold sensitive data, when asked to evaluate code snippets with clearly named variables they do not consistently identify those as being privacy-sensitive. Experts and non-experts differ on what is sensitive: non-experts generally agree more with each other, while experts disagree more, especially on which variables are most sensitive. These disagreements are even more pronounced when different categories of sensitive data are compared. After we detail our experiment, we provide a summary of the performance of artificial intelligence when given the same task. For this comparison, we queried the LLMs on snippets with variables clearly defined as sensitive by the GDPR or CCPA. In tWe note that there are also systematic failures in LLMs in identifying sensitive information embedded in code. We can conclude that querying an LLM about the sensitivity of a code snippet is an unreliable way forward. We close with possible paths forward to support developers, and those who rely on their code, in evaluating the risks of processing sensitive information in non-obfuscated code.

II. Background and Related Work

Development practices play a critical role in security and privacy. Significant work has been done to understand the role of incentives and usability in the production of insecure or less secure code.

Here, we discuss two areas of work with the greatest influence on the development of our research questions and experiment design. The first of these is analyses of coding practices, in particular, the examination of the use of permissions. The second is an examination of the factors that create security errors in code.

A. Privacy in the Permissions Landscape

If we consider the possibility that lack of awareness about privacy is due to insufficient expertise or knowledge, then research on permissions could prove illuminating. Privacy in the mobile phone ecosystem is managed in apps by using permissions for requesting data categories.

An early study found chronic over-permission and repeated research has reified this. Such over-permissions included requests that were deprecated or not accessible by the app, and, therefore, by definition, not needed for the program to operate [4]. The authors proposed that the driving forces behind this included bad information from peer sources (i.e., StackOverflow) and practices that placed functionality over security and privacy (e.g., including entire categories instead of specific permissions). A follow-up paper illustrated that the examination of SDK provided a more nuanced understanding of data use [5]. More recently, an examination of compliance with Apple privacy labels found the same types of errors and incorrect assertions about data use [6]. A study of IoT apps found that nearly one-third (1,973 out of 6,208 apps) of those interacting with IoT expose data without proper disclosure and thus without consent. The scale and scope of undisclosed data-shared illustrated a pervasive pattern of sharing data across jurisdictions, including highly sensitive information [7].

Previous work suggests that over-permissions and data leakage are not purposeful noncompliance but instead driven by the contested and multidimensional nature of privacy. Another work also examined significant differences in work practices, licensing models, and development cultures between the iOS and Android ecosystems, arguing that the source of the over-permission was Android's open-source nature. Developers have more flexibility and fewer privacy constraints than iOS, leading to different Android values, including permissions and user requests [8]. Except for the inherent open-source community nature, lack of formal knowledge is another possible reason identified as to why developers cannot embed privacy into software systems. For example, Senarath's findings revealed that most developers lack formal knowledge of privacy practices and that they try to integrate privacy features into software designs without much understanding [9], [10]. This may particularly be the case for dependencies where developers have significant confusion about the privacy risks of libraries [11]. A mixed method evaluation of developers' challenges in creating correct privacy labels for apps explored the issue as one of interaction design; our results indicate the lack of understanding of the sensitivity of data is a core challenge [12].

In a study close to the human-subjects research on developers, Chamila and Nalin conducted a qualitative study with 40 software developers. They revealed that most programmers believe they should be responsible for end-user security [13]. Yet this does not result in secure code, nor do the developers report following best practices. Their lack of security expertise and the extra time required for testing were two main reasons for not following the secure development standards. Our results indicate a similar lack of privacy expertise [4].

It is reasonable to hypothesize that greater expertise in security and privacy may yield a better understanding of sensitive data. However, a study of over 700 participants, augmented with a qualitative evaluation of interactions with experts, contravenes this hypothesis [14]. The level of sensitivity or privacy of both individual data and categories was contested among the large sample of MTurk participants, yet the small expert group also failed to find a consensus.

This study is specifically on popular code found in GitHub, as opposed to mobile apps, so part of our analysis was necessarily determining how to define privacy. We describe our definition of privacy after a brief discussion of related work on security errors in coding.

B. Security Errors in Code

If we consider an inability to identify sensitivity in code as a usability error, then the research on human factors in the creation of coding errors is relevant. Here, we focus on studies that, like our work, examine the interaction of developers and code. An early call for understanding the requirements of developers provided a set of ten principles for creating secure and usable APIs, focusing on the high cost of flawed cryptographic APIs [15]. They advocate for aligning APIs with pre-existing perceptions and recognizing the limits of the average developer's understanding of cryptography. Just as cryptographic APIs embed an expectation of high levels of security expertise, privacy requirements may depend upon developers' understanding of the compliance risks of all the contexts in which code may be used or the risks of a particular snippet or dependency in their targeted context.

Our research is further motivated by Acar et al.'s work [16] that call on a research agenda for developers. This research is aligned with the call for a better understanding of both developers' knowledge and the status quo. Specifically, we seek to better understand developers' privacy knowledge and measure the status quo by evaluating the interaction of human subject participants with code snippets obtained from popular GitHub repositories. As with previous work, we use GitHub as a sample for code; however, our recruitment of developers was less targeted [17].

Previous analysis of GitHub has shown that security practices are lacking, sometimes in obvious ways. An analysis of coding practices on GitHub has been done by Yasemin Acar et al.. They recruited 307 active Github users to complete the same security-relevant programming tasks, and they found being a professional did not increase a participant's likelihood of writing functional or secure code statistically significantly [17]. Another work on security issues about Github examined billions of files including real-time public commits and a snapshot covering 13% of open-source repos. They also found that the issue of confidential information being inadvertently disclosed on public repository platforms remains widespread and unresolved. This situation continuously exposes both developers and services to heightened risks of security breaches and potential misuse [18]. Besides the risks caused by the programmers, we found chronic exposure to authenticating information, such as keys and passwords [19], [20]. The authors found authenticating data in public repositories along with their variable names, including "pwd", and "password". A particularly famous example of a password leak was the administrator password *solarwinds123* being posted in public for some months and not changed for several weeks after this was reported, before the Solar Winds debacle [21].

Lack of reliable sources and inefficient sources have been identified as drivers for such insecure practices. An examination of the manner in which developers search for coding examples found that peer-based systems are a source of failures in practice [16]. In a comparison of four experimental groups of developers asked to implement security functions, those using Stackoverflow were found to be most likely to provide insecure implementations. Those using formal company documentation were most correct in terms of secure implementations but also most likely not to complete the tasks. A follow-up focused on misconfiguration and other security errors in Android found that 15.4% of a 1.3 million sample contained security-related code snippets from Stack Overflow. Out of these 97.9% contain at least one insecure code snippet [22]. The most widely used snippets included security warnings but were nonetheless included. Similarly, in our selection of snippets from GitHub, we focused on more popular and widely used code.

Schoenherr emphasizes the need for an ethics of cybersecurity that goes beyond traditional paradigms of computer and information ethics. It highlights the necessity to account for social and cognitive factors influencing users and calls for developers to understand these mental models to develop effective cybersecurity strategies [23]. Acar et al.'s findings of functionality over security were reified in a series of interviews with developers on insecure handling of passwords [24]. The reviewers implemented a mixed methods approach with a logged implementation task, a survey, and an interview using a think-aloud protocol with 20 developers. As with our results, there is a significant gap between perceptions and reality; however, we report on perceptions of privacy rather than security. In security, there are also increasingly available tools to identify data leaks, particularly in terms of static [25] and dynamic [26] code analysis tools. Our research asks if developers can identify privacy concerns beyond permissions and leaks. These results suggest that there is widespread confusion in developers' peer sources; our results echo this in the case of privacy.

The existence of nudges and sample code has proven effective in altering the security and privacy choices of developers in other domains [27]. Thus development and diffusion of usable, clear, secure examples are components of our research, and potentially a contribution in themselves.

III. Methodology

In this section, we start by outlining our data acquisition process, followed by a description of our recruitment

Category	Repos	Lines	Variable	Code Snippets
Unique Device ID	6	958	27	25
Individual Identifier	9	1643	29	28
Demographics	6	179	20	21
Internet Traffic	7	2549	46	57
Financial Information	8	1068	44	50
Biometrics	5	490	32	38
Multimedia Data	4	1388	20	31
Employment	7	943	29	31
Location	8	1472	50	58
Education	2	102	7	6

TABLE I. SUMMARY OF VARIABLE CODE CORPUS

procedures and the resulting participant pool. We provide a description and image of instructions provided to participants and a sample of the classification task.

A. Data Acquisition

One of the difficulties in exploring perceptions of privacy is the level to which privacy definitions would vary based on jurisdiction and content. In evaluating security compliance or permissions in code, there are pre-existing requirements and categories. In order to ground our definitions of privacy sensitivity, we began with the categories of information clearly identified in the CCPA and GDPR. Each enumerates data types that are considered privacy sensitive and, therefore, indicate sensitive data with compliance considerations. Reviewing the regulations, we chose ten information categories: 1. Unique Device ID, 2. Individual Identifier, 3. Demographics, 4. Internet traffic, 5. Financial information, 6. Biometrics, 7. Multimedia data, 8. Employment, 9. Education, and 10. Location. Detailed category and variable information can be found in Appendix B.

For this experiment, we focused our scope on Python code. We searched for identifiable variables in the selected categories, based on brainstorming and previous literature. GitHub is the biggest open-source code-sharing platform with a wide range of code from individuals [28], making it the obvious choice of source for acquiring a large code database. As we focused on code with potential privacy implications, we searched all Python GitHub repositories for a small set of data privacy-related keywords and data types. Searching for these keywords resulted in a corpus of 53 repositories, 30 keywords, and 86 variables with their related variables and code expressions in total. We used different keywords for each category domain to search the repositories on the GitHub platform. We also extract the nearby variables and code segments that have the value transmission with the picked variable.

Please see the following example of a code snippet containing *financial information* that would be classified as PII under GDPR and the CPRA. In addition, a compilation of such information might create risks because of its potential for abuse. Thus, we would expect all participants to classify this as sensitive to some degree. (All human subjects did in fact classify this as sensitive to some degree. Yet the artificial intelligence fared much worse than human judgement in this category. Note in TableVII, ChatGPT 3.5 identified only 12.5% of financial information snippets as sensitive; ChatGPT 4.0 identified only 25% of such snippets, and Bard correctly classified 37.5% of snippets including financial information as sensitive under GDPR or CCPR. Please see Section IV-D for details.)

```
online_bank_statement_provider = fields.Selection(
    selection=lambda self: self.env[
        "account.journal"
    ]._selection_online_bank_statement_provider(),
    help="Select the type of service provider (a model)",
)
online_bank_statement_provider_id = fields.Many2one(
    string="Statement Provider",
    comodel_name="online.bank.statement.provider",
    ondelete="restrict",
    copy=False,
    help="Select the actual instance of a configured
    provider.\n"
)
```

B. Experiment Design

We conducted an online experiment used to collect programmers' perceptions of variable names' sensitivity under certain conditions. The experiment consisted of two parts: the first part was to classify the variable name into different sensitivity levels, and the second part was to choose which category the variable belongs to. The variable names and their code snippets are selected from the dataset described in Table I for each category.

Our experiment combined survey questions about expertise with an online experiment on collecting perceptions of code and privacy. Our specific goal was to discover how our participants categorized and related the concepts of sensitivity with the code samples provided. Providing such an understanding is the core function of card sorting as a method [29]. Since card sorting was introduced in the seventies, it has been used to explore the decision-making of participants [30].

We developed a classification website using card sorting [30] for better usability and more accurate results. The classification interface for participants is shown in Figure 2. Card sorting is particularly appropriate for this experiment because it is designed to illuminate how participants categorize and relate concepts [29], [30]. Compared with the survey, card sorting decreases the cognitive burden on the participants as well as being more effective in measuring subject decisions [31]. One concern in our design was the cognitive load on participants, as each sorting action required that a participant to read multiple lines of code. We were informed by previous research that leveraged card sorting to illuminate privacy and security perceptions of expert and non-expert participants [32]–[34].

In this study, participants had to sort variables into different classifications that represent different levels of sensitivity: highly sensitive, sensitive, less sensitive, and not sensitive. In part, this is motivated by our grounding of sensitive data in legal doctrine. Whether data are sensitive and subject to compliance or not is sometimes conditional or contested. For example, extensive litigation was needed to determine if IP addresses are PII under the European Data Protection Directive (Directive 95/46/EC) [35], [36], which was the precursor to the GDPR [37]. We rejected a binary classification based on previous research, which found that if developers considered an item contextually sensitive or less sensitive, they might identify it as neither sensitive nor privacy-violating overall [8], [9], [11], [23]. Thus we wanted more than a simple Boolean choice.

Having decided against a simple yes/no decision, the obvious design would include a standard Likert Scale of one to five. In our experiment, we sought to determine which data are perceived as sensitive and a neutral response would not align with these goals. Further, marketing research has repeatedly found a pattern of people choosing "medium" or the middle answer when faced with a range of choices and a lack of certainty about their preferences [38]–[40]. The traditional five-point scale enables participants to make that medium or indeterminate choice. Thus, we provided the option to choose less sensitive or not sensitive, but no indeterminate choice. Our goal was to observe decision-making; thus, we removed the option not to decide.

An instructions page was provided before the sorting task itself, as shown in Figure 1. These instructions present the layout of the sorting pages, highlighting the information and actions required for the sorting task. During the sorting task, each participant was required to read a small snippet of code and classify each one of the forty snippets as shown in Figure 2.

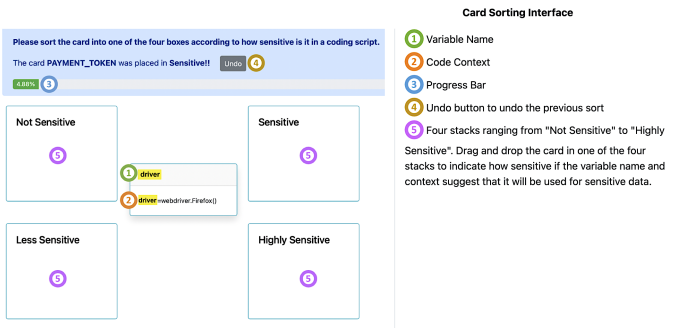

Fig. 1. The participants were given the following image for instructions. In the experiment itself, the variable was presented as part of a code snippet. The details were not included in the instruction to simplify the instructions.

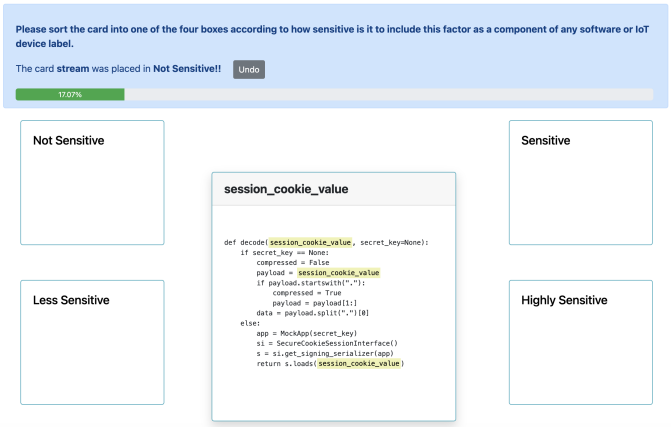

Fig. 2. Each participant had to read a small snippet of code and classify each one of the snippets shown using a drag-and-drop card sorting experiment. The example variable "session_cookie_value" is highlighted. The context of each snippet was provided so that under-specified variables (e.g., mark) or the purpose of the variable (e.g., date) was clear.

C. Recruitment and Participants

All recruitment and enrollment were preceded by review and approval by the institutional IRB. We primarily sought to recruit participants with computer science backgrounds who have some coding experience. In addition, we evaluated their level of expertise in cybersecurity. We advertised our survey on the computer science department email list and the Prolific platform. Each qualified response was paid 4 dollars based on our assumption that the survey would take fifteen minutes, thus resulting in a wage of $15 an hour. Qualified responses were those with valid answers to the attention check question, which took greater than the minimal possible time to read through the survey; and having completed the survey. Specifically, valid responses placed the attention check card in the correct category and required more than 5 minutes to complete. In total, 160 people with varied computing backgrounds completed the survey, but only 99 responses were selected for further analysis based on this criterion. The demographic information for the 99 participants is shown in Table II. The 99 participants are divided into different groups of more or less experts, based on their answers to the four security expertise questions. (These questions are provided in Appendix A.)

TABLE II. DEMOGRAPHICS OF PARTICIPANTS

Item	Options	n
Gender	Male	63
	Female	29
	Non-binary	5
	Prefer not to answer	2
Age	18-25	44
	26-35	37
	36-45	13
	46-55	3
	56+	2
Major	Computer Science	81
	Non-Computer Science	14
	Prefer not to answer	4
Programming Duration	Less than 1 year	10
	1 - 3 years	40
	4 - 6 years	26
	7 - 10 years	8
	More than 10 years	7
	Prefer not to answer	8
Programming Languages	1 - 2	25
	3 - 4	41
	5 - 6	18
	7 - 8	2
	More than 8	5
	Prefer not to answer	8

IV. RESULTS

We found that participants diverged significantly in opinions when it comes to determining the sensitivity levels of variables, with no strong consensus emerging. Interestingly, the data suggests that as programmers gain more security knowledge, their awareness of privacy issues might increase on average, but this differs across categories. While there is a general acknowledgment among developers that code can contain sensitive information via variables, there is a tendency to overlook certain categories of sensitive data. There is no notable difference between non-expert and expert groups on agreement on variable sensitivity: they all tend to have no or fair agreement on which variable should be unified and sensitive. however, the experts have a higher disagreement in general. We also found that the discrepancy varied across the different categories.

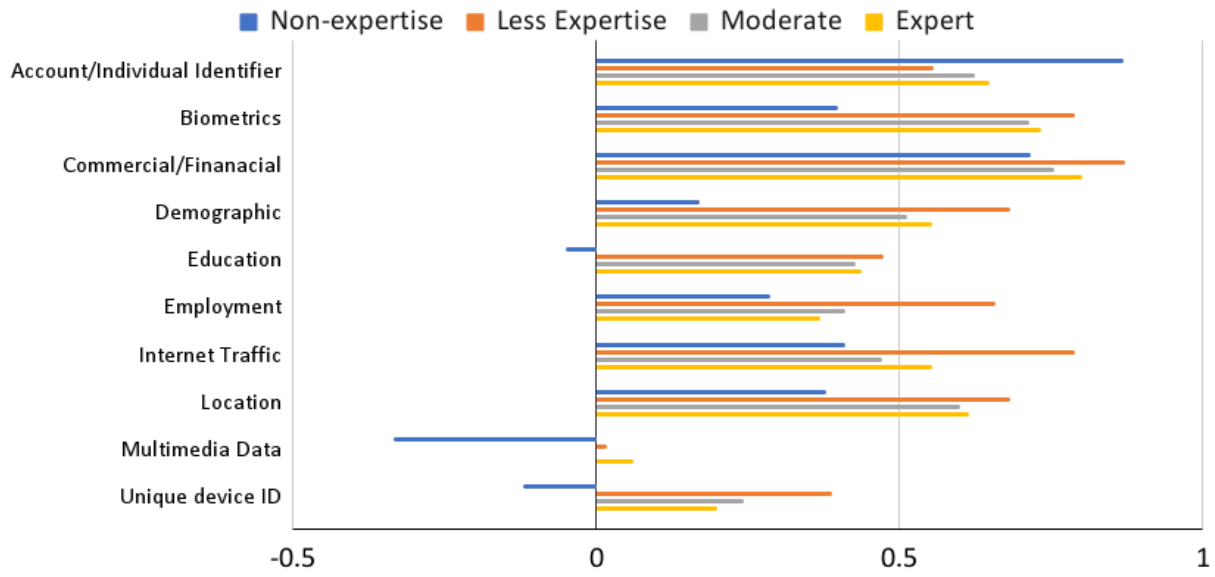

Fig. 3. Sensitivity identification for different security-expertise levels group

A. *Sensitivity Identification*

For each personal information category we explored in this study, Figure 3 depicts the sensitive identification ability corresponding to different security knowledge levels, calculated by the standardized scores with binary classification: “sensitive” and “not sensitive”. The Y-axis represents the percentage of four levels of expertise: the programmers are divided into four groups, ”non-expertise”, ”less expertise”, ”moderate expertise,” and ”expert”, based on their security knowledge. The X-axis displays the sensitivity scores calculated by each variable according to its category. For the scores, variables that have been accurately identified are accounted by a value of ’1’, whereas those that have been characterized as ”not sensitive” have a value of ’-1’. Based on the scores, most of the groups identified these 40 variables as sensitive, a score larger than 0, except for the programmers who answered no security questions correctly. They had negative scores in ”education”, ”multimedia data,” and ”unique device ID” categories. However, they put more emphasis on the ”account/individual identifier” category, which contains variables like ”email”, ”name”, and ”username”, (the full list of variables can be found in Appendix B) than any other group. Interestingly, the group with less expertise, defined as the group that answered only one technical question correctly, identified the variables as the most sensitive in eight categories and performed better than the other three groups. Less expertise group performs even better than moderate and expert groups in seven out of ten categories when their categorizations are better.

We perform the ANOVA test for the group’s evaluation of the variables’ sensitivity score based on its correlation with different categories in Table III. We did not include the participants’ education category since the differences in the classification of code are clearly more a function of expertise than individual variables. In each category, there is little difference between the classification of the variables. Only ‘Internet traffic’ shows significant in-category differences. These differences were marginally significant with 99 participants, an

TABLE III. ANOVA TEST FOR DIFFERENT CATEGORY

Category	F-value	p-value
Unique Device ID	1.362	0.246
Individual Identifier	0.638	0.635
Location	2.138	0.075
Demographic	1.500	0.201
Internet traffic	3.087	0.016
Financial	1.201	0.309
Biometrics	1.296	0.272
Employment	0.888	0.471
Multimedia Data	1.593	0.176

TABLE IV. ANOVA TEST FOR DIFFERENT GROUPS

Group	F-value	p-value
Expert	6.214	2.88E-27
Moderate	6.675	3.18E-30
Less Expertise	4.868	9.48E-19
Non Expertise	6.794	1.29E-30

F-value of 3.087 and a p-value between $0.015 < 0.05$. In all other cases, the variables in a given category were categorized in a similar manner by human participants.

The same ANOVA test has been applied to the correlation between the category sensitivity score with four groups of different expertise levels IV. That is, all the categories were combined for the participants to create four sets of data for the sensitivity level of 1-4. These datasets were compared across the four groups without regard to category. The differences were remarkable. Interestingly, the results show that all groups have a significantly different evaluation of categories’ sensitivity (all p-value< 0.001), which means the category sensitivity was distinctive at every expertise level.

The results imply that at each level of expertise, the participants classified variables in the same category in a similar way. Yet, when looking at overall patterns, different levels of expertise resulted in highly different levels of categorization. Additional expertise does not result in more accurate assessments; however, it does appear to create a systematic difference in the classification of data as more or less sensitive.

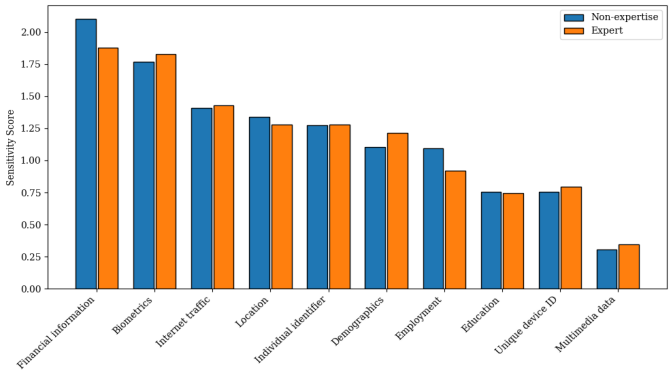

Fig. 4. Sensitivity Ranking and Score for Non-Expert and Expert illustrating

In the following analysis, we categorized individuals with no expertise or limited expertise as 'Non-expertise' and those with moderate to high expertise as 'Expert' for subsequent analyses. This approach was adopted to ensure a sufficiently large sample size in the 'Expert' group, facilitating meaningful comparisons with other groups. As illustrated in Figure 4, a score below '1' indicates that the item is deemed non-sensitive by all participant assessments. The 'Expert' group identified the categories of 'Multimedia,' 'Unique Device ID,' 'Education,' and 'Employment' as non-sensitive. In contrast, the only distinction for the 'Non-expertise' group was in the 'Employment' category. These findings suggest that the 'Non-expertise' group exhibits marginally better performance than the 'Expert' group in this context.

B. Sensitive Similarity for Category and Variables

In this section, we discuss the results of a similarity matrix. The data is derived using a cosine similarity algorithm on the sensitivity score of each variable. Full visualizations of these are provided in the released code and data link to present them at a large scale. A notable observation is the existence of strong similarity clusters. We selected 7 variables to identify the correlation between general programmers and programmers with more security knowledge. Figure 6 depicts the most and least correlated three variables among the less-expert (combined none and less programmers) group and the expert (combined moderate and expert) group. The most correlated variables are the same for the two groups, which are also the most sensitive variable information: "password", "PAYMENT_TOKEN," and "paypal_mapping_id". The cosine similarity of these three groups is close to 0.9. However, there are highly uncorrelated variables as well: "driver", "gdf_mask", "MP4ASampleEntryBox," and "skill". These are also those with lower average scores; i.e., they were identified as being less sensitive. Those participants with high levels of expertise appeared to split all the variables into three big categories. (This is clearly visible in the three darker square areas in the appended heatmap.) The first of these expert categories includes the variables "gdf_mask", "skill", "gmaps", and finally the variable "long". The second of these categories includes all the financial-related variables as well as "password", which all were rated as having a high level of sensitivity. This distribution shows they tend to group all the related variables together and rank them at a similar sensitive level. It is possible that more expert participants have heuristics or rules of thumb that cause these patterns. This can not be verified by the results; instead, it provides a research question for future qualitative research. Participants with less expertise did not appear to have the same grouping behavior.

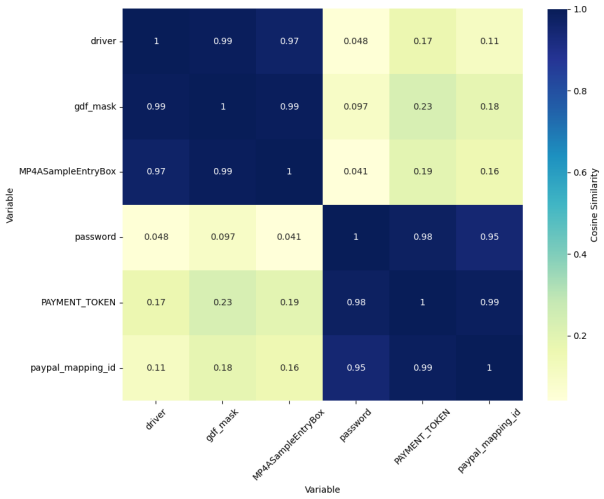

Fig. 5. The most and least correlated variables among participants with non or low expertise differs from the high expertise group with the inclusion of *MP4ASampleEntryBox* as shown in the fourth row

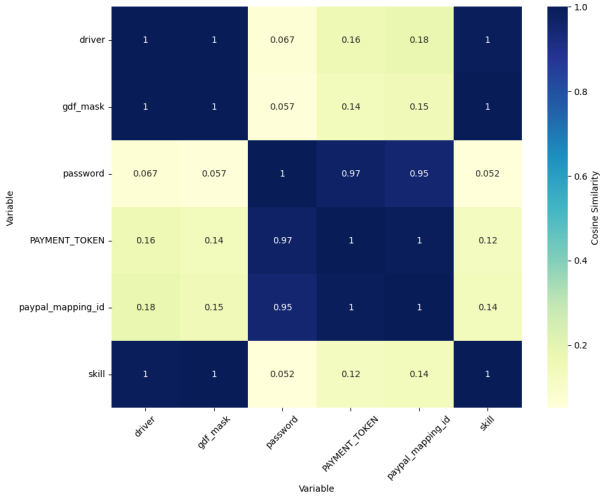

Fig. 6. The most and least correlated variables among participants with high expertise differs from those with less expertise in the inclusion of *skill* as shown in the bottom row.

To better understand the relationship between the variables and participants within different categories and groups, we established a "sensitivity score" scale ranging from -1 to 3, indicating levels from "not sensitive" to "highly sensitive." Figure 4 depicts the sensitivity scores for each category and both groups. Consistent with the sensitivity level findings, the top four categories ("financial information," "location," "individual identifier," and "demographic"). In all cases except educational variables, experts ranked data as more sensitive. The means test for each category and variable can be found in the Appendix. Location data is rated as being more sensitive than all other categories except financial by non-experts. While location is considered less sensitive than unique identifiers by experts, the general increase in awareness of sensitivity results in experts ranking location as more sensitive (9) than non-

experts (8).

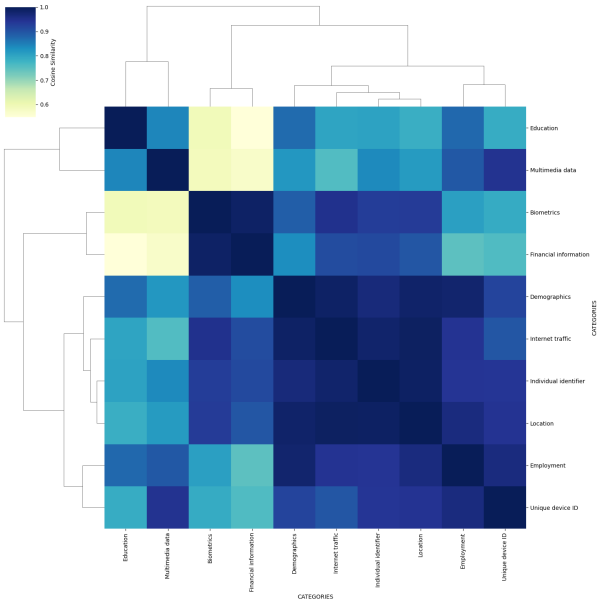

Fig. 7. Category similarity Matrix for those with less expertise reflects the differences in patterns of rankings from that of experts.

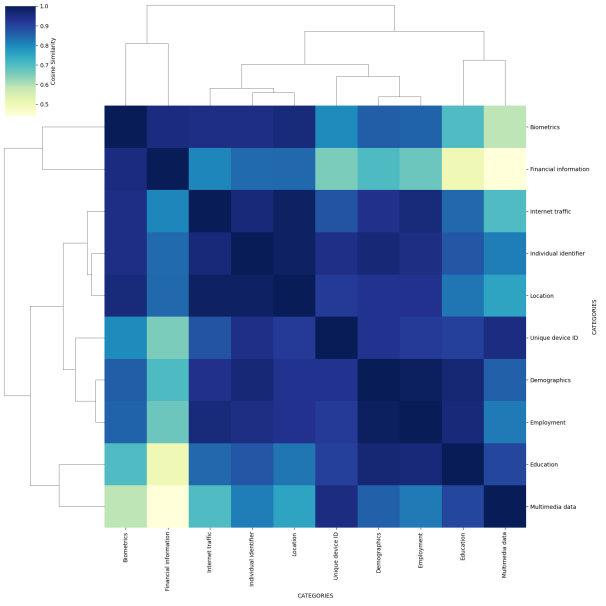

Fig. 8. Category similarity of the expert group shows different patterns of correlations than those found among those with less expertise.

From the category similarity matrix for the non-expert group, we note that ”demographics,” ”employment,” and ”education” tend to group together, and ”Internet traffic,” ”individual identifier” with ”location” categories tend to have similar sensitive level classification non-expert programmers. However, the expert group sorted them into three main clusters, ”education” and ”multimedia”, ”biometrics” and ”financial”, and all remaining categories are the third cluster. This is illustrated in Figure 7 and Figure 8. These provide a heatmap of the similarity matrix of the ten categories. The dendrogram lines connect the categories that have similar aggregated sensitivity levels. The similarity matrix calculation makes it infeasible to reject our hypothesis that expertise does not influence the categorization of variables as sensitive across different variable names.

C. Agreement on Variable Sensitivity

We found that the non-expert programmers have more unified opinions compared with security expert programmers in general, and a large discrepancy exists for different categories. Recalling our H1 is whether the programmers have a unified opinion on the sensitivity of privacy information in code and if there is any difference between the experts and general programmers. In order to evaluate that quantitatively, we used the Fleiss' Kappa score to evaluate raters' agreement on the sensitivity classification. Fleiss' kappa is a statistical measure for assessing the reliability of agreement between a fixed number of raters when assigning categorical ratings to a number of items or classifying items. In this context, a kappa value less than 0 means no agreement, and the level of agreement increases with this Kappa value, the max value is 1, which means substantial agreement.

Table V presents the Fleiss Kappa scores for both non-expert and expert groups across two different settings: one with 4 sensitive levels and another with 2 sensitive levels (sensitive and not sensitive). In this table, the non-expert group had a Fleiss Kappa score of 0.117 when evaluating data with 4 sensitive levels, while the expert group scored slightly lower with 0.112. When the data was simplified to only 2 sensitive levels, sensitive or not sensitive, the Fleiss Kappa scores increased for both groups: 0.217 for the non-expert group and 0.203 for the expert group. The increase in scores suggests that both groups found it easier to agree when there were fewer sensitive levels to consider. Based on a comparison between the two groups, the Fleiss Kappa scores for the non-expert group are consistently higher than those for the expert group. This suggests that the non-expert group tends to be more unified in their assessment of sensitivity levels for variables.

TABLE V. FLEISS KAPPA SCORE FOR 4 AND 2 SENSITIVE LEVELS: SENSITIVE AND NOT SENSITIVE

Fleiss Kappa score	Non-Expert group	Expert group
4 sensitive levels	0.117	0.112
2 sensitive levels	0.217	0.203

Table VI presents Fleiss Kappa scores based on four sensitivity levels for each category we evaluated, comparing the agreement among raters in both non-expert and expert groups. Each row in the table represents a different category of data, such as “Location,”, “Unique device ID,” “Demographics,” and so on. The scores range from negative to positive, with positive scores indicating a greater level of agreement among the raters. For instance, the category “Individual identifier” had the highest Fleiss Kappa scores for both groups, with 0.263 for non-experts and 0.231 for experts. This suggests that there is a relatively higher level of agreement among raters when it comes to assessing the sensitivity of individual identifiers.

On the other hand, some categories like “Biometrics” and “Education” have negative Fleiss Kappa scores for both groups, indicating a lack of agreement among the raters for these categories. Interestingly, some categories, such as “Employment” and “Unique device ID”, show significantly

different scores between non-experts and experts. For "Employment," experts have a Fleiss Kappa score of 0.158, considerably higher than the non-experts' score of 0.058. This discrepancy could reflect differing perspectives on data sensitivity between the two groups.

Overall, Table VI offers a detailed look into how different categories of data are perceived in terms of four levels of sensitivity by non-expert and expert groups, as evidenced by their Fleiss Kappa scores. However, we have to mention that this standard is not universally accepted because the number of categories and subjects will largely affect the kappa value; for example, the kappa is higher when there are fewer categories [41], so we cannot use the score to compare with other studies using the same standard, but it still feasible when we evaluate the results for different categories only within our own experiments.

TABLE VI. FLEISS KAPPA SCORE FOR TEN CATEGORIES

Fleiss' Kappa score	Non-Expert	Expert
Location	0.060	0.033
Unique device ID	0.052	0.111
Demographics	0.029	0.027
Internet traffic	0.015	0.014
Individual identifier	0.263^{+}	0.231^{+}
Employment	0.058	0.158
Biometrics	-0.012	-0.009
Education	-0.023	-0.019
Multimedia Data	0.031	0.010
Financial information	0.074	0.062

Recall that the lower the Fleiss Kappa score, the less the agreement. Scores of lower than 0.2 indicate an absence of consensus. The only significant agreement about sensitivity was for individual identifiers, where participants across levels of expertise identified these as sensitive. These results indicate that it is not reasonable to accept the hypothesis that there is significant agreement among programmers about what information is sensitive.

D. Large Language Models

A clear question arising from these results is the extent to which the judgment of LLMs can replace human judgment or highly specialized tools. To determine if this is a clear path forward, we selected sensitive code and developed a query to multiple LLMs: ChatGPT 3.5, Chat GPT 4.0, and Google BARD.

Based on limits on query size, we asked about four snippets per query: "Given below are 4 fragments of code. Each of them is separated by a '===='. Considering the first code as '1' and the fourth as '4,' identify if the fragments contain sensitive data according to GDPR or CCPA guidelines, indicating your response as 'code fragment number: Sensitive or Not Sensitive'." In cases where the language models' responses were ambiguous, we employed additional targeted prompts to clarify their assessments [42].

The results were not promising. As with human decision-making, there were clear patterns in failing to identify code as processing sensitive data. Demographic data and geofencing or location code were considered not sensitive. Code clearly containing the date of birth was classified as non-sensitive by both versions of ChatGPT and Bard. While the human participants identified unique identifiers as being sensitive, the LLMs misclassified code containing unique device identifiers, financial information, and biometrics as not sensitive; see Table VII for detailed information.

TABLE VII. PERCENTAGE OF FALSE NEGATIVES WHEN LARGE LANGUAGE MODELS WERE QUERIED

Category	Samples	ChatGPT 3.5	ChatGPT 4.0	Bard
Unique Device ID	11	90.91%	36.36%	54.55%
Individual Identifier	5	60%	20%	40%
Location	7	100%	0.00%	42.86%
Demographic	4	75%	25%	100%
Internet traffic	7	85.71%	42.86%	28.57%
Financial	8	87.5%	75%	62.5%
Biometrics	2	100%	0.0%	50%
Employment	7	85.71%	14.29%	57.14%
Education	2	50%	50%	100%
Multimedia Data	4	75%	100%	50%
Total	57	84.21%	40.35%	54.39%

While the evaluation of LLMs in the identification of sensitive data in code is not human subjects, this comparison illustrates that there is no simple or obvious way forward for providing developers the support needed for the identification and processing of privacy-sensitive information.

E. Results Summary

In this work, we proposed four research questions to address the larger issue of feasibility and accuracy of human participants' classifying code, which contains variables that indicate that PII is being processed. We measured these by the ability of participants to classify code snippets containing variables that reference PII as sensitive. Here, we summarize the results of these.

H1: There is agreement among programmers across a wide range of technical expertise as to which code snippets and variables have indicators that they are processing sensitive data.

We found that there is little agreement beyond three common variables about the sensitivity of code. Our results require that we reject H1.

H2: Technical expertise does not influence the identification of variables in code snippets as being indicators of processing sensitive data.

Participants with little knowledge fail to identify several PII categories, but they put more emphasis on Account/Identifier Identifiers. Most expert participants agree only on three categories (Individual Identifier, Employment, Unique Device ID) and disagree more in other categories. Our results require that we reject H2. This implies that increased technical education will not resolve the challenge of identification of data as sensitive (i.e., defined as PII by the GDPR).

H3: Is the difference in the classification of code more a function of expertise than an individual variable?

We applied an Analysis of Variance(ANOVA) test for the variable sensitivity score based on per category and per group. We found that whether the variable is sensitive or not is highly correlated with the category it belongs to, with a little variance in the classification of variables within a group. However, when the overall classification levels are compared across groups, the difference is highly significant. The relationship between the expertise of participants and the category of data

is suggestive; however, we could not reject H3. This could be seen as implying not a misunderstanding of the code but instead the sensitivity of the underlying variable, despite the differences in technical expertise.

H4: The categorization of code snippets is similar between LLMs and human participants.

LLMs failed at categorization with higher false positive rates than human participants. The categories where human participants erred were different from those where LLMs failed. Our results require that we reject H4. However, these differences suggest that interactions that leverage the strengths of both human and artificial intelligence may be a promising way forward.

V. Discussion

These results contribute to understanding the perceptions of developers, particularly in terms of the status quo of developers in terms of privacy perceptions. Of course, this is true only to the degree to which both our coding samples and our participants are representative, or at least illustrative. We found that additional expertise does not result in more accurate assessments; however, it does appear to create a systematic difference in the classification of data. The different groups had very different overall classifications of more experts, with experts having a mean estimate of (1.43) and least experts having a mean estimate of (1.24). Further, greater expertise resulted in less consensus about sensitivity. Thus increasing the expertise of programmers as a way to increase privacy awareness and sensitivity is not supported by our results.

One possible way forward might be the use of queries to LLMs but our initial results for these are less promising. Another possible next step is the development of privacy APIs, of which mobile app permissions might be considered both examples and sources of concern for the limitations of such an approach. Improved secure by default APIs have been identified as a possible way forward, particularly by Green and Smith [15]. Yet while the general applicability of usability principles to security APIs has been widely explored, the result has not been a theory that is both comprehensive and clearly actionable [43]. A further complication is that cryptographic security can be subject to general agreement, and privacy sensitivity and compliance vary widely.

Developing support for coders to identify privacy-sensitive operations or data in general-purpose code is an open challenge. This challenge will be exacerbated by the proliferation of AI, which will have more opaque models where determining the inclusion of sensitive data is particularly difficult. Note that this is an effort to address this, as one of the factors in the Software Bill of Materials (SBOM) in the ISO Standard from the Software Package Data Exchange (SPDX) AI & ML Working Group [44] is privacy sensitivity. Consider that all software provided to the US Department of Defense, and potentially used by the US government, will soon be required to document dependencies using an SBOM. This requirement, if expanded to include privacy sensitivity, requires improved support for the development and analysis of code for sensitivity. Should labels or SBOMs increase the demand for privacy-aware and secure code, there may be a demand for support for such development.

Our experiment was focused on Python. Additional languages may be more or less transparent, and a comparison of clarity of code or obfuscated code is beyond the scope of this work. The rules and standards can be applied to other coding languages in the future, and any tool developed to assist programmers and compliance personnel in identifying privacy sensitivity would have a larger scope than our human subjects research.

After our experiment design was complete, Privado released its code for finding privacy-sensitive code open source [45]. This validated our selection of variables and the choice to use embeddings. Privado searches for explicit strings as variable names, while our experiment included code embeddings. We found examples where Privado would misclassify code based only on the variable name. For example, the variable name “mask” was included in two snippets in our experiment. In one case, it was for analysis of biometrics, in the other, the variable was for masking IP addresses. Based on the context, “mask” may either indicate the use of highly sensitive biometric data for facial recognition, or it could indicate that the programmer is stripping IP addresses so that the data is neither uniquely identifiable nor sensitive.

VI. Future Work

In future work, we seek to both deepen our investigation, and design tools which integrate our current level of understanding. In terms of expanding our understanding, we will use qualitative measures to investigate how more or less expert developers group variables or use rules of thumb; and how their reasoning is reflected in their choices. Possible explorations include think-aloud experiments as developers categorize personal information as more or less sensitive, or interviews about their perspectives on privacy after the categorization.

Concurrently, we will seek to build upon warnings and visualizations to assist developers in identifying compliance and privacy risks based on the processing of PII in code. Visualizations have been found effective in a range of security challenges, and this is our first area of exploration. Specifically, our efforts moving forward will build on previous work on system configuration and access control. As with privacy, access control is nuanced and governed by policies that are difficult to implement in practice. An early study on the mitigation of human error in access control and system configuration was done by Maxion and Reeder [46]. They found that visualization improves the rate of completing the configuration tasks by a factor of three. The error in these completed activities was also reduced by up to 94%. An access control interaction evaluated by Vaniea et al. [47] mapped policy decisions into access control rules, suggesting that mapping privacy policy to code is a promising way forward. Similarly, another study that reaffirms the importance of visualization is the results by Andalibi, who documented significant improvement in identifying access control risks by providing a visualizer for manufacturer’s recommended access control policies [48]. Work by Xu and colleagues [49] investigated the uncertainties in access control policy decisions, and found that a lack of feedback forced the administrators who intend to resolve access control conflicts into a trial and error mode. We will explore if there is similar

potential for the efficacy of feedback in assisting developers to avoid privacy risks.

VII. Conclusion

There is an increase in the number of legal controls on data processing both within the US and across the globe. This work asked the core question of whether it is feasible to ask those with technical expertise to be able to identify privacy-sensitive code using two well-known privacy regulations: GDPR and CCPA. We found that even experts require some assistance in identifying sensitive data. In fact, greater expertise correlated with less consensus on which code snippets indicated the risk of processing privacy sensitive data. As privacy regulations, and thus differences between privacy regulations, there is no reason to expect that greater complexity and diversity will result in greater consensus. Thus, the work motivates future work in two dimensions. First is the need for automated guidance in evaluating the compliance of code for a given regulatory environment. Second, the development of interactions or compliance risk communication to enable programmers to know the level of risk of including a particular snippet or dependency. Third, the potential for creating interactions that inform non-technical decision-makers, such as those with expertise in the law, to collaborate with programmers to annotate code as more or less sensitive. As programming becomes more accessible, the use case for automated privacy compliance checking becomes stronger.

Data Availability

In terms of data availability, we release our tools, including the card sorting code and our set of code snippets, to enable the reproduction of the experiment. The code and dataset can be found here:https://github.com/xinyao1108/Cardsorting_ProgrammerPerception. As this is human subjects research, sharing the data would violate the IRB by risking the privacy of participants.

Acknowledgment

This research was supported in part by CTIA and the Comcast Innovation Fund. We would like to acknowledge Prof. Tatiana Ringenberg's contributions. Undergraduate researcher Paul Forst of Indiana University replicated the statistical analysis. Amol Sangar provided essential contributions to the coding and optimization of the card-sorting experimental harness. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of Comcast, CTIA nor Indiana University. We acknowledge support from the US Department of Defense [Contract No. W52P1J2093009]. This material is based upon work supported by the U.S. Department of Homeland Security under Grant Award Number 17STQAC00001-07-00." The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U.S. Department of Homeland Security nor the US Department of Defense.

References

[1] J. Parker. (2023) New york lifts ban on biometric technologies in k-12 schools. [Online]. Available: https://www.securityinfowatch.com/access-identity/biometrics/news/53074206/new-york-lifts-ban-on-biometric-technologies-in-k12-schools

[2] T. Kelley and B. I. Bertenthal, "Attention and past behavior, not security knowledge, modulate users' decisions to login to insecure websites," *Information & Computer Security*, vol. 24, no. 2, pp. 164–176, 2016.

[3] T. Kelley, M. J. Amon, and B. I. Bertenthal, "Statistical models for predicting threat detection from human behavior," *Frontiers in psychology*, vol. 9, p. 466, 2018.

[4] A. P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner, "Android permissions demystified," in *ACM Conference on Computer and Communications Security (CCS)*. "ACM SIGSAC", 2011, pp. 627–638.

[5] M. Backes, S. Bugiel, E. Derr, P. McDaniel, D. Octeau, and S. Weisgerber, "On demystifying the android application framework:{Re-Visiting} android permission specification analysis," in *25th USENIX security symposium (USENIX security 16)*, 2016, pp. 1101–1118.

[6] Y. Xiao, Z. Li, Y. Qin, X. Bai, J. Guan, X. Liao, and L. Xing, "Lalaine: Measuring and characterizing non-compliance of apple privacy labels at scale," 2022.

[7] Y. Nan, X. Wang, L. Xing, X. Liao, R. Wu, J. Wu, Y. Zhang, and X. Wang, "Are you spying on me?{Large-Scale} analysis on {IoT} data exposure through companion apps," in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 6665–6682.

[8] K. Shilton and D. Greene, "Linking platforms, practices, and developer ethics: Levers for privacy discourse in mobile application development," *Journal of Business Ethics*, vol. 155, pp. 131–146, 2019.

[9] A. Senarath and N. A. Arachchilage, "Why developers cannot embed privacy into software systems? an empirical investigation," in *Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018*, 2018, pp. 211–216.

[10] S. A. Horstmann, S. Domiks, M. Gutfleisch, M. Tran, Y. Acar, V. Moonsamy, and A. Naiakshina, ""those things are written by lawyers, and programmers are reading that." mapping the communication gap between software developers and privacy experts," *Proceedings on Privacy Enhancing Technologies*, vol. 1, pp. 151–170, 2024.

[11] R. Balebako, A. Marsh, J. Lin, J. Hong, and L. F. Cranor, "The privacy and security behaviors of smartphone app developers," in *Workshop on Usable Security*. Citeseer, 2014, pp. 1–10.

[12] T. Li, K. Reiman, Y. Agarwal, L. F. Cranor, and J. I. Hong, "Understanding challenges for developers to create accurate privacy nutrition labels," in *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, 2022, pp. 1–24.

[13] C. Wijayarathna and N. A. G. Arachchilage, "Am i responsible for end-user's security? a programmer's perspective," in *4th Workshop on Security Information Workers*, 2018.

[14] R. Balebako, R. Shay, and L. F. Cranor, "Is your inseam a biometric? a case study on the role of usability studies in developing public policy," *USEC: NDSS Colocated Usable Security Symposium*, vol. 14, 2014.

[15] M. Green and M. Smith, "Developers are not the enemy!: The need for usable security apis," *IEEE Security & Privacy*, vol. 14, no. 5, pp. 40–46, 2016.

[16] Y. Acar, S. Fahl, and M. L. Mazurek, "You are not your developer, either: A research agenda for usable security and privacy research beyond end users," *2016 IEEE Cybersecurity Development (SecDev)*, pp. 3–8, 2016.

[17] Y. Acar, C. Stransky, D. Wermke, M. L. Mazurek, and S. Fahl, "Security developer studies with {GitHub} users: Exploring a convenience sample," in *Thirteenth Symposium on Usable Privacy and Security (SOUPS 2017)*. "USENIX", 2017, pp. 81–95.

[18] M. Meli, M. R. McNiece, and B. Reaves, "How bad can it git? characterizing secret leakage in public github repositories," in *NDSS*, 2019.

[19] S. Website. (2013) Github kills search after hundreds of private keys exposed. [Online]. Available: https://it.slashdot.org/story/13/01/25/132203/github-kills-search-after-hundreds-of-private-keys-exposed

[20] slashdot. (2023) aws urges devs to scrub secret keys from github. [Online]. Available: https://developers.slashdot.org/d

[21] P. Datta, "Hannibal at the gates: Cyberwarfare & the solarwinds sunburst hack," *Journal of Information Technology Teaching Cases*, vol. 12, no. 2, pp. 115–120, 2022.

[22] F. Fischer, K. Böttinger, H. Xiao, C. Stransky, Y. Acar, M. Backes, and S. Fahl, "Stack overflow considered harmful? the impact of copy&paste on android application security," in *Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 121–136.

[23] J. Schoenherr, "Whose privacy, what surveillance? dimensions of the mental models for privacy and security," *IEEE Technology and Society Magazine*, vol. 41, no. 1, pp. 54–65, 2022.

[24] A. Naiakshina, A. Danilova, C. Tiefenau, M. Herzog, S. Dechand, and M. Smith, "Why do developers get password storage wrong? a qualitative usability study," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 311–328.

[25] O. Tripp, S. Guarnieri, M. Pistoia, and A. Aravkin, "Aletheia: Improving the usability of static security analysis," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, 2014, pp. 762–774.

[26] G. Pellegrino, C. Tschürtz, E. Bodden, and C. Rossow, "jäk: Using dynamic analysis to crawl and test modern web applications," in *Research in Attacks, Intrusions, and Defenses: 18th International Symposium, RAID 2015, Kyoto, Japan, November 2-4, 2015. Proceedings 18*. Springer, 2015, pp. 295–316.

[27] M. Tahaei and K. Vaniea, ""developers are responsible": What ad networks tell developers about privacy," in *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, 2021, pp. 1–11.

[28] "Github, howpublished = https://github.com/, note = Accessed: 2023-11-28."

[29] M. Kuniavsky, *Observing the User Experience: A Practitioner's Guide to User Research*. Elsevier, 2003.

[30] E. F. Cataldo, R. M. Johnson, L. A. Kellstest, and L. W. Milbrath, "Card Sorting as a Technique for Survey Interviewing*," *Public Opinion Quarterly*, vol. 34, no. 2, pp. 202–215, 01 1970. [Online]. Available: https://doi.org/10.1086/267790

[31] Q. Duan and Z. Tan, "Factors affecting low response effort in online survey tasks for passive stakeholders: Insights from a design ethnography research," in *International Conference on Applied Human Factors and Ergonomics*. Springer, 2020, pp. 402–409.

[32] F. Asgharpour, D. Liu, and L. J. Camp, "Mental models of security risks," in *Financial Cryptography and Data Security*, S. Dietrich and R. Dhamija, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 367–377.

[33] S. Jones and E. O'Neill, "Feasibility of structural network clustering for group-based privacy control in social networks," in *Proceedings of the Sixth Symposium on Usable Privacy and Security*, ser. SOUPS '10. New York, NY, USA: Association for Computing Machinery, 2010. [Online]. Available: https://doi.org/10.1145/1837110.1837122

[34] E. Chin, A. P. Felt, V. Sekar, and D. Wagner, "Measuring user confidence in smartphone security and privacy," in *Proceedings of the Eighth Symposium on Usable Privacy and Security*, ser. SOUPS '12. New York, NY, USA: Association for Computing Machinery, 2012. [Online]. Available: https://doi.org/10.1145/2335356.2335358

[35] J. J. McIntyre, "Balancing expectations of online privacy: Why internet protocol (ip) addresses should be protected as personally identifiable information," *DePaul L. Rev.*, vol. 60, p. 895, 2010.

[36] N. Witzleb and J. Wagner, "When is personal data about or relating to an individual a comparison of australian, canadian, and eu data protection and privacy laws," *Can. J. Comp. & Contemp. L.*, vol. 4, p. 293, 2018.

[37] W. Wiewiórowsk, *The History of the General Data Protection Regulation*. Brussels, Belguim: European Union Data Protection, 2023. [Online]. Available: https://edps.europa.eu/data-protection/data-protection/legislation/history-general-data-protection-regulation_en

[38] A. Schwartz, "Regulating for rationality," *Stanford Law Review*, pp. 1373–1410, 2015.

[39] B. Lurger, C. Vogrincic-Haselbacher, F. Caks, J. Anslinger, I. Dinslaken, and U. Athenstaedt, "Consumer decisions under high information load: How can legal rules improve search behavior and decision quality?" *Available at SSRN 2731655*, 2016.

[40] G. Howells, "The potential and limits of consumer empowerment by information," *Journal of Law and Society*, vol. 32, no. 3, pp. 349–370, 2005.

[41] K. A. Hallgren, "Computing inter-rater reliability for observational data: An overview and tutorial," *Tutorials in quantitative methods for psychology*, vol. 8, no. 1, p. 23, 2012.

[42] J. Wei, X. Wang, D. Schuurmans, M. Bosma, b. ichter, F. Xia, E. Chi, Q. V. Le, and D. Zhou, "Chain-of-thought prompting elicits reasoning in large language models," in *Advances in Neural Information Processing Systems*, vol. 35. Curran Associates, Inc., 2022, pp. 24 824–24 837. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/file/9d5609613524ecf4f15af0f7b31abca4-Paper-Conference.pdf

[43] P. L. Gorski and L. L. Iacono, "Towards the usability evaluation of security apis." *HAISA*, vol. 10, pp. 252–265, 2016.

[44] J. Dev, "Putting privacy on the map," 2023.

[45] Privado. [Online]. Available: https://github.com/Privado-Inc/privado

[46] R. A. Maxion and R. W. Reeder, "Improving user-interface dependability through mitigation of human error," *International Journal of human-computer studies*, vol. 63, no. 1-2, pp. 25–50, 2005.

[47] K. Vaniea, Q. Ni, L. Cranor, and E. Bertino, "Access control policy analysis and visualization tools for security professionals," in *Symposium on Usable Security and Privacy*, 2008, pp. 7–15.

[48] V. Andalibi, J. Dev, D. Kim, E. Lear, and L. J. Camp, "Is visualization enough? evaluating the efficacy of mud-visualizer in enabling ease of deployment for manufacturer usage description (mud)," in *Annual Computer Security Applications Conference*, 2021, pp. 337–348.

[49] T. Xu, H. M. Naing, L. Lu, and Y. Zhou, "How do system administrators resolve access-denied issues in the real world?" in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, 2017, pp. 348–361.

APPENDIX A
EXPERTISE QUESTIONS AND ANSWERS

- Q1: What is the purpose of an X.509 certificate for websites?
  - The certificate provides encryption
  - The certificate protects information
  - The certificate shows the website is registered and valid
  - The certificate actively is secure and safe against malicious stuff, including hackers
  - The website is trustworthy and has proper privacy protection and is accountable for information use
  - I Do Not Know
- Q2: SQL injection is a technique to:
  - Inject a malicious virus into the SQL database engine
  - Inject a security patch into the SQL database engine in response to the discovery of new threats
  - Inject a statement that checks the database integrity through a website
  - Inject root user privileges to a regular user without using the graphical user interface (GUI) of the database
  - Inject a malicious statement into the database through a website
  - I Do Not Know
- Q3: Which option is correct for the difference between a passive and active IDS(Intrusion Detection System)?
  - Passive IDS is software based and active is hardware based

- Passive IDS provides only alerts and active IDS can retaliate by sending malicious code to the attacker
- There are no real differences, they are just brand names
- Passive IDS is included in a Firewall while active IDS is a standalone network component
- Active IDS can reprogram the Firewall and passive IDS does not
- I Do Not Know

- Q4: Without any other changes in the default settings of a web server, what can the motivation for closing port 80 be?
  - Block incoming XMLhttp Request
  - Block File Transfer Protocol daemon
  - Block Hypertext Transfer Protocol daemon
  - Block incoming and outgoing requests from SMB/CIFS clients
  - Block Hypertext Transfer Protocol Secure daemon
  - I Do Not Know

APPENDIX B
VARIABLES AND CATEGORY DATASET

These are the categories of data selected as privacy sensitive from the GDPR and the CCPA.

CATEGORIES	Variables
Location	addr, address, latitude, locs, long
Unique device ID	addrinfo, api_key, gdf_mask, input_ips, pings
Demographics	age, gmap3, mdy, sender, yrmtdy
Internet traffic	browserhistory, his, session_cookie_structure, session_cookie_value
Individual identifier	driver, email, name, password, username
Employment	employees, new_empID, profile, skill
Biometrics	face, face_locations, known_face_encodings
Education	institution
Multimedia data	m3u8_url, MP4ASampleEntryBox, stream
Financial information	online_bank_statement_provider, tax, transaction_id, PAYMENT_TOKEN, paypal_mapping_id

The CCPA lists the following data categories.
(A) "Unique identifier" or "unique personal identifier" means a persistent identifier that can be used to recognize a consumer, a family, or a device that is linked to a consumer or family, over time and across different services, including, but not limited to, a device identifier; an Internet Protocol address; cookies, beacons, pixel tags, mobile ad identifiers, or similar technology; customer number, unique pseudonym, or user alias; telephone numbers, or other forms of persistent or probabilistic identifiers that can be used to identify a particular consumer or device that is linked to a consumer or family.
(B) Any personal information described in subdivision (e) of Section 1798.80.
(C) Characteristics of protected classifications under California or federal law. Race, age, gender, orientation, religion,
(D) Commercial information, including records of personal property, products or services purchased, obtained, or considered, or other purchasing or consuming histories or tendencies.
(E) Biometric information.
(F) Internet or other electronic network activity information, including, but not limited to, browsing history, search history, and information regarding a consumer's interaction with an internet website application, or advertisement.
(G) Geolocation data.
(H) Audio, electronic, visual, thermal, olfactory, or similar information.
(I) Professional or employment-related information.
(J) Education information, defined as information that is not publicly available personally identifiable information as defined in the Family Educational Rights and Privacy Act (20 U.S.C. Sec. 1232g; 34 C.F.R. Part 99).
(K) Inferences drawn from any of the information identified in this subdivision to create a profile about a consumer reflecting the consumer's preferences, characteristics, psychological trends, predispositions, behavior, attitudes, intelligence, abilities, and aptitudes.
(L) Sensitive personal information.

The GDPR identifies the following data categories, stating, 'Sensitive personal information means':
(1) Personal information that reveals:
(A) A consumer's social security, driver's license, state identification card, or passport number.
(B) A consumer's account log-in, financial account, debit card, or credit card number in combination with any required security or access code, password, or credentials allowing access to an account.
(C) A consumer's precise geolocation.
(D) A consumer's racial or ethnic origin, religious or philosophical beliefs, or union membership.
(E) The contents of a consumer's mail, email, and text messages unless the business is the intended recipient of the communication.
(F) A consumer's genetic data.
(2)
(A) The processing of biometric information for the purpose of uniquely identifying a consumer.
(B) Personal information collected and analyzed concerning a consumer's health.
(C) Personal information collected and analyzed concerning a consumer's sex life or sexual orientation.