ABC: A Cryptocurrency-Focused Threat Modeling Framework

Ghada Almashaqbeh
Columbia University
ghada@cs.columbia.edu

Allison Bishop
Proof Trading and Columbia University
allison@cs.columbia.edu

Justin Cappos
New York University
jcappos@nyu.edu

***Abstract*—Cryptocurrencies are an emerging economic force, but there are concerns about their security. This is due, in part, to complex collusion cases and new threat vectors, that could be missed by conventional security assessment strategies. To address these issues, we propose ABC, an Asset-Based Cryptocurrency-focused threat modeling framework capable of identifying such risks. ABC's key innovation is the use of collusion matrices. A collusion matrix forces a threat model to cover a large space of threat cases while simultaneously manages this process to prevent it from being overly complex. We demonstrate that ABC is effective by presenting real-world use cases and by conducting a user study. The user study showed that around 71% of those who used ABC were able to identify financial security threats, as compared to only 13% of participants who used the popular framework STRIDE.**

I. Introduction

Cryptocurrencies and blockchain technologies are an emerging economic force. Since Bitcoin emerged in 2009 [1], the number of these "digital currencies" has grown into the thousands in 2018, with a total market capital exceeding $130 billion [2]. Although early systems focused only on providing a virtual currency exchange medium [1], [3], nowadays there is increasing interest in providing other types of distributed services, such as computation outsourcing [4] and file storage [5]. These newer applications suggest increased adoption of cryptocurrency-based systems in the coming years.

Yet, despite the many advantages they offer — decentralization, transparency, and lowered service costs — there is still a big gap between the promise of cryptocurrencies and their performance in practice. A major stumbling block is the perception that these systems are not secure, and the large number of security breaches announced in the past few years give credence to these doubts [6]–[8]. Therefore, a better understanding of the security of cryptocurrencies is needed in order to ensure their safe deployment in emerging applications.

The best practice for designing a secure system requires a threat modeling step to investigate potential security risks. Such a model can guide system designers in deploying the proper countermeasures, and assessing the security level of a system. Although threat modeling has been thoroughly studied in the literature, existing paradigms primarily target software applications [9] or distributed systems that have a small number of participant types [10]. These threat modeling techniques were frankly not designed to be scalable for a set of diverse, mutually distrustful parties as found in cryptocurrencies. Such systems, especially these providing distributed services, consist of parties that play different roles (miners, clients, and different types of servers), and an attacker may control any subset of these parties. Adding to the complexity, the attacker may seek to target any role in the system and may launch a diverse array of attacks with different intended outcomes. The complexity of reasoning about and managing threat cases becomes unwieldy as these sets grow.

To address this issue, we propose ABC, an **A**sset-**B**ased **C**ryptocurrency-focused threat modeling framework. ABC introduces a novel technique, called a collusion matrix, that allows users to investigate the full threat space and manage its complexity. A collusion matrix is a comprehensive investigation and threat enumeration tool that directly addresses collusion by accounting for all possible sets of attacker and target parties. ABC helps in reducing the combinatorial growth of these cases by ruling out irrelevant ones and merging threat cases that have the same effect. Such explicit consideration of attacker collusion is particularly important in permissionless cryptocurrencies that allow anyone to join.

ABC's models are also tailored to better consider the threat domain of cryptocurrencies. This is done by introducing new threat categories that account for the financial motivations of attackers and the new asset types, i.e., critical system components, these systems introduce. ABC identifies these categories by listing the assets in a system, such as the blockchain, and outlining the secure behavior of each asset. Then, the threat categories are defined as any violation of the security requirements for these assets. This approach produces a series of system-specific threat classes as opposed to an a priori-fixed list of generalized ones. Another feature of ABC is acknowledging that financial incentives and economic analysis can play major roles in other steps in the design process. These tools can be used in risk assessment and in mitigating some types of attacks that cannot be neutralized cryptographically.

To demonstrate the framework's effectiveness, we conducted a user study and prepared use cases. The study compared the performance of subjects in building threat models using ABC and the popular framework STRIDE. Among the obtained results, we found that around 71% of those who applied ABC were able to identify financial threats in a cryptocurrency system, as compared to less than 13% of those applying STRIDE. In addition, while none of the STRIDE session participants spotted collusion between attackers, around 46% of those who used ABC identified these scenarios. For

978-1-7281-1878-9/19/$31.00 ©2019 IEEE

the use cases, we applied ABC to three real-world systems, including Bitcoin, Filecoin, and CacheCash. These cases attest to the usefulness of ABC's tools where they integrated well into CacheCash's design phase, and they revealed several missing threat scenarios in the public design of Filecoin. These findings confirm the potential of ABC as an effective tool for assessing and improving the security of cryptocurrencies.

II. Related Work

To orient readers to the current state-of-the-art in threat modeling, we summarize here some prior work done in this area. We also highlight relevant works on threat identification and security analysis in cryptocurrencies.

Threat modeling frameworks. The STRIDE framework is one of the most popular strategies in this field [9]. STRIDE is an acronym of the threat categories it covers, namely, Spoofing, Tampering, Repudiation, Information disclosure, Denial of service (DoS), and Elevation of privilege. This framework is a multistep procedure that involves understanding the software application functionality, capturing its operation flow using data flow diagrams (DFDs), mapping the components of these DFDs to the threat categories mentioned previously, and employing threat tree patterns to derive concrete threat cases. Though several solutions have extended STRIDE to accommodate more complex systems [10], and cover other security requirements, e.g., privacy [11], its model does not fit cryptocurrencies. Another study [12] has borne out this premise, where the authors extended STRIDE's threat categories to handle Bitcoin-like community currencies. However, their approach targets only community fund operation, where more modifications would be needed to handle other components of the currency exchange medium.

Other paradigms have pursued slightly different approaches. KAOS [13] is a goal-oriented requirements engineering framework that has been extended to cover security. ANOA [14] is a generic framework to define and analyze anonymity of communication networks. And the framework presented in [15] targets the secure design of data routing protocols. Finally, other works build specialized threat models for specific classes of distributed systems, e.g., storage systems [16], rather than introducing a framework. These works indicate that different types of systems have different requirements when performing threat modeling. This reinforces the idea that emerging systems, such as cryptocurrencies, need specialized threat modeling tools.

Security analysis of cryptocurrencies. Most of the work done so far in this category can be divided into two classes. The first formalizes the security properties of consensus protocols and blockchains [17], while the second discusses specific security attacks on cryptocurrencies. For example, in a series of studies on Bitcoin, Bonneau et al. [18] present several security threats, Androulaki et al. [19] evaluate its anonymity property, and Gervais et al. [20] study how tampering with the network links affects participants' view of the blockchain. Other works dealing with different types of cryptocurrencies include Luu et al. [21] who focus on Ethereum smart contract security, and Sanchez et al. [22] who analyze the security of Ripple.

While these attack descriptions are very useful, they only outline specific threat scenarios for a given system. Our goal, however, is to develop a framework that allows reasoning about the full set of potential attacks facing a variety of cryptocurrency types.

III. Stepping through the ABC Framework

Having highlighted the need for a cryptocurrency-specific threat modeling framework, we now present the ABC model. As a systematized approach, applying ABC starts by understanding the functionality of the cryptocurrency system under design with a focus on its asset types and the financial motivations of the participants (Section III-A). This is followed by identifying the impactful threat categories and mapping them to the system assets (Section III-B). After that, ABC directs system designers to extract concrete attack scenarios using a new tool called a collusion matrix, which helps in exploring and analyzing the full threat space (Section III-C). Lastly, ABC acknowledges that financial incentives affect other design steps including risk assessment and threat mitigation (Section III-D).

To make the discussion easier to follow, we illustrate the ABC process by describing its application to the following simplified system, which was inspired by Golem [4]:

CompuCoin is a cryptocurrency that provides a distributed computation outsourcing service. Parties with excessive CPU power may join the system as servers to perform computations on demand for others. Clients submit computation jobs to servers, wait for the results and proofs of correctness, and then pay these servers cryptocurrency tokens. The mining process in CompuCoin is tied to the amount of service provided to the system. That is, the probability of selecting a server to mine the next block on the blockchain is proportional to the amount of computation it has performed during a specific period.

The full threat model of CompuCoin is available online [23]. Several excerpts from this model are embedded in the discussion of ABC steps that follows.

A. System Model Characterization

Understanding the system is an essential step in the threat modeling process. A misleading or incomplete system description can lead a designer to overlook serious threats and/or incorporate irrelevant ones. Therefore, an accurate system model must outline the use scenarios of the system, the assumptions on which it relies, and any dependencies on external services. In addition, the system model must be aware of all participant roles, and any possible motivation they might have to attack the system. For the latter, evaluators need to consider how the financial interests of these participants shape their behaviors.

Moreover, a system model must define the critical components that need to be protected from attackers. These components represent the assets that would compromise the whole system if attacked. To capture the features of the system, ABC identifies these assets based on its functionality. In detail, ABC divides the system into modules, and labels the

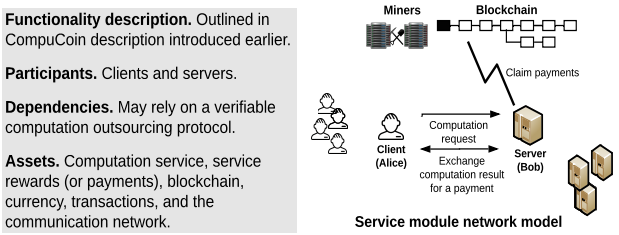

Fig. 1: System model characterization of CompuCoin.

valuable components of each module as assets. These assets could be concrete or abstract resources [10]. For example, the blockchain and the currency can be considered concrete assets, while preserving user privacy would be an abstract asset.

Finally, a system model includes graphic illustrations of its work flow. For distributed systems, it is useful to draw network models [10], in which system modules are represented by graphs showing all participants and assets, and the interactions between them. These graphs are helpful when enumerating the concrete threat scenarios as we will see in Section III-C.

Running example application. Figure 1 illustrates how this step would look in CompuCoin. It shows the various components of the system model, in addition to a network model of the computation outsourcing service. It should be noted that the asset list in this figure is not exhaustive, and is limited by the rather brief description provided for CompuCoin. Moreover, these assets could be divided differently. For instance, one may merge service payments and the currency together. However, we believe that a fine-grained division provides a more comprehensive threat treatment.

B. Threat Category Identification

After understanding the system model, the next step is to identify the broad threat categories that must be investigated. For each system component, system analysts outline all threat classes that may apply. Here ABC steps away from conventional practice of using an a priori-fixed list, and instead uses an adaptive approach inspired from requirements engineering [11] that defines threats as violations of system security goals. Given that assets are the target of security breaches, ABC defines these threat classes as violations of asset security requirements. This allows deriving cryptocurrency-specific threat categories because ABC identifies the assets in a way that aligns with the functionality of these systems.

Accordingly, ABC examines each asset in the system and applies the following procedure to identify its threat classes:

- Define what constitutes secure behavior for the asset, and use that knowledge to derive its security requirements. These requirements include all conditions that, if met, would render the asset secure. For example, CompuCoin's servers provide a computation outsourcing service and collect payments in return. We may consider the service payment asset secure if: a) servers are rewarded properly for their work, and b) that they earned the payments they collected.
- Define the threat categories of an asset as violations of its security requirements. Tying this to the above example, the service payment asset would have the following threat classes: service slacking, where a server collects payments without performing all the promised work, and service theft, where a client obtains service for a lower payment than the agreed upon amount.

TABLE I: CompuCoin threat categories.

Asset	Security Threat Category
Service	Service corruption (provide corrupted service for clients).
	Denial of service (make the service unavailable to legitimate users).
	Information disclosure (service content/related data are public).
	Repudiation (the server can deny a service it delivered).
Service payments	Service slacking (a server collects payments without performing all the promised work).
	Service theft (a client obtains correct service for a lower payment than the agreed upon amount).
Blockchain	Inconsistency (honest miners hold copies of the blockchain that may differ beyond the unconfirmed blocks).
	Invalid blocks adoption (the blockchain contains invalid blocks that does not follow the system specifications).
	Biased mining (a miner pretends to expend the needed resources for mining to be elected to extend the blockchain).
Transactions	Repudiation (an attacker denies issuing transactions).
	Tampering (an attacker manipulates the transactions in the system).
	Deanonymization (an attacker exploits transaction linkability and violates users' anonymity).
Currency	Currency theft (an attacker steals currency from others in the system).
Network	Denial of service (interrupt the operation of the underlying network).

The previous steps are highly dependent on how system analysts define the security properties of an asset, especially if there is no agreed-upon definition in the literature. For example, several works studied the security of the blockchain [17], [18]. Yet, there is no unified security notion for the service asset because each type may have different requirements.

Running example application. Applying this step to CompuCoin produced the threat categories listed in Table I (the detailed process can be found in the extended version of the paper [24]). We found this table useful when building threat models for all the use cases found in Section V, where we mapped the listed categories to the assets in each system. In this process, we found that some threat types were not applicable due to the absence of some assets. Notably, Bitcoin's only assets are the ones related to the currency exchange medium. On the other hand, some systems required replicating some of these categories among all instances of an asset, e.g., in Filecoin all service asset threats were replicated for the two service types this system provides, namely, file storage and retrieval. This shows how the system characteristics affect the threat category identification step in ABC.

C. Threat Scenario Enumeration and Reduction

Once the threat categories have been identified, the next step is to enumerate concrete attack scenarios under each threat type. It is important in this step to be as comprehensive as possible by considering all potential attackers, target parties, and the set of actions attackers may follow to achieve their goals. This also involves considering collusion between several attackers who may cooperate on attacking the system.

Detecting collusion is particularly important in cryptocurrencies. This is because the presence of monetary incentives may motivate attackers to collude in more ways than traditional distributed systems. The popular centralization problem

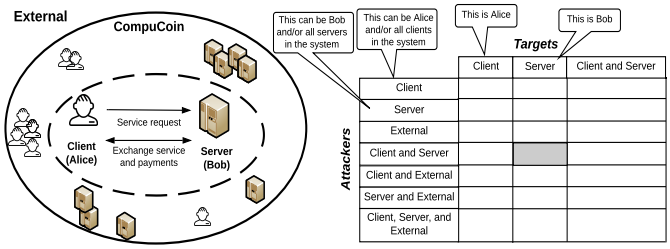

Fig. 2: Collusion matrix of service theft threat in CompuCoin.

caused by mining pools attests to this fact, where these pools can collude and perform devastating attacks. Even miners may collude by accepting, or rejecting, updates on the network protocol which leads to hard forks in the system. ABC can detect these, and other, collusion cases at early stages of the system design.

To achieve this, ABC introduces collusion matrices that instruct analysts to enumerate all collusion cases, and reason about the feasibility of all threats scenarios in the system. A collusion matrix is two-dimensional, with the rows representing potential attackers and the columns representing the target parties. For the rows we list all participant roles in the system, both individually and in every possible combination. We also add a category called "external" that represents all entities outside the system. The same is done for the columns, with the exception that "external" is excluded. By definition, an external party is not part of the system, and hence, can not be a target. Each cell in these matrices represents a potential threat case to be investigated.

An example of a collusion matrix for the service theft threat in CompuCoin is shown in Figure 2. The dashed ellipse in the accompanying network model encloses a service session, which is an interaction between a server and a client. Any entry with multiple parties on the attacker side in this matrix indicates collusion. Note that a participant label may represent slightly different roles depending on where it is placed. For example, in Figure 2 the label "server" on the target side corresponds to a single server (i.e., Bob) since a service session involves only one server. However, the label "server" on the attacker side represents all servers in the system, including Bob. Hence, the cell in grey shade in Figure 2 does not suggest that a server colludes to attack itself, but instead, it represents the case where other servers collude with Alice against Bob.

For each threat category mapped to the assets in the system, a separate collusion matrix is created and analyzed as follows:

1) Enumeration: In this step, system analysts examine each cell and enumerate all strategies that attackers can use against the target parties, and documenting the process. It is useful to consider the network model of the system components as they show the interactions between the participants and the system assets.

2) Reduction: While examining each cell, system analysts reduce the number of threat cases by:

- Eliminating cells representing scenarios that will not produce a threat to the system. This consists of crossing out the eliminated cells and documenting the rationale for elimination. For example, in Figure 2, the cells that have the client as a target are irrelevant to the service theft threat. This is because a client does not provide a service to others. Other cases can also be crossed out if they are neutralized by system assumptions or by early design choices. For example, requiring all transactions to be signed by their originators rules out transaction repudiation and tampering attacks.
- Merging together scenarios (and the corresponding cells) that have the same effect, or those that do not become stronger with collusion. For example, in Figure 2, the grey shaded cell in which Alice is colluding with other servers to avoid paying Bob is reduced to the case that Alice is a sole attacker. This is because only Alice pays for the service she receives from Bob, while other servers are not part of the protocol.

Attacker \ Target	*Client*	*Server*	*Client and Server*
External	**Clients cannot be targets because they do not serve others.**	**Servers and external cannot attack because they do not ask/pay for service.**	**Reduced to the case of attacking servers only, clients do not serve others (cannot be targets).**
Server			
Server and External			
Client		**(1)** Refuse to pay after receiving the service. **(2)** Issue invalid payments.	
Client and External		**Reduced to the case of an attacker client. A client does not become stronger when colluding with other servers or external entities.**	
Server and Client			
Client, Server, and External			

Fig. 3: Analyzing service theft matrix in CompuCoin.

3) Documentation: System analysts should document all threat scenarios that remain after the reduction step. That is, each documented case should outline the attack description, the target asset(s), the attacker(s), the flow of actions, all preconditions that make the attack feasible, and the reasons behind merges and deletions (if any).

The overall number of matrices and the size of each matrix depend on the system parameters, such as number of participant roles and assets. The above reduction steps eliminates a substantial number of cells in a documented, principled way, which saves time and effort.

Running example application. The CompuCoin threat model has 11 collusion matrices [23]. We present one of them here; the service theft threat collusion matrix as illustrated in Figure 3. As shown, 21 cells can be reduced to just 2 threat scenarios (merged and ruled-out cells are displayed in pink and black shades, respectively). In this matrix, ten cases have been ruled out. This includes all cells under the column with the "client" header, for the reasons explained previously, and the first three cells under the column with the "server" header. This is because "external" and/or "server" cannot be attackers because they do not ask/pay for the service[1].

Ten other merged cases are shown in Figure 3. This includes all cells under the column with the "client and server" header, which are reduced to attacking only servers. This is again

[1]One may say that an external may join the system as a client to perform the attack. This case is covered under the client role in the matrix.

because clients do not serve others. The rest of the merges cover the last three cells in the column with the "Server" header. In these cells, a client is colluding with an external entity and/or other servers to make the target server lose payments. Such collusion will not make a client stronger (these parties can drop/withhold payments, however, this is covered under DoS threat). Hence, all these cells are reduced to the case of a solo client attacker.

D. Risk Assessment and Threat Mitigation

The outcome of the threat modeling process, i.e., the documented list of impactful threat cases, can provide the designers with a guiding map to secure the system. During this process, it is useful to prioritize threats based on the amount of damage they can cause. This falls under the purview of risk management, a separate task from threat modeling, carried out using frameworks like DREAD [9].

ABC integrates with risk management by leveraging existing techniques for threat mitigation. For example, many threat vectors can be addressed using rational financial incentives that are often called detect-and-punish mechanisms. That is, when a cheating incident is detected, the miners punish the attacker financially. These approaches can use a game theoretic approach [25] to set the design parameters in a way that makes cheating unprofitable. By modeling interactions between the players as an economic game, the financial gain of all player strategies can be computed. Then, the parameters are configured to make honest behaviors more profitable than cheating. The same procedure can be used to quantify the damage these financial threats may cause. In other words, a threat that could give the attacker a big payoff should be prioritized over a threat that yields minimal profits. This reinforces the idea that cryptocurrencies require an expanded model for exploring risks and countering them.

Running example application. To illustrate this step in CompuCoin, we consider the distilled threat scenarios found in Figure 3. Both threats can be neutralized financially by designing proper techniques to make the client lock the payments in an escrow, along with a penalty deposit. The client loses the latter if he should cheat, perhaps by issuing invalid payments that carry his signature. The penalty deposit amount can be computed as the maximum payoff a client may obtain by cheating. This makes cheating unprofitable, and hence, unappealing to rational clients.

IV. Evaluation

To evaluate the effectiveness of ABC, we set up a user study that compares its performance against STRIDE [9], where we asked participants to build threat models for a simple cryptocurrency using these frameworks. STRIDE was chosen because it is an example of the models that system designers will turn to in the absence of a cryptocurrency-specific framework [12]. In what follows, we discuss the study methodology and its results.

A. Methodology

We recruited 53 participants, primarily masters students in systems security programs. We used five subjects as a pilot group to test and refine our materials. The remaining 48 participants were divided randomly into two groups of 24, one of which tested STRIDE, while the other tested ABC. Each framework session spanned three hours and was divided into two parts: a group tutorial and individual completion of threat models. The group tutorial started with a 20 minute overview of cryptocurrencies, followed by a one-hour training in the framework to apply. The participants were then given a 25 minute break to reduce any fatigue effects. The session resumed with a 15 minute overview of ArchiveCoin, the system that subjects will build a threat model for. ArchiveCoin is a simplified Filecoin [5]-inspired cryptocurrency system that focuses mainly on the service and its rewards in order to fit the study session period.

The individual completion of threat models spanned the remaining hour of the study session. Given that the allocated session period was short, we asked the subjects to look into just one threat category in Step 3, namely, the service theft of file retrieval. This category was not used in the tutorial clarifying examples to avoid biasing the results. Participants performed Steps 1 and 2 (system model characterization and threat category identification), and then submitted their answers. Only at this point, were they given the materials for Step 3 (eliciting threat scenarios), which asks them to consider only service theft of file retrieval. This was done so that participants who missed this threat when answering Step 2 could not alter their responses. Our study instrument and all supporting materials are available online [23].

B. Findings

Due to space limitations, we present a subset of the study results. Mainly, we discuss whether participants were able to build more accurate threat models when using ABC than when using STRIDE. An extended version can be found in [24].

To quantify accuracy, we compute the recall and precision values for the concrete threat scenarios found by each subject as compared to reference models we built for ArchiveCoin using each framework[2]. We found that participants who applied ABC produced a larger number of valid threat cases than STRIDE session subjects (the average recall values are 0.48 and 0.4, respectively). At the same time, ABC participants identified a lower number of irrelevant cases than those who applied STRIDE (the former scored an average precision value of 0.57, while it was 0.48 for the latter).

We believe that these results can be attributed due to several factors. First, ABC directed participants to consider the financial aspects of the system, which affected the elicited threat scenarios. We found that 88% of ABC session participants identified the payment process as a system component and the currency as an asset. On the other hand, only 38% of STRIDE's participants did so. Second, the collusion matrices helped ABC participants to reason about the threat space in

[2]The recall is computed as $TP/(TP+FN)$, and precision is computed as $TP/(TP+FP)$, where a true positive TP is a correctly identified threat, false negative FN is an undetected threat, and false positive FP is an incorrectly defined threat. The recall (precision) indicates how many valid (invalid) threats a subject defined. Both take values between 0 and 1.

a way that reduced random speculations. In addition, these matrices aided the subjects in spotting threats resulted from collusion between attackers. This is confirmed by our results, which showed that none of the subjects in the STRIDE session identified a possible collusion case between a client and servers, while 11 subjects in the ABC session identified this case[3]. This is expected as the threat tree patterns in STRIDE focus only on solo attackers, leaving spotting collusion to the subjects, which increases the required effort.

Third, ABC's threat categories, found in Table I, directed the participants to consider the impactful threats that must be investigated. Our results showed that only 13% of the participants in the STRIDE session identified the service theft of file retrieval threat in Step 2, while 71% subjects in the ABC session did so. This shows that the STRIDE threat categories, which are general and do not account for financial incentives, may cause system analysts to miss such threat types. In contrast, ABC categories guided the subjects toward service and payment related threats, even though they had little experience in such type of systems.

All these factors affected the overall correctness of the threat models built by the subjects. We found that the ABC session scored an average of 64% as compared to 50% for STRIDE. This is expected based on the previous results, where ABC was ahead of STRIDE in their scores.

V. Experiences

We applied ABC to several cryptocurrencies to understand its real world effectiveness. To do so, we built threat models for three cryptocurrencies that represent different stages in a system design lifetime: Bitcoin [1], a well-established system, Filecoin [5], a system that is under development and close to being launched, and CacheCash, a new system in its early development stages. Due to space limitation, we provide only a brief summary of the main findings regarding Filecoin and CacheCash models. A detailed version is available online [24].

Filecoin is a distributed file storage and retrieval network [5]. Recently, its team raised around $250 million in preparation for an official launch. One of the study authors spent 47 hours to develop a 882 case threat model for Filecoin. In this model, we found 3 unaddressed issues in Filecoin's design, mostly dealing with collusion cases that were not considered. We reached out to the Filecoin team, which mentioned efforts they have undertaken to resolve these problems. We withhold details about these issues until later as part of the responsible disclosure process.

CacheCash is a cryptocurrency that provides a distributed content delivery service. We developed ABC while designing CacheCash as a mechanism to cope with the substantial number (525) of collusion cases. During that time, we observed the benefit of rational financial incentives and game theoretic analysis to threat mitigation and risk management processes. To date, we found ABC useful for CacheCash in both the pre-design threat modeling step, and the after-design security analysis of the system modules. The CacheCash threat model will be published upon system launch.

[3]Most of them didn't provide a clear description of the attack. Hence, these incomplete answers *were not* counted as correct threats when grading Step 3.

VI. Conclusions

In this paper, we introduce ABC, a cryptocurrency-focused threat modeling framework. Its design is motivated by the observation that traditional threat modeling frameworks do not fit cryptocurrencies, thus leaving them vulnerable to unanticipated attacks. ABC introduces collusion matrices, a technique that allows designers to investigate hundreds of threat cases in a reasonable amount of time. Both the user study and the use cases confirm that our framework is effective in unraveling hidden threat cases. In an on-going work with the Linux foundation, we are applying ABC to cloud native security technologies. This shows the potential of ABC to improve the security of a wide array of distributed systems.

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.
[2] *CryptoCurrency Market Capitalizations*, https://coinmarketcap.com/.
[3] *Ripple*, https://ripple.com/.
[4] *Golem*, https://golem.network/.
[5] *Filecoin*, https://filecoin.io/.
[6] *Understaning the DAO Attack*, https://www.coindesk.com/understanding-dao-hack-journalists/.
[7] *Hackers Stole $32 Million in Ethereum*, https://thehackernews.com/2017/07/ethereum-cryptocurrency-hacking.html.
[8] *The Bitfinex Bitcoin Hack: What We Know (And Don't Know)*, https://www.coindesk.com/bitfinex-bitcoin-hack-know-dont-know/.
[9] M. Howard and S. Lipner, *The security development lifecycle*. Microsoft Press Redmond, 2006, vol. 8.
[10] S. Myagmar, A. J. Lee, and W. Yurcik, "Threat modeling as a basis for security requirements," in *SREIS*, 2005.
[11] M. Deng, K. Wuyts, R. Scandariato, B. Preneel, and W. Joosen, "A privacy threat analysis framework: supporting the elicitation and fulfillment of privacy requirements," *Requirements Engineering*, vol. 16, no. 1, 2011.
[12] D. Vandervort, D. Gaucas, and R. St Jacques, "Issues in designing a bitcoin-like community currency," in *FC*, 2015.
[13] A. Van Lamsweerde, "Elaborating security requirements by construction of intentional anti-models," in *IEEE ICSE*, 2004.
[14] M. Backes, A. Kate, P. Manoharan, S. Meiser, and E. Mohammadi, "Anoa: A framework for analyzing anonymous communication protocols," in *IEEE CSF*, 2013.
[15] M. Hollick, C. Nita-Rotaru, P. Papadimitratos, A. Perrig, and S. Schmid, "Toward a taxonomy and attacker model for secure routing protocols," *Computer Communication Review*, vol. 47, no. 1, 2017.
[16] R. Hasan, S. Myagmar, A. J. Lee, and W. Yurcik, "Toward a threat model for storage systems," in *ACM StorageSS*, 2005.
[17] R. Pass, L. Seeman, and A. Shelat, "Analysis of the blockchain protocol in asynchronous networks," in *EUROCRYPT*, 2017.
[18] J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. A. Kroll, and E. W. Felten, "Sok: Research perspectives and challenges for bitcoin and cryptocurrencies," in *IEEE S&P*, 2015.
[19] E. Androulaki, G. O. Karame, M. Roeschlin, T. Scherer, and S. Capkun, "Evaluating user privacy in bitcoin," in *FC*, 2013.
[20] A. Gervais, H. Ritzdorf, G. O. Karame, and S. Capkun, "Tampering with the delivery of blocks and transactions in bitcoin," in *ACM CCS*, 2015.
[21] L. Luu, D.-H. Chu, H. Olickel, P. Saxena, and A. Hobor, "Making smart contracts smarter," in *ACM CCS*, 2016.
[22] P. Moreno-Sanchez, N. Modi, R. Songhela, A. Kate, and S. Fahmy, "Mind your credit: Assessing the health of the ripple credit network," *arXiv preprint arXiv:1706.02358*, 2017.
[23] *Supplemental Material*, https://ssl.engineering.nyu.edu/papers/abc-material.zip.
[24] G. Almashaqbeh, A. Bishop, and J. Cappos, "Abc: A cryptocurrency-focused threat modeling framework," *arXiv preprint arXiv:1903.03422*, 2019, https://arxiv.org/abs/1903.03422.
[25] S. Tadelis, *Game theory: an introduction*. Princeton University Press, 2013.