5-2021

Smart contract security: A practitioners' perspective

Zhiyuan WAN

Xin XIA

David LO
Singapore Management University, davidlo@smu.edu.sg

Jiachi CHEN

Xiapu LUO

See next page for additional authors

Smart Contract Security: a Practitioners' Perspective

Zhiyuan Wan*†, Xin Xia‡‖, David Lo§, Jiachi Chen‡, Xiapu Luo¶, Xiaohu Yang*
*College of Computer Science and Technology, Zhejiang University
‡Faculty of Information Technology, Monash University
§School of Information Systems, Singapore Management University
¶Department of Computing, Hong Kong Polytechnic University
{wanzhiyuan,yangxh}@zju.edu.cn, {xin.xia,jiachi.chen}@monash.edu, davidlo@smu.edu.sg, csxluo@comp.polyu.edu.hk

***Abstract*—Smart contracts have been plagued by security incidents, which resulted in substantial financial losses. Given numerous research efforts in addressing the security issues of smart contracts, we wondered how software practitioners build security into smart contracts in practice. We performed a mixture of qualitative and quantitative studies with 13 interviewees and 156 survey respondents from 35 countries across six continents to understand practitioners' perceptions and practices on smart contract security. Our study uncovers practitioners' motivations and deterrents of smart contract security, as well as how security efforts and strategies fit into the development lifecycle. We also find that blockchain platforms have a statistically significant impact on practitioners' security perceptions and practices of smart contract development. Based on our findings, we highlight future research directions and provide recommendations for practitioners.**

***Index Terms*—Security, Empirical study, Smart contract, Practitioner**

I. Introduction

Blockchain is a distributed ledger that provides an open, decentralized, and fault-tolerant transaction mechanism. Blockchain technology has attracted considerable attention from both industry and academia since it is originally introduced for Bitcoin [40] to support the exchange of cryptocurrency. Blockchain technology evolves to facilitate general-purpose computations with a wide range of decentralized applications. The *Smart contract* technology is one appealing decentralized application that enables the computations on top of a blockchain.

A smart contract is a piece of executable code that runs on a blockchain to enforce the terms of an agreement between untrusted parties. Blockchain technology assures that a smart contract is immutable and contract initiated transactions are autonomously and truthfully executed. There are multiple blockchain platforms that support smart contracts [67], e.g., Ethereum, Hyperledger Fabric, and Corda, with Ethereum being the most prominent platform [4].

During the last decade, smart contracts have been plagued by security incidents, which led to losses reaching millions of dollars [64]. In June 2016, an attacker exploited vulnerabilities in the DAO smart contract to empty out around 4 million Ethers (worth around 50 million dollars). In July 2017, over 150 thousand Ethers (worth over 34 million dollars) had been stolen due to an exploit in widely-used Parity Wallet [47]. In November 2017, over 500 thousand Ethers (worth over 150 million dollars) were frozen due to a vulnerability in the very same wallet [48].

To address the security issues of smart contracts, researchers have proposed a broad range of defense solutions, including language-based security (e.g., [20], [16], [11]), static analysis (e.g., [38], [64], [63]), and runtime verification (e.g., [56]). Vyper [20] removes some of the language functionalities in Solidity to eliminate vulnerabilities and adds new features to support security and readability. In terms of static analysis, Oyente [38] leverages symbolic execution to traverse feasible execution paths on control flow graphs and detect vulnerabilities in smart contracts; Securify [64] defines compliance and violation patterns based on known vulnerabilities; SmartCheck [63] translates smart contract code into an XML-based parse-tree and check the parse-tree against specific XPath patterns. With respect to runtime verification, Sereum [56] uses taint analysis to monitor runtime data flows during the execution of smart contracts for preventing re-entrancy attacks.

Despite numerous efforts in assuring the security of smart contracts, little is known about how software practitioners build security into smart contracts in practice. Thus, we followed a mixed methods approach to investigate the practitioners' perceptions and practices with respect to smart contract security. We started with semi-structured interviews with 13 software practitioners with experience in smart contract development, who have an average of 6.5 years of software professional experience. Through the interviews, we qualitatively investigated the security awareness and practices that our interviewees experienced in smart contract development. We derived 6 competing priorities in smart contract development, a list of 11 security motivators[1] and 9 security deterrents for smart contract practitioners, 5 sources where practitioners acquire security knowledge, 11 security strategies and 11 factors that affect the adoption of security tools. We further performed an exploratory survey with 156 smart contract practitioners from 35 countries across six continents to quantitatively validate the security perceptions and practices that are uncovered in our interviews. The survey respondents work

†Also with PengCheng Laboratory.
‖Corresponding author.

[1]*Security motivators* are the factors that motivate practitioners to address security; on the contrary, *security deterrents* are the factors that deter practitioners from devoting efforts to security [7].

on multiple blockchain platforms, i.e., public blockchains (80), consortium blockchains (49), and private blockchains (20), and hold various job roles, i.e., development (130), testing (3), and project management (16). We investigated the following research questions:

RQ1. What are practitioners' perceptions regarding smart contract security?

85% and 69% of the survey respondents perceive the importance of security and privacy in smart contracts, respectively. The security motivators include practitioners' awareness of importance, workplace environment, and perceived negative consequences of security issues. Meanwhile, the security deterrents include competing priorities in smart contract development and no formal process to address smart contract security.

RQ2. How does security fit into the development lifecycle of smart contracts?

This research question investigates security efforts, security strategies, and the adoption of security tools in smart contract development. On average, security efforts account for 29% of the overall efforts during the development process of smart contracts. To ensure smart contract security, practitioners distribute efforts across different stages in the development lifecycle. They tend to spend significantly more effort towards security at the construction and testing stages than at other stages. In terms of security strategies and tool adoption, 72% of the respondents frequently leverage more than one security strategy. 58% of the respondents frequently used the code reuse strategy. 54% of the respondents frequently adopt security tools, especially the security plugins in integrated development environments.

RQ3. Do blockchain platforms influence practitioners' perceptions and practices on smart contract security?

Blockchain platforms significantly impact security perceptions and practices of practitioners in smart contract development, including security motivators (e.g., the immutability of smart contracts), security deterrents (e.g., the pressure of feature delivery), the amount of security efforts across stages in the development lifecycle (e.g., security efforts at the construction stage), and strategies to address smart contract security (e.g., code review).

Based on the findings, we discuss the disconnect between the high awareness of smart contract security of practitioners and the frequent occurrence of security problems in smart contracts. We also provide practical lessons about code reuse, tool implications, and proactive defense to ensure smart contract security. In addition, we highlight several research avenues across blockchain platforms.

This paper makes the following contributions:

- We perform a mixture of qualitative and quantitative studies to investigate the security perceptions and practices in smart contract development;
- We provide practical implications for practitioners and outlined future avenues of research.
- We provide the interview guide, questionnaire, and survey responses publicly accessible for future investigation by others[2].

The remainder of the paper is structured as follows. Section II briefly reviews related work. In Section III, we describe the methodology of our study in detail. In Section IV, we present the results of our study. We discuss the implications of our results in Section V and threats to the validity of our findings in Section VI. Section VII draws conclusions and outlines avenues for future work.

II. Related Work

A. Security Practices in Software Development

Practitioners work within organizations, teams, communities, and cultures. Previous studies investigated the social factors that could impact various aspects of security practices, e.g., security tool adoption [72], [71], [68]. Organization and team policies are a driving factor to tool adoption [10], though many organizations do not encourage the adoption of security tools. Large organizations make more use of security tools than small ones [72]. Existing tools fail to meet the expectations of practitioners by generating low-quality warning messages [10], interrupting work flow [31], [10], [60], producing excessive false positives [31], [10], and integrating poorly with Integrated Development Environments (IDEs) [31]. In this work, we investigated the factors that impact security practices in smart contract development.

Security is expected to be included in developing high-quality software systems, but is rarely listed as an explicit requirement [51]. Practitioners prioritize functional requirements over security and focus on tasks for which outcomes are easy to measure [72], [51], [6]. Pressures from budget and deadlines can also lead to lowering the priority of security practices [74]. Some organizations attempt to use penetration testing to motivate practitioners, but the motivation is hard to sustain without continuous support [65]. In this work, we investigated how practitioners prioritize security in smart contract development.

Building security in from the start requires a large amount of knowledge. Weir et al. found that enthusiasm about security and motivation to learn are more likely to drive the acquisition of security knowledge for developers than task driven [70]. Alternatively, security experts could act as a roving source of security knowledge, but face challenges to convince others of the importance of security and examine all generated code with limited resources [61]. Practitioners leverage various information sources to gain knowledge on code security, e.g., documentation [39], [2], and Stack Overflow [3]. Acar et al. [3] conducted an empirical study to investigate how the use of information sources impacts code security. They found that developers who use Stack Overflow are more likely to produce functional code, but less likely to write secure code. This paper investigates the involvement of security experts in smart contract development and information sources of smart contract security.

[2]http://doi.org/10.5281/zenodo.4005112

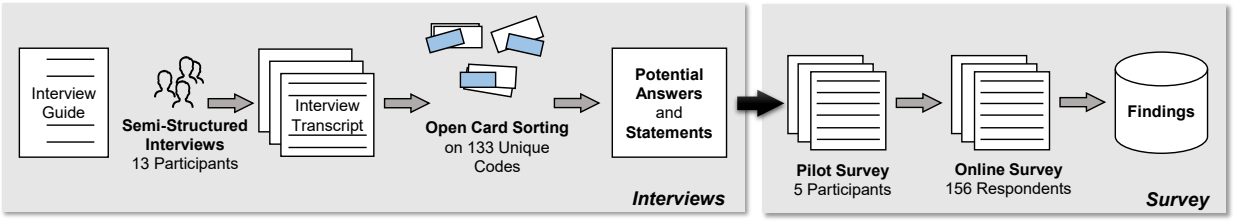

Fig. 1. Research methodology.

In the course of software design and construction, misuse of application programming interfaces (APIs) can introduce security vulnerabilities [19], [21], [24]. Developers incorrectly use an API because they do not conduct an additional check but trust the API to do the right thing [44]. Acar et al. compared the usability of five Python cryptographic APIs and suggested that documentation with examples is more helpful than a simple API [2]. Nadi et al. [39] performed an empirical study on how developers use Java cryptography APIs. They found that developers struggle with Java cryptography APIs and prefer task-based solutions. In addition, developers have difficulty in using security-related APIs, for instance, APIs in Transport Layer Security (TLS) and Security Sockets Layer (SSL) [22], [45]. This paper investigates whether the use of smart contract related APIs may introduce security risks.

B. Smart Contract Security

Security vulnerabilities spread across smart contracts of various blockchain platforms, e.g., Ethereum [30], [38], [56], Hyperledger Frabric [75] and EOS [52]. Security vulnerabilities result from multiple causes, e.g., reentrancy [56], delegate-call injection [53], and integer overflow and underflow [41]. Different programming languages of smart contracts and blockchain architectures lead to different vulnerabilities [9]. In this work, we investigate practitioners' awareness of security vulnerabilities in their smart contracts.

Previous studies proposed a wide range of approaches and tools for securing smart contracts, including recommending best practices in programming smart contracts, implementing specific programming languages, static analysis, and runtime monitoring. For instance, ConsenSys [13] provides extensive best practices for Ethereum smart contract security, including code patterns to learn and pitfalls to avoid. *Vyper* [20] and *Bamboo* [16] provide language-based support to eliminate smart contract vulnerabilities. Static analysis tools leverage symbolic execution [14], [8], [34], [38], [43], [50], abstract interpretation [25], [32], [59], [64], [76], formal verification [26], [5], [27], [29], [28], [49], fuzzing [37], [30], [69] and model checking [63] to identify smart contract vulnerabilities. DappGuard [15] and Sereum [56] monitor the runtime execution of a smart contract to prevent potential exploitations of vulnerabilities. In this work, we investigate the adoption of security strategies and tools of smart contracts in practice and explore the expectations of practitioners.

III. Methodology

Our research methodology followed a mixed methods approach [17], as depicted in Fig. 1. The approach follows a sequential explanatory strategy, involving two phases – a first qualitative phase of interviews, followed by a second quantitative phase of an exploratory survey[3]. The survey builds on the results of the interviews. Specifically, we collected data from two sources: (1) We interviewed 13 software practitioners with experience in smart contract development and derived a list of statements and potential answers for survey questions from the results of interviews; (2) We surveyed 156 respondents with experience in smart contract development. To preserve the anonymity of participants, we anonymized all items that constitute of Personally Identifiable Information (PII) before analyzing the data, and further considered aliases as PII throughout our study (e.g., refer to the interviewees as P1 - P13).

A. Interviews

The left part of Fig. 1 describes the process of interviews.

1) Protocol: The first author conducted a series of face-to-face interviews with 13 software practitioners with experience in smart contract development. Each interview took 30-45 minutes. The interviews were semi-structured and made use of an *interview guide*[4]. To develop the interview guide, we obtained an initial set of open-ended questions through brainstorming within the authors of this paper, focusing on practitioners' perceptions and practices concerning smart contract security.

The interview comprised three parts. In the first part, we asked some demographic questions about the experience of the interviewees in smart contract development. The questions covered various aspects of experience, including programming, design, testing, and project management.

In the second part, we asked open-ended questions about the security awareness and practices of smart contract development. The purpose of this part was to allow the interviewees to speak freely about their opinions and experience without the interviewer biasing their responses.

In the third part, we asked the interviewees to discuss the sources where they obtain security-related knowledge, as well as strategies and tools that they have used for security assurance of smart contracts in the practices.

[3]The interviews and survey were approved by the relevant institutional review board (IRB). Participants were instructed that we wanted their opinions; privacy and sensitive resources were not explicitly mentioned

[4]Interview guide online: http://doi.org/10.5281/zenodo.4005112

At the end of each interview, we thanked the interviewee and briefly informed her of our next plans.

2) Participant Selection: We recruited full-time software practitioners with experience in smart contract development from blockchain companies (e.g., Hyperchain[5]), IT companies (e.g., Alibaba) and open-source smart contract projects. Interviewees were recruited by emailing our contact in each company or project, who disseminated the news of our study to their colleagues. Volunteers would inform us if they were willing to participate in the study with no compensation. With this approach, 13 volunteers with varied experience in years contacted us – 7 interviewees from four companies and 6 interviewees from three open-source projects. In the remainder of this paper, we denote these 13 interviewees as P1 to P13. These 13 interviewees have an average of 6.5 years of professional experience in software development (min: 3, max: 13, median: 6.5, sd: 2.7), and an average of 2.3 years in smart contract development (min: 1, max: 5, median: 2, sd: 1.1). Table I summarizes the number of interviewees who perceived themselves as having "extensive" experience (in comparison to "none" and "some" experience) in a particular role.

3) Data Analysis: We conducted a thematic analysis [18] to process the recorded interviews by following the steps below:

Transcribing and Coding. After the last interview was completed, we transcribed the recordings of the interviews, and developed a thorough understanding by reviewing the transcripts. The first author read the transcripts and coded the interviews using NVivo qualitative analysis software [1].

To ensure the quality of codes, the second author verified initial codes created by the first author and provided suggestions for improvement. After incorporating these suggestions, we generated a total of 427 cards that contain the codes - 30 to 41 cards for each coded interview. After merging the codes with the same words or meanings, we have a total of 133 unique codes.

Open Card Sorting. Two of the authors then separately analyzed the codes and sorted the generated cards into potential themes for thematic similarity (as illustrated in LaToza et al.'s study [35]). The themes that emerged during the sorting were not chosen beforehand. We then use the Cohen's Kappa measure [12] to examine the agreement between the two labelers. The overall Kappa value between the two labelers is 0.76, which indicates substantial agreement between the labelers. After completing the labeling process, the two labelers discussed their disagreements to reach a common decision. To reduce bias from the two authors sorting the cards to form initial themes, they both reviewed and agreed on the final set of themes. Eventually, we derived 6 competing priorities, a list of 11 security motivators and 9 security deterrents, 5 sources of security knowledge, and 11 security strategies, and 11 factors that affect the adoption of security tools.

B. Survey

The right part of Fig. 1 describes the process of our online survey.

[5]https://www.hyperchain.cn/en

TABLE I
NUMBER OF INTERVIEWEES WITH "EXTENSIVE" EXPERIENCE IN A PARTICULAR ROLE.

Role	Smart Contract	non-Smart-Contract
Programming	10	12
Design	8	6
Management	3	4
Testing	3	3

1) Protocol: We conducted an IRB-approved anonymous online survey with professional smart contract practitioners. The survey aims to validate and quantify the observations from our interviews. We followed Kitchenham and Pfleeger's guidelines for personal opinion surveys [33] and used an anonymous survey to increase response rates [66]. A respondent has the option to specify that she prefers not to answer or does not understand the description of a particular question. We include this option to reduce the possibility of respondents providing arbitrary answers.

Recruitment of Respondents. To recruit respondents from the population of smart contract practitioners, we spread the survey to a broad range of companies from various locations worldwide. No identifying information was required or gathered from our respondents. To get a sufficient number of respondents from diverse backgrounds, we followed a multi-pronged strategy to recruit respondents:

- We contacted professionals from blockchain companies and IT companies that launch blockchain projects around the world and asked their help to disseminate our survey within their organizations. Specifically, we sent emails to our contacts in Alibaba, Baidu, Hengtian, Hyperchain, IBM, Morgan Stanley, and other companies, encouraging them to disseminate our survey to some of their colleagues who have experience in smart contract development. By following this strategy, we aimed to recruit respondents working with smart contracts in the industry from diverse organizations.
- We sent an email with a link to the survey to 1,986 practitioners who contributed to 12 blockchain repositories that support smart contracts (e.g., *ethereum/go-ethereum*, *EOSIO/eos* and *hyperledger/fabric*) and 580 smart contract repositories (e.g., *ethereum/solidity* and *EOSIO/eosio.contracts*) hosted on GitHub and solicited their participation. We aimed to recruit open-source practitioners who have smart contract experience in addition to professionals working in the industry.

  Out of these emails, eight emails received automatic replies notifying us of the absence of the receiver; two emails indicated the receivers left the original organizations; four receivers replied that they only have experience in blockchain but not smart contract development.

2) Survey Design: The survey includes different types of questions, e.g., multiple-choice and free-text answer questions. The potential answers and statements of multiple-choice questions were derived from the results of our interviews. For these questions, we include an "*I don't know*" option in case some statements are not applicable to the experience of respondents,

or respondents had a poor understanding of the statements.

The survey consists of four sections, grouping questions by topic to minimize the cognitive load on participants and allow them to consider the topic more deeply [36]. Specifically, the following four sections have been captured in the survey (the complete questionnaire is available online as supplemental material[6]):

Demographics. We collected demographic information about the respondents to allow us to (1) filter respondents who may not understand our survey (i.e., respondents without any experience in smart contracts), (2) breakdown the results by groups (e.g., public, consortium, and private blockchains). Specifically, we asked two questions:

- *Do you have experience with smart contracts?*
- *What best describes the **primary blockchain platform** that you currently work on*?

In terms the second question, we provided four options for primary blockchain platforms, including (1) *public blockchain*, (2) *consortium blockchain*, (3) *private blockchain*, and (4) *other*.

Based on the selections of respondents, we could exclude invalid responses and divide the survey respondents into three groups. To focus the respondents' attention on a particular blockchain platform in the survey, they were explicitly asked to answer each following question with respect to their experience with the *primary blockchain platform* they specified.

We received a total of 203 responses, and further excluded 46 responses made by respondents who claimed that they do not have experience in smart contract development. We also excluded one response made by a respondent who described her job role as sales. In the end, we had a set of 156 valid responses. The 156 respondents reside in 35 countries across six continents as shown in Fig. 2. The top two countries in which the respondents reside are China (61) and the United States (16). The respondents have an average of 6.3 years of professional experience (min: 0.5, max: 40, median: 4, sd: 6.9), with an average of 2 years of experience in smart contract development (min: 0.1, max: 6, median: 2, sd: 1.4). Our survey respondents are distributed across different demographic groups (job roles and primary blockchain platforms) as shown in Fig. 3. Seven respondents who selected *Other* as their primary blockchain platforms and explained that they simultaneously work on more than one blockchain. We excluded the responses of the seven respondents from any comparisons between groups of different blockchains.

Perceptions on Smart Contract Security. This section investigates practitioners' perceptions of smart contract security, specifically, the importance of security, awareness of security problems, as well as the motivators and deterrents to smart contract security.

Security Practices in Smart Contract Development. This section focused on security practices in smart contracts, including practitioners' efforts towards security, their strategies for achieving security, and tools for securing smart contracts.

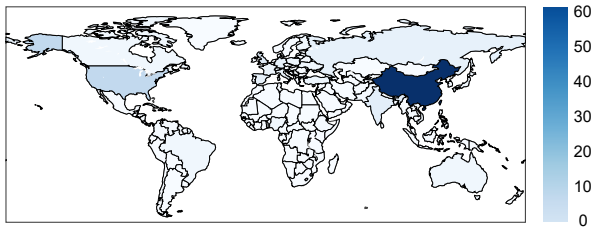

Fig. 2. Countries in which survey respondents reside. The darker the color is, the more respondents reside in that country. The legend indicates the number of respondents.

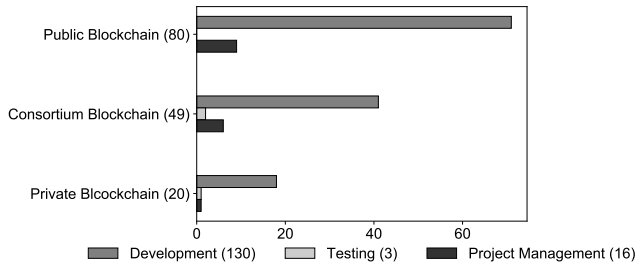

Fig. 3. Survey respondents demographics. The number indicates the count of each demographic group.

More details about the questions and format are available in Section IV, along with the corresponding results.

We piloted the preliminary survey with a small set of smart contract practitioners who were different from our interviewees and survey takers. We obtained feedback on (1) whether the length of the survey was appropriate, and (2) the clarity and understandability of the terms. We made minor modifications to the preliminary survey based on the received feedback and produced a final version. Note that the collected responses from the pilot survey are excluded from the presented results in this paper.

To support respondents from China, we translated our survey to Chinese before publishing the survey. We chose to make our survey available both in English on Google Forms, and in Chinese on a popular survey website in China[7]. The reason is that English is an international lingua franca, and Chinese is the most spoken language. We expected that a large number of our survey recipients are fluent in one of these two languages. We carefully translated our survey to make sure there exists no ambiguity between English and Chinese terms in our survey. Also, we polished the translation by improving clarity and understandability according to the feedback from our pilot survey.

3) Data Analysis: We analyzed the survey results based on the question types. For multiple-choice questions, we reported the percentage each option is selected. In terms of open-ended questions, we followed an inductive approach in which two authors separately performed open card sorting and regularly discussed emerging themes until an agreement was reached.

Factor Analysis. To identify meaningful clusters of closely related information, we used factor analysis to analyze the Likert-scale ratings of the statements with respect to the security motivators and security deterrents in smart contract

[6]Questionnaire Online: http://doi.org/10.5281/zenodo.4005112

[7]https://www.wjx.cn

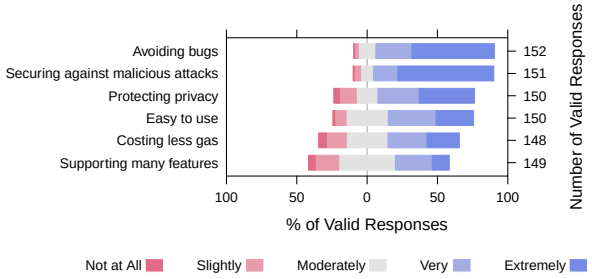

Fig. 4. Importance of Different Requirements.

development. Specifically, we used principal axis factor analysis in the `psych` R library [54] to group related information with a cut-off point of $|0.4|$ for factor loading. We used the `fa.parallel` function in the `psych` R library to select the optimal number of factors for factor analysis. Thus, we reduced a large set of variables into a smaller set (*factors*) while retaining the majority of original information [62].

Comparison. We classified our respondents into different groups based on their primary blockchain platforms (i.e., public, consortium, and private blockchains), and compared the survey results of different groups of respondents. For instance, we used the Wilcoxon rank-sum test for Likert-scale answers to perform the comparison. All statistical tests assumed a p-value < 0.05 as a significant level. Bonferroni correction was applied to adjust p-values in multiple comparisons.

IV. Results

We explain the results of three research questions that investigate smart contract security from the perspective of practitioners.

A. RQ1: Perceptions of Smart Contract Security

In RQ1, we explored practitioners' priorities in smart contract development, what motivates them and deters them to address smart contract security, and their experience of security problems. To understand practitioners' priorities in smart contract development, we presented our respondents with six statements that describe the requirements of smart contracts. Respondents ranked the importance of each requirement on a 5-point Likert scale (*not at all*, *slightly*, *moderately*, *very*, *extremely*). To explore what drives practitioners to address smart contract security, we presented our respondents with a list of 11 statements that describe potential security motivators and 9 statements that explain potential security deterrents. Respondents indicated their level of agreement with each statement on a 5-point Likert scale (*strongly disagree*, *disagree*, *neutral*, *agree*, *strongly agree*). In addition, we asked the respondents to report whether their smart contracts have ever experienced a security problem as well as the sources where they gain security knowledge.

Importance of Security. Fig. 4 shows respondents' ratings of the importance of various requirements in their smart contracts. In addition to avoiding bugs, 85% and 69% of the respondents considered security and privacy *very* or *extremely* important, respectively. The ratings were higher than the requirement of costing less gas in smart contracts.

TABLE II
Factor analysis of motivators to smart contract security.

Variables (Motivators as presented in the survey)	Factor loading
Awareness of Importance	
[M7] I see software security as my responsibility.	0.76
[M6] Software security is a shared responsibility by all those involved in the development lifecycle.	0.71
[M8] I care about my users' experience in security and privacy.	0.68
[M1] Software security is in my company's culture.	0.51
Workplace Environment	
[M3] My company is audited for smart contract security by an external entity.	0.73
[M2] My company mandates security practices in smart contract development.	0.61
Perceived Negative Consequences	
[M4] My company would lose customers in case of a security breach.	0.78
[M5] Security breaches would hurt my company's reputation.	0.66
[M9] Customers would lose money in case of a security breach.	0.56
Motivators not belonging to any factor	
[M10] The deployed smart contracts are immutable.	
[M11] It is challenging to detect and trace attacks on smart contracts deployed to blockchains.	

Security Motivators. We asked the respondents "*I care about smart contract security because ...*" and presented the 11 potential motivators for smart contract security. As shown in Fig. 5, the top two security motivators are respondents' internal motivations[8], i.e., to protect their users and the reputation of their companies. Meanwhile, external motivations[9] are reportedly less motivating, i.e., the immutability of smart contracts and external auditing.

We used factor analysis to cluster the 11 motivators into three factors as shown in Table II. Two motivators did not conform to any particular factor. We named the factors as *awareness of importance*, *workplace environment* and *perceived negative consequences*. Out of the three factors, *workplace environment* is the only external motivation.

Security Deterrents. Respondents generally opposed statements that imply deferring or ignoring security, as suggested by the longer red bars in comparison with blue bars (Fig. 6). The top two deterrents of smart contract security are a lack of awareness of security attacks, followed by a formal process.

Our factor analysis combined 8 out of the 9 deterrents into two factors; 1 deterrent did not correspond to any particular factor (Table III). The first factor *competing priorities and no process* describes how a lack of security can arise from systemic causes within an organization or a team. The other factor *irrelevance of security* characterizes the personal-level awareness of security risks that can deter practitioners from addressing smart contract security.

Experiencing Security Problems. 40% of our respondents reported that they had experienced at least one out of three potential security problems, i.e., vulnerabilities in unshipped

[8]Internal motivation: people stand behind a behavior out of their interests and values [58].

[9]External motivation: people do a behavior for reasons external to the self [58].

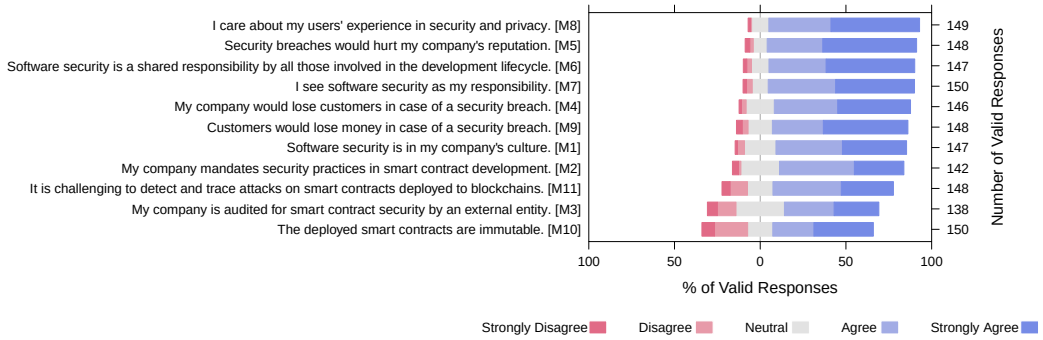

Fig. 5. Motivators of smart contract security.

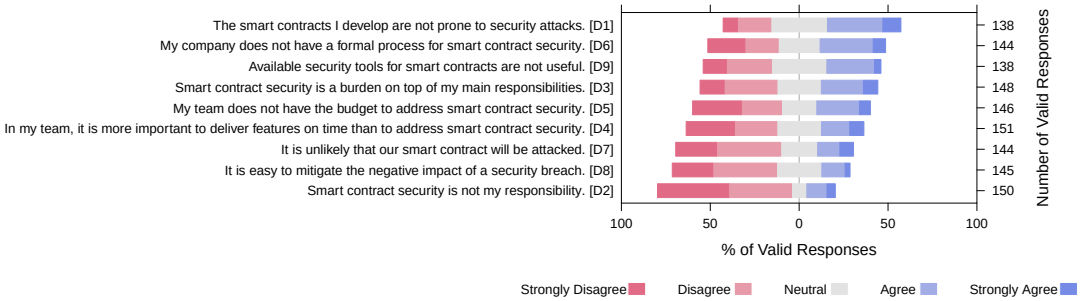

Fig. 6. Deterrents of smart contract security.

TABLE III
FACTOR ANALYSIS OF DETERRENTS TO SMART CONTRACT SECURITY.

Variables (Deterrents as presented in the survey)	Factor loading
Competing Priorities and No Process	
[D5] My team does not have the budget to address smart contract security.	0.86
[D6] My company does not have a formal process for smart contract security.	0.79
[D4] In my team, it is more important to deliver features on time than to address smart contract security.	0.74
[D2] Smart contract security is not my responsibility.	0.52
[D3] Smart contract security is a burden on top of my main responsibilities.	0.43
Irrelevance of Security	
[D7] It is unlikely that our smart contract will be attacked.	0.64
[D1] The smart contracts I develop are not prone to security attacks.	0.58
[D8] It is easy to mitigate the negative impact of a security breach.	0.56
Deterrents not belonging to any factor	
[D9] Available security tools for smart contracts are not useful.	

code, vulnerabilities in shipped code, and security breaches. Identification of vulnerable code before smart contracts were shipped was the most frequently reported (22%) security problem in our survey. 19% of the respondents indicated that vulnerabilities were discovered in shipped smart contracts. 10% reported that their smart contracts experienced a security breach. We note that these numbers are not mutually exclusive; 10% of the respondents reported multiple security problems.

Sources of Security Knowledge. Official forums of blockchain platforms (60%), research papers (53%), question and answer websites (47%) are the top three most popular sources for respondents to acquire security knowledge about smart contracts. We note that these numbers are not mutually exclusive; 74% of our respondents use more than one source to gain knowledge of smart contract security.

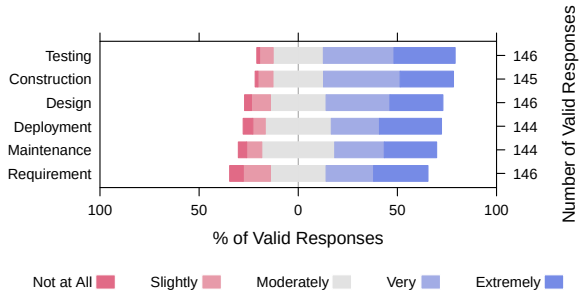

Fig. 7. Security efforts across stages in development lifecycle.

B. RQ2: Security Practices in Smart Contract Development

The survey had several questions exploring the efforts and strategies that development teams employ to ensure smart contract security. 44% of our respondents received support from professional security experts.

Efforts towards Security. Respondents reported the percentage of efforts directed towards security out of the overall efforts in the development lifecycle of smart contracts. They also reported to what extent security was considered for each stage in the development lifecycle (i.e., requirement, design, construction, testing, deployment, and maintenance).

Our respondents indicated that, on average, security efforts account for 29% (min: 0%, max: 100%, median: 20%, sd: 26%) of the overall efforts in smart contract development. 14 respondents indicated that their teams do not spend any effort on security.

We used the Wilcoxon rank sum test to determine if the distribution of security efforts statistically significantly differs

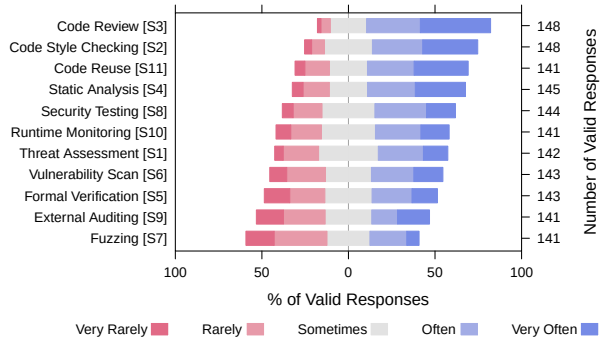

Fig. 8. Strategies for handling smart contract security.

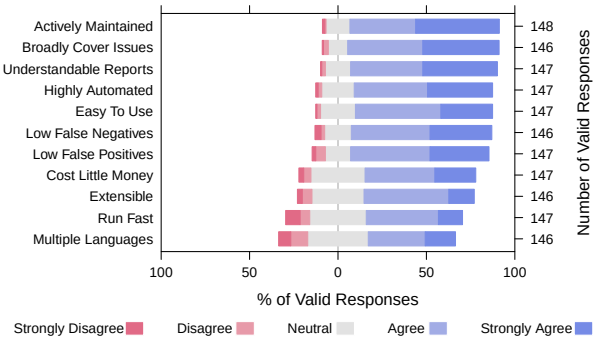

Fig. 9. Factors that affect adoption of security tools.

across different stages in the process. As shown in Fig. 7, security effort at the testing stage was statistically significantly higher than that at the requirement (p-value = 0.01) and maintenance stages (p-value = 0.04). Security effort at the construction stage was statistically significantly higher than that at the requirement stage (p-value = 0.05). Although our interviewees (P4 and P6) mentioned that "*we try to get it (security) right from the beginning*", security effort at the requirement stage was ranked at the bottom across different stages in the development lifecycle.

Strategies to Address Smart Contract Security. We provided a list of 11 statements that describe potential security strategies, and asked the respondents to rate each statement from the list on a 5-point Likert scale (*very rarely*, *rarely*, *sometimes*, *often*, *very often*).

Our respondents combine various strategies to address smart contract security in practice. 72% of our respondents frequently leverage more than one security strategy in smart contract development. As shown in Fig. 8, code review is the most frequently used security strategy – 72% of our respondents indicated that they *often* or *very often* rely on code review to address smart contract security. 61% and 58% of the respondents *often* or *very often* do code style checking and reuse code from reliable sources, respectively. Only 28% of the respondents *often* or *very often* integrate fuzzing into the development lifecycle.

A total of 24 respondents provided free-form text comments regarding other security strategies they use in practice. Out of the 24 respondents, 11 drilled down the aforementioned strategies; the other 13 respondents identified additional strategies (followed by their corresponding frequency) as follows:

- Security by Design (5): "*Security concerns should be built into the framework and exposed via documented, developer-friendly APIs so that good security is easy and bad security is hard.*"
- Programming Languages (3): "*... use the most stable version of Solidity avoiding the latest one.*"
- Dependency Management (2): "*Dependency management to ensure we're using recent versions.*"
- Learning from Past Experiences (2) "*... failure code of others in the past.*"
- Seeking Support from Experts (1): "*... ensured by cryptography designing together with experts.*"

Tools to Address Smart Contract Security. We further investigated the adoption of security tools for smart contracts. 54% of our respondents indicated that they frequently adopt security tools in smart contract development. Security plug-ins in Integrated Development Environments (IDE) are the most popular security tool for smart contracts – 45% of our respondents indicated that they *often* or *very often* rely on security plugins in IDEs to address smart contract security. We further investigated how frequently the security tools for smart contracts have been adopted in practice. 19%, 12%, 14% and 12% of the respondents reported that they *often* or *very often* use *Mythril* [14], *Oyente* [38], *SmartCheck* [63] and *Slither* [23], respectively.

In addition, we investigated what factors affect the adoption of security tools in smart contract development. We provided a list of 11 statements that describe potential factors and asked the respondents to rate each statement from the list on a 5-point Likert scale (*strongly disagree*, *disagree*, *neutral*, *agree*, *strongly agree*). As shown in Fig. 9, active maintenance is the most important factor in the adoption of security tools (85% *agree* or *strongly agree*). 86% of the respondents *agree* or *strongly agree* that coverage of security issues affects the adoption of security tools.

C. RQ3: Effect of Blockchain Platforms on Security Perceptions and Practices

In RQ3, We explore whether blockchain platforms influence security motivators and deterrents to smart contract security, as well as security efforts across different stages and strategies towards smart contract security.

We summarize the results of comparisons in Table IV. The *Statement* column shows the statements presented to respondents in the survey. These statements describe the motivators, deterrents, stages in the development lifecycle, and security strategies. The following column indicates the labels we used to identify statements throughout the paper.

The *Effect Size* column indicates the difference between *Public Blockchain* and *Consortium Blockchain* in the first subcolumn, *Public Blockchain* and *Private Blockchain* in the second subcolumn, and *Consortium Blockchain* and *Private Blockchain* in the third subcolumn. We use Cliff's delta to measure the magnitude of the differences because Cliff's delta is reported to be more robust and reliable than Cohen's delta [57]. Cliff's delta represents the degree of overlap

TABLE IV

IMPACT OF BLOCKCHAIN PLATFORMS ON SMART CONTRACT SECURITY. ORANGE CELLS INDICATE WHERE THE FORMER GROUP IS MORE NEGATIVE ABOUT THE STATEMENT THAN THE LATTER GROUP; BLUE CELLS INDICATE WHERE THE FORMER GROUP IS MORE POSITIVE. GREEN CELLS REPRESENT STATISTICALLY SIGNIFICANT DIFFERENCES. THE NUMBER IN "()" INDICATES THE SIZE OF EACH GROUP.

Statement		Effect Size: Public (80) vs. Consortium (49)	Effect Size: Public (80) vs. Private (20)	Effect Size: Consortium (49) vs. Private (20)	P-value: Public (80) vs. Consortium (49)	P-value: Public (80) vs. Private (20)	P-value: Consortium (49) vs. Private (20)
Motivators:							
Software security is in my company's culture.	[M1]	0.16	0.09	-0.09	1.000	1.000	1.000
My company mandates security practices in smart contract development.	[M2]	0.16	-0.02	-0.20	1.000	1.000	1.000
My company is audited for smart contract security by an external entity.	[M3]	0.20	0.14	-0.06	0.740	1.000	1.000
My company would lose customers in case of a security breach.	[M4]	0.18	0.09	-0.11	0.778	1.000	1.000
Security breaches would hurt my company's reputation.	[M5]	0.07	0.10	0.02	1.000	1.000	1.000
Software security is a shared responsibility by all those involved in the development lifecycle.	[M6]	-0.09	-0.05	0.05	1.000	1.000	1.000
I see software security as my responsibility.	[M7]	0.18	0.23	0.03	0.739	0.890	1.000
I care about my users' experience in security and privacy.	[M8]	0.16	0.19	0.01	0.972	1.000	1.000
Customers would lose money in case of a security breach.	[M9]	0.02	-0.01	-0.04	1.000	1.000	1.000
The deployed smart contracts are immutable.	[M10]	0.39	0.14	-0.24	0.002	1.000	1.000
It is challenging to detect and trace attacks on smart contracts deployed to blockchains.	[M11]	-0.13	-0.10	0.01	1.000	1.000	1.000
Deterrents:							
The smart contracts I develop are not prone to security attacks.	[D1]	-0.19	-0.09	0.12	0.674	1.000	1.000
Smart contract security is not my responsibility.	[D2]	-0.36	-0.28	0.08	0.003	0.345	1.000
Smart contract security is a burden on top of my main responsibilities.	[D3]	0.06	-0.22	-0.28	1.000	1.000	0.608
In my team, it is more important to deliver features on time than to address smart contract security.	[D4]	-0.43	-0.54	-0.12	0.000	0.001	1.000
My team does not have the budget to address smart contract security.	[D5]	-0.23	-0.35	-0.21	0.268	0.130	1.000
My company does not have a formal process for smart contract security.	[D6]	-0.17	-0.33	-0.23	0.999	0.244	1.000
It is unlikely that our smart contract will be attacked.	[D7]	0.03	-0.12	-0.16	1.000	1.000	1.000
It is easy to mitigate the negative impact of a security breach.	[D8]	-0.06	-0.24	-0.21	1.000	0.922	1.000
Available security tools for smart contracts are not useful.	[D9]	-0.02	-0.34	-0.37	1.000	0.280	0.223
Security Efforts at Stages:							
Requirement	[E1]	0.23	0.01	-0.22	0.174	1.000	0.945
Design	[E2]	0.27	-0.06	-0.34	0.059	1.000	0.186
Construction	[E3]	0.28	0.00	-0.25	0.042	1.000	0.675
Testing	[E4]	0.21	0.22	0.09	0.255	0.806	1.000
Deployment	[E5]	0.21	-0.08	-0.30	0.251	1.000	0.394
Maintenance	[E6]	0.27	0.08	-0.15	0.057	1.000	1.000
Security Strategies:							
Threat Assessment	[S1]	0.06	0.34	0.32	1.000	0.265	0.435
Code Style Checking	[S2]	0.14	0.17	0.03	1.000	1.000	1.000
Code Review	[S3]	0.40	0.38	0.09	0.001	0.079	1.000
Static Analysis	[S4]	0.04	0.26	0.28	1.000	0.886	0.867
Formal Verification	[S5]	-0.03	0.13	0.19	1.000	1.000	1.000
Vulnerability Scan	[S6]	-0.02	0.17	0.26	1.000	1.000	1.000
Fuzzing	[S7]	-0.14	0.10	0.24	1.000	1.000	1.000
Security Testing	[S8]	0.11	0.24	0.18	1.000	1.000	1.000
External Auditing	[S9]	0.17	0.19	0.04	1.000	1.000	1.000
Runtime Monitoring	[S10]	0.06	0.13	0.10	1.000	1.000	1.000
Code Reuse	[S11]	0.28	0.22	-0.04	0.079	1.000	1.000

between two sample distributions, ranging from -1 to $+1$. The extreme value ± 1 occurs when the intersection between both groups is an empty set. When the compared groups tend to overlap, Cliff's Delta approaches zero. The magnitudes can be assessed with the thresholds as specified in [57]: if $|\delta| < 0.147$, the effect size is negligible; if $0.147 \leqslant |\delta| < 0.33$, the effect size is small; if $0.330 \leqslant |\delta| < 0.474$, the effect size is medium; and otherwise the effect size is large. Effect sizes are additionally colored on a gradient from blue to orange based on the magnitudes of difference: blue color means the former group is more positive about the statement, and orange color means the latter group is more positive about the statement.

The *P-value* column indicates whether the differences for each statement are statistically significant between *Public Blockchain* and *Consortium Blockchain* in the first subcolumn, *Public Blockchain* and *Private Blockchain* in the second subcolumn, and *Consortium Blockchain* and *Private Blockchain* in the third subcolumn. Statistically significant differences at a 95% confidence level (Bonferroni corrected p-value < 0.05) are highlighted in green.

Based on the observed statistically significant differences and effect sizes, we can say with some certainty that:

- **Security Motivators:** The blockchain platforms significantly impact the security motivator in terms of the immutability of smart contracts. The immutability of smart contracts drives practitioners of public blockchains more intensively than practitioners of consortium blockchains. In addition, the practitioners of public blockchains tend to be more motivated to address smart contract security than those of consortium and private blockchains.
- **Security Deterrents:** The blockchain platforms significantly affect the deterrents to security with respect to competing priorities in smart contract development. Practitioners of public blockchains tend to be more willing to prioritize security tasks over feature delivery and take the responsibility of addressing smart contract security, in comparison with those of consortium and private blockchains.
- **Security Efforts across Stages:** The blockchain platforms statistically significantly impact the security efforts at the construction stage in the development lifecycle. Practitioners of public blockchains spend more efforts towards security throughout the six stages in the development lifecycle, especially at the construction stage, in comparison with practitioners of consortium blockchains. In addition, practitioners of consortium blockchains tend to put less emphasis on security at the requirement, construction, deployment, and maintenance stages, in comparison with practitioners of private blockchains.
- **Security Strategies:** The blockchain platforms significantly affect the code review strategy that practitioners use to address smart contract security. Practitioners of public blockchains tend to perform code review more frequently than practitioners of consortium and private blockchains. Aside from code review, we observed no significant difference in the frequency of use of other secu-

rity strategies between public and consortium blockchain practitioners. Private blockchain practitioners tend to use security strategies less frequently than public and consortium blockchain practitioners.

V. Discussion

We reflect on our findings of research questions, delving into security awareness and risks of smart contracts, as well as code reuse and tool implications in smart contracts. We also highlight the avenues of future research across blockchain platforms.

A. Security Awareness and Risks of Smart Contracts

The vast majority of our respondents acknowledge the importance of smart contract security (RQ1). They prioritize security over the reduction of execution cost (e.g., gas consumption) in smart contract development. Our respondents spend 29% of overall efforts on average in conducting security-related tasks. Previous studies found that developers generally exhibit a "security is not my responsibility" attitude [73]. On the contrary, 85% of our respondents see smart contract security as their responsibility. **Smart contract practitioners tend to have a higher awareness of security than practitioners in other software areas.**

Despite the high awareness of security among smart contract practitioners, 40% of our survey respondents indicated that their smart contracts suffered from security problems, including security breaches (RQ1). The percentage is higher than that of software in general (33%) as reported in a recent survey [7]. The frequent occurrence of security problems in smart contracts may stem from the optimism bias [55] ([D1]) and lack of a formal security process ([D6]) as suggested in RQ1. In addition, smart contracts on public blockchains are visible and accessible to all users, even malicious attackers. The inherent features of smart contracts make them more prone to security attacks than traditional software. Future work may focus on **standardizing and operationalizing the process of building security in smart contracts.**

B. Code Reuse and Tool Implications in Smart Contracts

58% of the respondents frequently reuse code from reliable sources in smart contract development (RQ2). For instance, OpenZeppelin proposes the *SafeMath* libraries [46] to help developers of Ethereum smart contracts perform proper validation on numeric inputs and prevent integer overflow and underflow vulnerabilities. Only two respondents in our survey mentioned that they use dependency management as a security strategy. Previous studies found that improper use of libraries, including security-related APIs, can introduce security vulnerabilities [19], [21], [24], [39]. Future studies could put more effort into providing **documentations of smart contract libraries with helpful examples** and **tools to facilitate library updates** for smart contract development.

To facilitate code reuse, Ethereum Virtual Machine provides an opcode, `DELEGATECALL` [34], for dynamically loading the bytecode of a callee contract into the caller contract at runtime. A DoS attack against the Parity wallet leveraged the vulnerability due to improper use of `DELEGATECALL`. Thus, code reuse in smart contracts can impose a higher risk than its counterpart in traditional software, highlighting the importance of **security auditing on broadly used smart contracts and libraries.**

Active maintenance ranks on top of the factors that affect the adoption of security tools for smart contracts (RQ2). Among the four tools we investigated, the most frequently used tool, *Mythril* [14], has released 102 versions since its first release on October 4, 2017 – an average of 3 releases per month. The active maintenance would enable security tools to uncover the latest emerging security issues. As suggested in our survey, the practitioners tend to rely on security plugins in IDEs and prefer tools that cover a broad range of security issues. Thus, the future work could focus on **automatically incorporating emerging security issues into security tools** and **integrating various security tools into the IDEs.** In addition, the strategy of chasing behind the attackers is not adequate to address smart contract security. Practitioners could **proactively defense smart contracts against security attacks via external auditing and fuzzing.**

C. Studies across Blockchain Platforms

The results of RQ3 indicates that blockchain platforms impact practitioners' perceptions and practices on smart contract security. Nonetheless, previous studies usually focus on one blockchain platform. It could be interesting to investigate **the difference in security issues across different blockchains**, and **whether and how existing tools can be used across different blockchains**. In addition, practitioners of public blockchains tend to be more motivated to address smart contract security than those of consortium and private blockchains. The potential reason could be the accessibility of public blockchains to any users, even malicious attackers. Future research could investigate **whether the practitioners of consortium and private blockchains make an economic decision of security strategies based on risk assessment**.

VI. Threats to Validity

Internal Validity. In our study, the interviewees were selected by a contact at each company or open-source project who identified the practitioners to be interviewed. The procedure partially alleviates the threat of selection bias since the interviewer has no contact with interviewees before the interviews. The threat of selection bias would always be present when the interviewees were not fully randomly sampled. However, given that our interviews include practitioners with various job roles and from different companies and open-source projects, the threat has limited effect.

As for the survey, it is possible that some of our respondents had a poor understanding of the statements for rating. Their responses may introduce noise to the data that we collected. To reduce the impact of this issue, we included an "I don't know" option in the survey and ignored responses marked as such. We also dropped responses that were submitted by people whose

job roles are none of these: software development, testing, and project management. Two of the authors translated our survey to Chinese to ensure that respondents from China could understand our survey well. To reduce the bias of presenting the survey bilingually, we carefully translated our survey to ensure there is no ambiguity between English and Chinese terms. We polished the translation by improving clarity and understandability according to the feedback from our pilot survey.

Construct Validity. In our interviews, the evaluation apprehension was ameliorated by the anonymity of the interviewees, as well as the guaranty that all the information obtained during the interviews would be used only by the researchers. The interviewer might have influenced the interviewees by asking specific questions. To mitigate this risk, we used open-ended questions to elicit as much information as possible from practitioners. The interviewees may have had a different understanding of the questions than what we had intended. To minimize this aspect, we encouraged the interviewees to ask questions at all times.

In our survey, the results are based on respondents' self-reported responses, which may be subject to bias and not exactly represent reality. We followed recommendations to reduce social-desirability bias by ensuring respondents' anonymity [42]. The questionnaire in our survey is based on interview results instead of validated scales. Although we use factor analysis to analyze the results, it may be insufficient to validate the scales.

Conclusion Validity. The interviews were conducted at different locations and each interview was done in one work session. Thus, answers were not influenced by internal discussions. To ensure that the interview instrument is of high quality to obtain reliable measures, we conducted several pilots to improve the questions and layout of the interview guide prior to conducting the interviews.

In addition, we did our best to randomly select survey respondents from both companies and open-source projects. Our survey respondents come from 35 countries across six continents who are work in various job roles with a wide range of experience.

External Validity. To improve the generalizability of our findings regarding smart contract development, we interviewed 13 interviewees from blockchain companies and open-source blockchain projects. We further surveyed 156 respondents from 35 countries across six continents who are working for various companies or contributing to open-source blockchain/smart contract projects that are hosted on GitHub, in various job roles.

VII. Conclusion

This work proposed a mixed qualitative and quantitative approach to explore practitioners' perceptions and practices on smart contract security. We recognized the disconnect between the security awareness of smart contract practitioners and the occurrence of security problems in smart contracts. We also provided practical lessons about code reuse, tool implications, and proactive defense to ensure smart contract security. Besides, we observed several differences between smart contract security and regular security: (1) Smart contract practitioners tend to have a higher security awareness than regular practitioners; (2) Smart contracts are more prone to security attacks than regular software; (3) More frequent code reuse in smart contract development imposes higher security risk than regular software development. Future studies could put more effort into investigating the differences in various aspects of smart contracts on top of different blockchain platforms, and generalize existing tools across different blockchain platforms.

Acknowledgements

This research was partially supported by the National Key R&D Program of China (No. 2020YFB1005400), Australian Research Council's Discovery Early Career Researcher Award (DECRA) Funding Scheme (DE200100021), ARC Discovery Grant (DP200100020), National Science Foundation of China (No. U20A20173), Hong Kong RGC Project (No. 152193/19E), and the National Research Foundation, Singapore under its Industry Alignment Fund - Prepositioning (IAF-PP) Funding Initiative. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore.

References

[1] Nvivo qualitative data analysis software, 2021.

[2] Y. Acar, M. Backes, S. Fahl, S. Garfinkel, D. Kim, M. L. Mazurek, and C. Stransky. Comparing the usability of cryptographic apis. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P '17)*, pages 154–171. IEEE, 2017.

[3] Y. Acar, M. Backes, S. Fahl, D. Kim, M. L. Mazurek, and C. Stransky. You get where you're looking for: The impact of information sources on code security. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P '16)*, pages 289–305. IEEE, 2016.

[4] M. Alharby and A. Van Moorsel. Blockchain-based smart contracts: A systematic mapping study. *arXiv preprint arXiv:1710.06372*, 2017.

[5] S. Amani, M. Bégel, M. Bortin, and M. Staples. Towards verifying ethereum smart contract bytecode in isabelle/hol. In *Proceedings of the 7th ACM SIGPLAN International Conference on Certified Programs and Proofs*, pages 66–77, 2018.

[6] H. Assal and S. Chiasson. Security in the software development lifecycle. In *Proceedings of the 14th Symposium on Usable Privacy and Security (SOUPS '18)*, pages 281–296, 2018.

[7] H. Assal and S. Chiasson. 'think secure from the beginning' a survey with software developers. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–13, 2019.

[8] J. Chang, B. Gao, H. Xiao, J. Sun, Y. Cai, and Z. Yang. scompile: Critical path identification and analysis for smart contracts. In *Proceedings of the International Conference on Formal Engineering Methods*, pages 286–304. Springer, 2019.

[9] H. Chen, M. Pendleton, L. Njilla, and S. Xu. A survey on ethereum systems security: Vulnerabilities, attacks, and defenses. *ACM Computing Surveys (CSUR)*, 53(3):1–43, 2020.

[10] M. Christakis and C. Bird. What developers want and need from program analysis: an empirical study. In *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering (ASE '16)*, pages 332–343, 2016.

[11] M. Coblenz. Obsidian: a safer blockchain programming language. In *Proceedings of the IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-Companion '17)*, pages 97–99. IEEE, 2017.

[12] J. Cohen. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46, 1960.

[13] ConsenSys. Ethereum smart contract security best practices. https://consensys.github.io/smart-contract-best-practices, 2018. Online; accessed February 2021.

[14] ConsenSys. Mythril. https://github.com/ConsenSys/mythril, 2020. Online; accessed February 2021.

[15] T. Cook, A. Latham, and J. H. Lee. Dappguard: Active monitoring and defense for solidity smart contracts. https://courses.csail.mit.edu/6.857/2017/project/23.pdf, 2020. Online; accessed February 2021.

[16] Cornell Blockchain. Bamboo: a language for morphing smart contracts. https://github.com/CornellBlockchain/bamboo, 2020. Online; accessed February 2021.

[17] J. W. Creswell and J. D. Creswell. *Research design: Qualitative, quantitative, and mixed methods approaches*. Sage publications, 2017.

[18] D. S. Cruzes and T. Dyba. Recommended steps for thematic synthesis in software engineering. In *Proceedings of the International Symposium on Empirical Software Engineering and Measurement*, pages 275–284. IEEE, 2011.

[19] M. Egele, D. Brumley, Y. Fratantonio, and C. Kruegel. An empirical study of cryptographic misuse in android applications. In *Proceedings of the ACM SIGSAC conference on Computer and communications security (CCS '13)*, pages 73–84, 2013.

[20] Ethereum Foundation. Vyper documentation. https://vyper.readthedocs.io/en/latest/?badge=latest, 2020. Online; accessed February 2021.

[21] S. Fahl, M. Harbach, T. Muders, L. Baumgärtner, B. Freisleben, and M. Smith. Why eve and mallory love android: An analysis of android ssl (in) security. In *Proceedings of the ACM conference on Computer and communications security (CCS '12)*, pages 50–61, 2012.

[22] S. Fahl, M. Harbach, H. Perl, M. Koetter, and M. Smith. Rethinking ssl development in an appified world. In *Proceedings of the ACM SIGSAC conference on Computer and communications security (CCS '13)*, pages 49–60, 2013.

[23] J. Feist, G. Grieco, and A. Groce. Slither: a static analysis framework for smart contracts. In *Proceedings of the IEEE/ACM 2nd International Workshop on Emerging Trends in Software Engineering for Blockchain*, pages 8–15. IEEE, 2019.

[24] M. Georgiev, S. Iyengar, S. Jana, R. Anubhai, D. Boneh, and V. Shmatikov. The most dangerous code in the world: validating ssl certificates in non-browser software. In *Proceedings of the ACM conference on Computer and communications security (CCS '12)*, pages 38–49, 2012.

[25] N. Grech, M. Kong, A. Jurisevic, L. Brent, B. Scholz, and Y. Smaragdakis. Madmax: Surviving out-of-gas conditions in ethereum smart contracts. *Proceedings of the ACM on Programming Languages*, 2(OOPSLA):1–27, 2018.

[26] I. Grishchenko, M. Maffei, and C. Schneidewind. Foundations and tools for the static analysis of ethereum smart contracts. In *Proceedings of the International Conference on Computer Aided Verification*, pages 51–78. Springer, 2018.

[27] I. Grishchenko, M. Maffei, and C. Schneidewind. A semantic framework for the security analysis of ethereum smart contracts. In *Proceedings of the International Conference on Principles of Security and Trust*, pages 243–269. Springer, 2018.

[28] E. Hildenbrandt, M. Saxena, N. Rodrigues, X. Zhu, P. Daian, D. Guth, B. Moore, D. Park, Y. Zhang, A. Stefanescu, et al. Kevm: A complete formal semantics of the ethereum virtual machine. In *Proceedings of the IEEE 31st Computer Security Foundations Symposium*, pages 204–217. IEEE, 2018.

[29] Y. Hirai. Defining the ethereum virtual machine for interactive theorem provers. In *Proceedings of the International Conference on Financial Cryptography and Data Security*, pages 520–535. Springer, 2017.

[30] B. Jiang, Y. Liu, and W. Chan. Contractfuzzer: Fuzzing smart contracts for vulnerability detection. In *Proceedings of the 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE '18)*, pages 259–269. IEEE, 2018.

[31] B. Johnson, Y. Song, E. Murphy-Hill, and R. Bowdidge. Why don't software developers use static analysis tools to find bugs? In *Proceedings of the 35th International Conference on Software Engineering (ICSE '13)*, pages 672–681. IEEE, 2013.

[32] S. Kalra, S. Goel, M. Dhawan, and S. Sharma. Zeus: Analyzing safety of smart contracts. In *Proceedings of the Network and Distributed Systems Security Symposium (NDSS '18)*, 2018.

[33] B. A. Kitchenham and S. L. Pfleeger. Personal opinion surveys. In *Guide to advanced empirical software engineering*, pages 63–92. Springer, 2008.

[34] J. Krupp and C. Rossow. teether: Gnawing at ethereum to automatically exploit smart contracts. In *Proceedings of the 27th USENIX Security Symposium (USENIX Security '18)*, pages 1317–1333, 2018.

[35] T. D. LaToza, G. Venolia, and R. DeLine. Maintaining mental models: A study of developer work habits. In *Proceedings of the 28th International Conference on Software Engineering (ICSE '06)*, pages 492–501, New York, NY, USA, 2006. ACM.

[36] J. Lazar, J. H. Feng, and H. Hochheiser. *Research Methods in Human-Computer Interaction*. John Wiley & Sons, 2010.

[37] C. Liu, H. Liu, Z. Cao, Z. Chen, B. Chen, and B. Roscoe. Reguard: finding reentrancy bugs in smart contracts. In *Proceedings of the IEEE/ACM 40th International Conference on Software Engineering: Companion (ICSE-Companion '18)*, pages 65–68. IEEE, 2018.

[38] L. Luu, D.-H. Chu, H. Olickel, P. Saxena, and A. Hobor. Making smart contracts smarter. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS '16)*, pages 254–269, 2016.

[39] S. Nadi, S. Krüger, M. Mezini, and E. Bodden. Jumping through hoops: Why do java developers struggle with cryptography apis? In *Proceedings of the 38th International Conference on Software Engineering (ICSE '16)*, pages 935–946, 2016.

[40] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Bitcoin White Paper*, 2008.

[41] National Vulnerability Database. Cve-2018-10299. https://nvd.nist.gov/vuln/detail/CVE-2018-10299, 2018. Online; accessed February 2021.

[42] A. J. Nederhof. Methods of coping with social desirability bias: A review. *European journal of social psychology*, 15(3):263–280, 1985.

[43] I. Nikolić, A. Kolluri, I. Sergey, P. Saxena, and A. Hobor. Finding the greedy, prodigal, and suicidal contracts at scale. In *Proceedings of the 34th Annual Computer Security Applications Conference*, pages 653–663, 2018.

[44] D. S. Oliveira, T. Lin, M. S. Rahman, R. Akefirad, D. Ellis, E. Perez, R. Bobhate, L. A. DeLong, J. Cappos, and Y. Brun. Api blindspots: Why experienced developers write vulnerable code. In *Proceedings of the 14th Symposium on Usable Privacy and Security (SOUPS '18)*, pages 315–328, 2018.

[45] M. Oltrogge, Y. Acar, S. Dechand, M. Smith, and S. Fahl. To pin or not to pin—helping app developers bullet proof their tls connections. In *Proceedings of the 24th USENIX Security Symposium (USENIX Security '15)*, pages 239–254, 2015.

[46] OpenZeppelin. Safemath. https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/math/SafeMath.sol, 2018. Online; accessed February 2021.

[47] Parity Technologies. The multi-sig hack: A postmortem. https://www.parity.io/the-multi-sig-hack-a-postmortem/, 2008. Online; accessed February 2021.

[48] Parity Technologies. Security alert: Parity wallet (multi-sig wallets). https://www.parity.io/security-alert-2/, 2008. Online; accessed February 2021.

[49] D. Park, Y. Zhang, M. Saxena, P. Daian, and G. Roşu. A formal verification tool for ethereum vm bytecode. In *Proceedings of the 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '18)*, pages 912–915, 2018.

[50] A. Permenev, D. Dimitrov, P. Tsankov, D. Drachsler-Cohen, and M. Vechev. Verx: Safety verification of smart contracts. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P '20)*, pages 18–20, 2020.

[51] A. Poller, L. Kocksch, S. Türpe, F. A. Epp, and K. Kinder-Kurlanda. Can security become a routine? a study of organizational change in an agile software development group. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work and Social Computing (CSCW '17)*, pages 2489–2503, 2017.

[52] L. Quan, L. Wu, and H. Wang. Evulhunter: Detecting fake transfer vulnerabilities for eosio's smart contracts at webassembly-level. *arXiv preprint arXiv:1906.10362*, 2019.

[53] Qureshi, Haseeb. A hacker stole $ 31m of ether–how it happened, and what it means for ethereum. https://www.freecodecamp.org/news/a-hacker-stole-31m-of-ether-how-it-happened-and-what-it-means-for-ethereum-9e5dc29e33ce/, 2017. Online; accessed February 2021.

[54] W. Revelle. psych: Procedures for personality and psychological research, 2017.

[55] H.-S. Rhee, Y. U. Ryu, and C.-T. Kim. Unrealistic optimism on information security management. *Computers & Security*, 31(2):221–232, 2012.

[56] M. Rodler, W. Li, G. O. Karame, and L. Davi. Sereum: Protecting existing smart contracts against re-entrancy attacks. In *Proceedings of the Network and Distributed Systems Security Symposium (NDSS '19)*, 2019.

[57] J. Romano, J. D. Kromrey, J. Coraggio, and J. Skowronek. Appropriate statistics for ordinal level data: Should we really be using t-test and cohen'sd for evaluating group differences on the nsse and other surveys. In *Proceedings of the Annual Meeting of the Florida Association of Institutional Research*, pages 1–33, 2006.

[58] R. M. Ryan and E. L. Deci. Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being. *American psychologist*, 55(1):68, 2000.

[59] M. Suiche. Porosity: A decompiler for blockchain-based smart contracts bytecode. *DEF CON*, 25:11, 2017.

[60] T. W. Thomas, H. Lipford, B. Chu, J. Smith, and E. Murphy-Hill. What questions remain? an examination of how developers understand an interactive static analysis tool. In *Proceedings of the 12th Symposium on Usable Privacy and Security (SOUPS '16)*, 2016.

[61] T. W. Thomas, M. Tabassum, B. Chu, and H. Lipford. Security during application development: An application security expert perspective. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–12, 2018.

[62] B. Thompson. Exploratory and confirmatory factor analysis: Understanding concepts and applications. *Applied Psychological Measurement*, 31(3):245–248, 2007.

[63] S. Tikhomirov, E. Voskresenskaya, I. Ivanitskiy, R. Takhaviev, E. Marchenko, and Y. Alexandrov. Smartcheck: Static analysis of ethereum smart contracts. In *Proceedings of the 1st International Workshop on Emerging Trends in Software Engineering for Blockchain*, pages 9–16, 2018.

[64] P. Tsankov, A. Dan, D. Drachsler-Cohen, A. Gervais, F. Buenzli, and M. Vechev. Securify: Practical security analysis of smart contracts. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS '18)*, pages 67–82, 2018.

[65] S. Türpe, L. Kocksch, and A. Poller. Penetration tests a turning point in security practices? organizational challenges and implications in a software development team. In *Proceedings of the 12th Symposium on Usable Privacy and Security (SOUPS '16)*, 2016.

[66] P. K. Tyagi. The effects of appeals, anonymity, and feedback on mail survey response patterns from salespeople. *Journal of the Academy of Marketing Science*, 17(3):235–241, Jun 1989.

[67] Z. Wan, X. Xia, and A. E. Hassan. What do programmers discuss about blockchain? a case study on the use of balanced lda and the reference architecture of a domain to capture online discussions about blockchain platforms across the stack exchange communities. *IEEE Transactions on Software Engineering*, 2019.

[68] Z. Wan, X. Xia, A. E. Hassan, D. Lo, J. Yin, and X. Yang. Perceptions, expectations, and challenges in defect prediction. *IEEE Transactions on Software Engineering*, 46(11):1241–1266, 2020.

[69] J. Wang, B. Chen, L. Wei, and Y. Liu. Superion: Grammar-aware greybox fuzzing. In *Proceedings of the IEEE/ACM 41st International Conference on Software Engineering (ICSE '19)*, pages 724–735. IEEE, 2019.

[70] C. Weir, A. Rashid, and J. Noble. How to improve the security skills of mobile app developers? comparing and contrasting expert views. In *Proceedings of the 12th Symposium on Usable Privacy and Security (SOUPS '16)*, 2016.

[71] J. Witschey, O. Zielinska, A. Welk, E. Murphy-Hill, C. Mayhorn, and T. Zimmermann. Quantifying developers' adoption of security tools. In *Proceedings of the 10th Joint Meeting on Foundations of Software Engineering (FSE '15)*, pages 260–271, 2015.

[72] S. Xiao, J. Witschey, and E. Murphy-Hill. Social influences on secure development tool adoption: why security tools spread. In *Proceedings of the 17th ACM conference on Computer Supported Cooperative Work and Social Computing (CSCW '14)*, pages 1095–1106, 2014.

[73] S. Xiao, J. Witschey, and E. Murphy-Hill. Social influences on secure development tool adoption: why security tools spread. In *Proceedings of the 17th ACM conference on Computer Supported Cooperative Work & Social Computing*, pages 1095–1106, 2014.

[74] J. Xie, H. R. Lipford, and B. Chu. Why do programmers make security errors? In *Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC '11)*, pages 161–164. IEEE, 2011.

[75] K. Yamashita, Y. Nomura, E. Zhou, B. Pi, and S. Jun. Potential risks of hyperledger fabric smart contracts. In *Proceedings of the IEEE International Workshop on Blockchain Oriented Software Engineering (IWBOSE '19)*, pages 1–10. IEEE, 2019.

[76] Y. Zhou, D. Kumar, S. Bakshi, J. Mason, A. Miller, and M. Bailey. Erays: reverse engineering ethereum's opaque smart contracts. In *Proceedings of the 27th USENIX Security Symposium (USENIX Security '18)*, pages 1371–1385, 2018.