TRIDEnT: Towards a Decentralized Threat Indicator Marketplace

Nikolaos Alexopoulos
alexopoulos@tk.tu-darmstadt.de
Technische Universität Darmstadt
Germany

Emmanouil Vasilomanolakis
emv@cmi.aau.dk
Aalborg University
Denmark

Stephane Le Roux
leroux@lsv.fr
LSV, ENS Paris-Saclay & CNRS,
Université Paris-Saclay
France

Steven Rowe
steven.rowe@web.de
Technische Universität Darmstadt
Germany

Max Mühlhäuser
max@informatik.tu-darmstadt.de
Technische Universität Darmstadt
Germany

ABSTRACT

Sophisticated mass attacks, especially when exploiting zero-day vulnerabilities, have the potential to cause destructive damage to organizations and critical infrastructure. To timely detect and contain such attacks, collaboration among the defenders is critical. By correlating real-time detection information (threat indicators) from multiple sources, defenders can detect attacks and take the appropriate measures in time. However, although the technical tools to facilitate collaboration exist, real-world adoption of such collaborative security mechanisms is still underwhelming. This is largely due to a lack of trust and participation incentives for companies and organizations. This paper proposes *TRIDEnT*, a novel collaborative platform that aims to enable parties to exchange network threat indicators, thus increasing their overall detection capabilities. *TRIDEnT* allows parties that may be in a *competitive* relationship, to selectively *advertise*, *sell* and *acquire* threat indicators in the form of (near) real-time peer-to-peer streams. To demonstrate the feasibility of our approach, we instantiate our design in a decentralized manner using Ethereum smart contracts and provide a fully functional prototype.

CCS CONCEPTS

• **Applied computing** → **Electronic data interchange**; • **Security and privacy** → *Network security*;

KEYWORDS

collaborative security; threat indicator sharing; trust; smart contracts; Ethereum

ACM Reference Format:
Nikolaos Alexopoulos, Emmanouil Vasilomanolakis, Stephane Le Roux, Steven Rowe, and Max Mühlhäuser. 2020. TRIDEnT: Towards a Decentralized Threat Indicator Marketplace. In *The 35th ACM/SIGAPP Symposium on Applied Computing (SAC '20),March 30-April 3, 2020, Brno, Czech Republic.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3341105.3374020

SAC '20, March 30-April 3, 2020, Brno, Czech Republic

ACM ISBN 978-1-4503-6866-7/20/03…$15.00
https://doi.org/10.1145/3341105.3374020

1 INTRODUCTION

In recent years, cyber-attacks have grown in impact and have affected millions of people and organizations all over the world. Recent examples, like the infamous Wannacry ransomware attack [13] and the Mirai Internet of Things (IoT) botnet [5], incurred losses calculated to amount to hundreds of millions of US Dollars [7]. Other attacks were aimed at acquiring restricted data, like the so-called Red October malware which stole vital information from government and research institutions. The latter operated undetected for five years, according to Kaspersky estimates [22]. Similar attacks are provisioned to become more common and disruptive, since attacker incentives grow as more people depend on interconnected devices (IoT) and critical infrastructure components. Worryingly, all attacks mentioned above were relatively simple, using known "old" vulnerabilities (e.g. Wannacry used the infamous NSA-related EternalBlue exploit), yet caused significant disruption. Mass attacks exploiting yet-unknown (zero-day) vulnerabilities can have orders of magnitude larger attack surface and therefore be destructive.

To timely detect signs of attacks and take appropriate action, firms and organizations use, among other countermeasures, Intrusion Detection Systems (IDSs). IDSs utilize techniques like signature matching (e.g. *Snort* [36] and *Bro* [30]), or sophisticated anomaly-based detection algorithms for the detection of unknown attacks (zero-days) [9]. However, given the advancing complexity and severity of attacks, isolated IDSs that only monitor one part of the network are not adequately effective. Attackers may have objectives that require a series of steps, including information gathering (e.g. mass network scans) and vulnerability assessment that may not be treated as a threat when viewed in isolation. Consequently, these complex, multi-step intrusion attempts (e.g. Advanced Persistent Threats) can only be detected by correlating information from different parts of the network [10]. Transfer of information is also of notable benefit in scenarios where adversaries mount mass attacks targeting a large number of defenders by utilizing similar techniques (e.g. the same zero-day vulnerability). Early warning signs (e.g. in the form of a network trace, a malicious URL or IP address) from defenders that were targeted by a given attack can greatly benefit others against (similar) future attacks against them. The latter scenario is particularly important, since such mass attacks are widespread in the wild [2], and as already noted, can have

destructive consequences. Threat indicator (TI) sharing is therefore critical in defending against modern attacks, as it greatly enhances the defensive capabilities of isolated IDSs.

The topic of collaborative security through TI sharing and correlation has attracted research interest in itself, with a variety of proposals on exchange mechanisms [25, 26, 41, 47], and standardization efforts [11, 46]. On the one hand, such proposals allow the exchanged data to retain its utility, and on the other hand protect the source of the data by filtering out sensitive information (e.g. IP addresses). However, there is still great reluctance from companies and organizations to share TI data, as financial incentives for doing so are not clear. This further highlights the central role of (security) economics [3, 4] in any real-world security problem. Companies are reluctant to engage in sharing activities, due to the fact that their competitors can take advantage of their (financial) investments in the detection effort and the ensuing sharing process, i.e. competitors may *free-ride*[1]. Apart from the basic cost of personnel and hardware/software resources dedicated to maintaining the sharing infrastructure, information leakage when openly publishing (even anonymized) TI may lead to additional risks. These can come in the form of law-infringement violations [39] or leaking information to an attacker concerning the defender's location and defensive capabilities [38]. Finally, distrust among the participants further aggravates privacy concerns, while also limiting confidence in the received information. For these reasons, TI sharing is viewed cautiously by companies and organizations [32].

Existing TI sharing platforms (e.g. MISP [43], IBM's "X-Force Exchange"[2], or Facebook's "ThreatExchange"[3]) do not consider incentives and trust, and are intended to be operated by central trusted third parties. The latter is an important limitation, since organizations (or even governments) in any kind of a competitive relationship will be uneasy with providing valuable data to services fully controlled by their competitors.

Motivated by the aforementioned challenges, we propose a design towards addressing the issues of incentives and trust via a fully decentralized TI marketplace that we call *TRIDEnT*[4]. *TRIDEnT* is an open, carefully designed data stream marketplace that offers the required functionality for entities to share TI, while providing the environment and mechanisms for trust relations between them to develop. Our contributions can be summarized by the following points:

The TRIDEnT platform design: *TRIDEnT* is a collaborative platform for TI sharing that facilitates the creation of P2P channels for collaboration among interested parties. In *TRIDEnT*, indicator producers offer their data in the form of live streams[5] , selectively to parties of their choice, thus creating a sharing overlay based on their trust relations. To attract interested collaborators, they advertise their streams by including information about the data, in the form of tags. For example, producers may advertise tags relating to the type of attacks (e.g. Distributed Denial of Service (DDoS), malware), the detector (e.g. IDS, honeypot), the type of network (e.g. backbone, corporate), and so on. Interested parties can buy streams with some form of currency[6] to pay the producers and start streaming in real time. A rating and trust management system is included to ensure that participants offer good quality TI data.

Instantiation on Ethereum: We showcase the feasibility of our system in an open, decentralized setting (anyone can participate), with an implementation on the *Ethereum* platform. For readers unfamiliar with the basic functionality of Ethereum, a brief overview is given in Section 2. We describe the design along with the basic components of the prototype and show that transaction fees incurred by the system are projected to remain negligible. Note that the same design can straightforwardly be adapted to a permissioned setting, e.g. by implementing it on *Hyperledger Fabric* [8], in the case of a closed group of known, yet selfish and competitive organizations.

The remainder of this paper is structured as follows. Section 2 offers some background information, required for the comprehension of the paper. Section 3 presents the collaborative platform and focuses on its architecture and operations. Then, Section 4 presents a prototype implementation of *TRIDEnT* and its evaluation. Section 5 goes over the related work, while Section 6 discusses limitations of our work and reports our conclusions.

2 BACKGROUND

In this section we provide some basic background about *i)* collaborative security and *ii) Ethereum* and smart contracts.

Collaborative Security and Collaborative Intrusion Detection Systems (CIDSs) The term collaborative security is an encapsulation of the idea that *combining knowledge from different sources can be of benefit for the detection and mitigation of attacks*. The reasoning behind this argument is well studied [26, 41, 47], and can be summarized in the following: detectors, e.g. anomaly detection algorithms, can be improved by enhancing the input data, TI correlation is more effective when the data volume increases, a number of attacks, e.g. malware spreading, can be contained even before seen locally when they are anticipated as a result of collaboration. For instance, Böck et al. [6] recently showed that correlation of data from different sensors is necessary to enumerate advanced P2P botnets. Collaborative Intrusion Detection Systems (CIDSs) further formulate the aforesaid idea by implementing it in the form of a system [41]. Besides academic work, e.g. [15, 42], there have been various attempts to realize CIDSs [33, 40]. Nevertheless, the majority of the proposed systems are either theoretical or assume that participants are trusted and are willing to collaborate.

Ethereum and smart contracts *Ethereum* [45] is a decentralized platform (distributed ledger) for Turing-complete applications, facilitated by smart contracts, with a native cryptocurrency called *ether*. The state of the platform is collectively maintained by its users via a consensus mechanism, thus guaranteeing its correct operation under reasonable security assumptions (honest majority of computing power and relatively good network connectivity). Thus, the platform is as if it were operated by a trusted party with a public internal state. Smart contracts can be written in high level

[1] Actually free-riding is the Nash equilibrium in many security information sharing models [24]
[2] https://exchange.xforce.ibmcloud.com/
[3] https://developers.facebook.com/programs/threatexchange/
[4] short for Trustworthy collaboRative Intrusion DETection.
[5] these TI streams are also commonly referred to as Threat Intelligence feeds.
[6] For the rest of the paper we refer to the exchange medium as the *TRIDEnT token*, however any currency can be used, i.e. a dedicated token is not technically necessary.

languages (e.g. *Solidity*), compiled and executed by the *Ethereum Virtual Machine (EVM)*.

3 THE TRIDENT SHARING PLATFORM

We begin the presentation of our platform by familiarizing the reader with its basic components and their interaction, along with a high-level overview of their functionality. We then proceed to describe in more detail the more interesting parts and design choices. An avid reader can infer all details about our platform from the smart contract pseudocode of our implementation (see Section 4).

3.1 System architecture and overview

TRIDEnT can be conceptually dissected into the following layers:

Distributed Ledger Layer: The system is built on top of a distributed ledger (DL), e.g. the Ethereum platform [45], which offers strong consistency and availability guarantees and allows participants to execute arbitrary (Turing-complete) operations on its state. This is the base layer of the framework and lays the trust foundations for the following layers, emulating a trusted third party. The benefits of using a DL as a sharing infrastructure have been documented recently [1, 44].

Trust Management Layer: As participants can behave maliciously at times and towards certain parties, a trust management mechanism (i.e. generalized reputation system) is necessary to support good behavior. Ratings from buyers follow transactions (stream establishment), and are stored on the DL. At each time, a peer is able to calculate a local trust score for each other peer, via a trust calculation algorithm of her choice. In our design (see Section 3.3) we use an adaptation of the Bayesian mechanism of [34].

Marketplace Layer: The system provides economic incentives for honest behavior by utilizing an economic mechanism. Tokens can be thought of as a currency special to the system, used to buy streams from other participants. In the case where a native cryptocurrency is offered by an underlying distributed ledger, then this can be used as a token.

Data Overlay Layer: After establishing a connection, a seller and a buyer open a channel where TI data and tokens are exchanged. Multiple such channels create sharing overlays. We employ off-chain transactions (see the Raiden network[7]) to enable seamless and near-instantaneous token transfers.

An intuitive description of the basic functionalities of the platform, along with the connections among the layers can be seen in Figure 1. A simplified example operation scenario follows. Organization B, acting as a TI *seller* advertises the information they can offer (e.g. TI from their industrial IDS) on the marketplace. Organization A, acting as a TI *buyer* (i.e. party interested in acquiring alert data) queries the marketplace where advertisements of TI streams are posted. The buyer also computes trust scores for each seller by consulting previous ratings, and decides whether she wants to make a subscription offer matching a stream's requested price. Assuming the buyer makes an offer for a stream, and the corresponding seller decides to accept it (the seller can also consult the buyer's trust score as risk in these transactions is bidirectional), a data stream and a payment channel are established. The buyer may now rate the stream (and indirectly the seller) based on its perceived quality and whether or not it matches the advertisement. The rating is stored on the ledger. Details about the platform's operation follow.

[7] https://raiden.network/

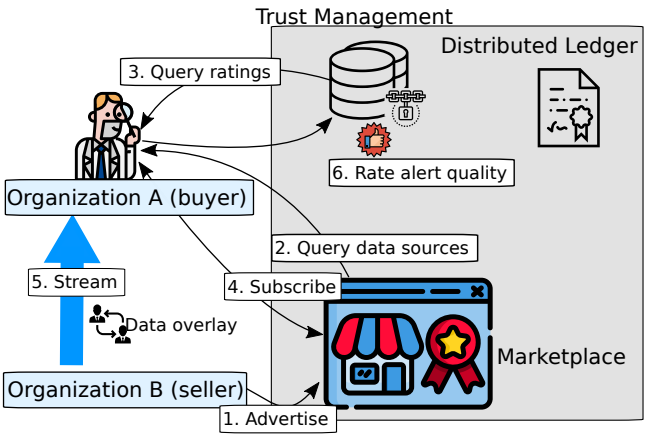

Figure 1: *TRIDEnT* workflow. Arrows show data flow.

3.2 Marketplace functionality

The core functionality of the platform is summarized by the marketplace function definitions of Table 1, which are to be instantiated by a smart contract. Our actual instantiation of these functions is reserved for Section 4. Users register on the platform by providing their public key and "burning" an amount of *ether* to create an initial trust value (`register`). This is a widespread technique to mitigate sybil attacks, further discussed in Section 3.3. Then they can advertise streams (`advertise`), make offers (`mkOffer`) for existing advertisements, or accept offers (`accOffer`) for their own advertisements. In particular, when making an offer, a security deposit has to be provided in order to incentivize the act of providing ratings (see below). The marketplace also offers functionality for deleting offers (`delOffer`) and unsubscribing from streams (`unsubscribe`), while enforcing the necessary constraints. Finally, ratings can be provided via calling the `rate` function. A deposit is used to incentivize buyers to provide ratings for streams. Note that our design is generic and can in theory provide incentives and trust for any kind of information sharing platform. However, security information sharing is particularly challenging, as it will become obvious in the following paragraphs, and we therefore focus on it. In the remainder of this section we present some notable characteristics of *TRIDEnT*'s operation.

TI advertisement format: A TI advertisement includes three important characteristics of the stream, namely the expected mean throughput of the stream (e.g. TI per hour), the price of streaming (e.g. tokens per indicator batch), and a list of strings (*tags*) offering information on the type and origins of the TI in the stream. A simple example is shown in Table 2. Note that the formats presented in this section serve as simple abstract templates to showcase the functionality of *TRIDEnT*, meaning that they can be readily enhanced as required.

Function	Description	Constraints
`register`	used for initial registration burns ether	
`advertise`	create new advert. with chosen tags	
`rmAdvert`	remove given advert. + related offers and subscr.	
`mkOffer`	create offer for given advert.	deposit has to be provided
`delOffer`	delete given offer	caller has to be the advert. publisher or offer creator
`accOffer`	delete offer and create subscr.	caller has to be the advert. publisher
`unsubscribe`	delete subscription	caller has to be the advert. publisher or the subscriber
`rate`	add rating	only one rating/subscr. caller has to be subscriber timer must not be expired

Table 1: Marketplace function definitions with constraints.

With regard to the tags, we have classified this information into three categories: (i) type of the detector (e.g. honeypot, IDS, firewall, etc.), (ii) type of the network (e.g., university network, home network, backbone), and (iii) type of observed attacks (e.g., DDoS attacks, malware, botnet, APTs, port scans, etc.). Here we note that the third type of information is valid only for streams that provide historical data (not live data). Whereas in first inspection, it may seem useless to enable exchange of stale TI in our system, this is often useful. For example, companies specializing in developing solutions for detecting and anticipating specific kinds of attacks would be very interested in TI that have been labeled accordingly.

Advertising and subscribing to streams. Participants of the platform can subscribe to TI streams of other parties and also advertise their own streams on the ledger. Participants are typically IDSs or honeypots but also other CIDSs or humans, e.g., security experts. For the platform as a whole, it is desirable that parties who consume TI data from the other parties, also provide their own data. An advertisement contains the id of the publisher (seller) that can be used to calculate her trust value, and tags that describe which kind of TI the subscriber (buyer) can expect, as explained above. This allows the node to browse all available TI streams.

```
Publisher: GoodIDS
Expected throughput: 10/hour
Price: 1$/indicator
Detector type: IDS
Type of network: industrial
Type of attacks: DDoS, botnet
```

Table 2: Example advertisement.

Stream establishment and operation. Figure 2 presents the stream establishment protocol of *TRIDEnT*, involving a Buyer, a Seller and the system's smart contract, which facilitates operations. First, involved parties create and register a (public) key to be used for authentication and encryption. In practice, an implementation of OpenPGP (e.g. GnuPG) could be used for all cryptographic operations in this protocol, potentially with different keys for authentication and encryption, as per standard practice. Upon deciding to make an offer for a stream corresponding to an advertisement S_i, the Buyer calls the `mkOffer` function of the smart contract, thus transmitting the offer to the DL. Upon deciding to accept the offer, the seller publishes on the ledger the signed and encrypted address of the socket (IP address, port) where the buyer can connect to start streaming. The address is encrypted with the buyer's public key (initialized upon registration) and ensuing connections to the socket (assuming initial authentication is successful) are encrypted and authenticated. Execution paths concerning decisions not to accept the offer are pruned from Figure 2 for brevity.

Regarding the TI data format, we propose the state-of-the-art STIX format [29], although other formats could also be used. For a comparison of TI exchange formats see [28]. A general template is given in Table 3. A specific STIX indicator in JSON format signaling a malicious URL follows in Table 4. We refer the reader to the documentation of STIX[8] for a more detailed description. *TRIDEnT* is agnostic to the specific format used.

```
Time: creation/sending time
Source: origin of attack
Target: target of attack
Classification: name and/or CVE
Assessment: e.g. severity, potential impact, etc.
```

Table 3: Generic TI format.

```
{
"type": "indicator",
"id": "indicator–9299f726-ce06-492e-8472-2b52ccb53191",
"created_by_ref": "identity–39012926-a052-44c4-ae48-…",
"created": "2017-02-27T13:57:10.515Z",
"modified": "2017-02-27T13:57:10.515Z",
"name": "Malicious URL",
"description": "This URL is potentially associated with malicious
 activity and is listed on several blacklist sites.",
"pattern": "[url:value = 'http://paypa1.banking.com']",
"valid_from": "2015-06-29T09:10:15.915Z",
"labels": ["malicious-activity"]
}
```

Table 4: Example STIX indicator (JSON format). More examples can be found in the docs[9].

[8] https://stixproject.github.io/about/
[9] https://oasis-open.github.io/cti-documentation/stix/examples

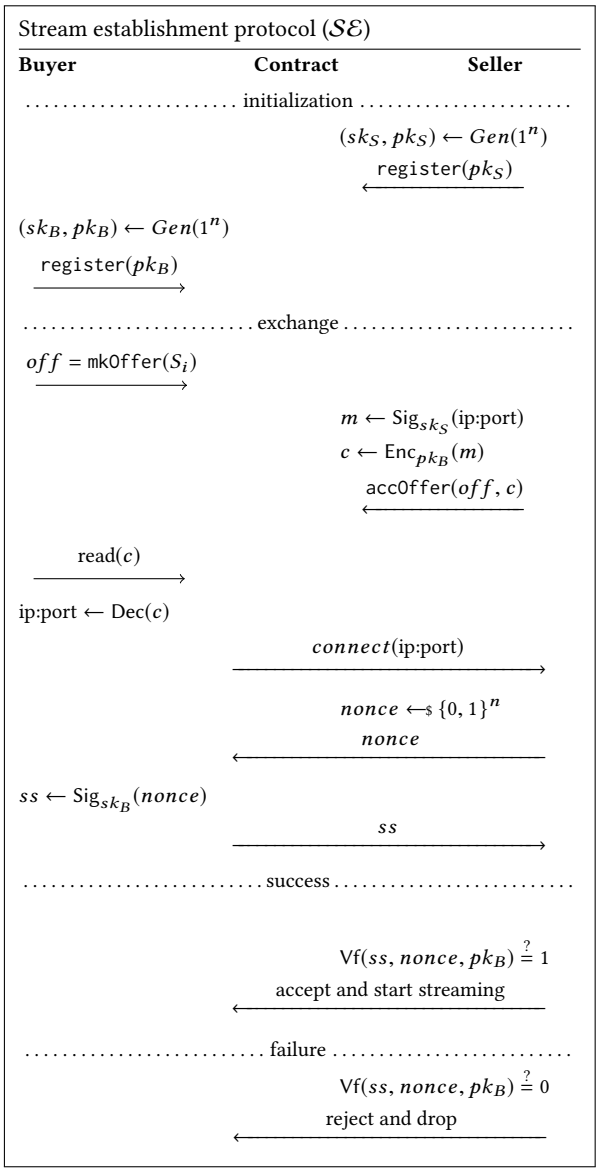

Figure 2: Stream channel establishment via a smart contract. Notation follows standard practice, with *Gen* the key generation operation (with security parameter 1^n), Sig **the digital signature operation,** Enc **and** Dec **the operations of encryption and decryption, and** Vf **signature verification. Typewriter font is used for calls to the marketplace functions. Some signature verification operations are omitted for brevity.**

In order to support real-time micro-payments, we employ off-chain payment channels, as provided by the Raiden Network. Specifically we utilize the μ-Raiden variant, which supports only one receiver but is simpler and cheaper in terms of transaction costs. A uni-directional payment channel is created upon successful establishment of a stream between the buyer and the seller. Subsequent payments take place via the channel, thus allowing per-indicator payments with practically zero transaction fees.

3.3 Building trust

As recounted earlier, trust among the participants of collaborative security mechanisms is of paramount importance. The decentralized smart-contract-based design of *TRIDEnT* does not require a trusted party to act as the operator of the sharing service, thus avoiding the need for the participants' reliance on a common provider. However, some parties might try to profit from the streams of the other parties while themselves providing fake or bad quality data. There is also a risk of malicious participants that may try to destroy the marketplace, e.g., by providing wrong TI data or by bad-mouthing honest parties. These violations cannot be controlled and ruled out by smart contracts. Therefore, some form of trust management, such as a reputation system, is required in order to protect participants from malicious insiders that disseminate low quality, malformed, or misleading data. *TRIDEnT* comes with a built-in trust bootstrapping and rating system, thus providing the foundations for trust assessment.

Trust bootstrapping: Since new parties may enter the system at any time (*TRIDEnT* is meant to be an open system), a way to bootstrap their standing in the community is needed, even if they do not have real-world social ties to other participants. At the same time, this bootstrapping mechanism should mitigate sybil attacks. In *TRIDEnT*, trust bootstrapping is achieved via proof-of-burn. To register in the system, a participant "burns" an amount of cryptocurrency in order to prove her commitment to the community. Cryptocurrency "burning" refers to the act of rendering an amount of currency provably un-spendable, equivalent to actual burning of currency notes. The amount required to achieve a baseline trust value and start interacting in the *TRIDEnT* community should remain low in order not to be considered an entry barrier for new participants. On the other hand, the amount required to achieve considerably higher initial trust should increase quickly, as to make it difficult for malicious participants to acquire high initial trust and harm the system. The exact baseline amount is a deployment decision, not addressed in this paper.

Rating: Transaction ratings are a widely used and empirically effective tool for building trust in marketplace environments. Rating in *TRIDEnT* is intertwined with the other marketplace operations. After a stream is established, the buyer can rate the seller by calling the `rate` function of the smart contract. This rating is thus part of the global state of the underlying Distributed Ledger (DL) and therefore visible to all participants. After the stream is closed, the buyer can finalize her rating by calling the `rate` function once again. This is intended on the one hand to allow early rating (since streams may be very long-lived), and on the other hand to discourage sellers from providing good quality data up until they are rated positively, and then decreasing quality. The perceived quality of a stream depends on whether data provided are judged to be real, useful, and corresponding to its advertisement.

A major problem faced by rating systems is user unwillingness/laziness to provide ratings, especially in the case where it

involves extra effort or possible additional transaction fees. In *TRIDEnT*, users are monetarily incentivized to provide ratings by mechanisms built in the system. Specifically, upon making an offer, the buyer deposits a small but not negligible fixed fee to the marketplace smart contract. This fee is returned to the buyer if the offer is rejected or, in two rounds, upon submitting a rating[10]. It is easy to see that if the fixed fee is considerably larger than the transaction fee required to call the `rate` function, buyers are incentivized to follow up any purchase with a rating.

The mechanism described above and implemented in our system, offers clear incentives for buyers to submit ratings. However, incentives for providing *correct* ratings may also be needed. Although these fall outside of the scope of this paper, *TRIDEnT* can straightforwardly support algorithms proposed in literature to overcome this problem, e.g. the elegant idea proposed by Jurca and Faltings [21]. The idea is to establish a side-payment channel organized by a set of broker agents that buy and sell ratings. They reward (pay) an agent who submitted a rating if the following rating (coming from another agent) agrees with hers. Under well-defined assumptions (the next experience is more likely to be the same as the previous: $Pr[C^i_{t+1}|C^j_t] > 0.5$ for $i = j$), truthful reporting is a Nash equilibrium. In *TRIDEnT*, no broker agents are required; the mechanism can be added to the existing protocol and executed by smart contracts. In this sense, *TRIDEnT* is an extensible platform.

Local trust computation: The local trust computation algorithm employed by each party to decide whether or not to engage in exchange activities with another party is subject to choice, and technically not part of the core *TRIDEnT* infrastructure. Generally, trust decisions in this setting are more complex than just computing a reputation score and require human engagement. However, we still consider trust scores as valuable decision-supporting tools, and in our instantiation we use an adapted version of *CertainTrust (CT)* [34] to calculate a trust score for each participant. In contrast to simplistic "average rating" trust representations, CT, since it is based on Bayesian statistics principles, can model the perceived accuracy of the derived trust value, as well as incorporate prior information (corresponding to Bayesian priors). More details about our construction follow.

In the Bayesian representation of *CertainTrust / CertainLogic* [18, 34, 35], *trust* is calculated as the expectation/probability:

$$E = c_e \cdot t + (1 - c_e) \cdot f \tag{1}$$

where t is the point estimate of the (probability of success) parameter of a binomial distribution, f is the a priori expectation of the same parameter, and c_e is a factor that "fades" the effect of the prior value, as more pieces of evidence are taken into account, in accordance with Bayesian statistical inference. Evidence in this model are either positive or negative experiences (binary ratings). The number of positive and negative experiences is denoted by r and s respectively, and $n = r + s$. The mapping between the evidence space (r, s) and the Bayesian probability space is given by:

$$t = \begin{cases} \frac{r}{r+s} & \text{if } r + s > 0 \\ 0 & \text{else} \end{cases} \tag{2}$$

[10] According to our current implementation, the fee is returned after the first rating a buyer submits (see lines 44-48 of Smart Contract 1).

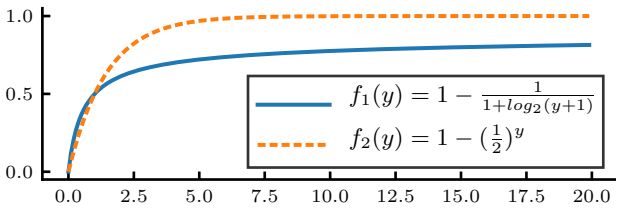

Figure 3: Comparison of proof-of-burn candidate functions.

$$c_e = \begin{cases} 0 & \text{if } n = 0 \\ \frac{N \cdot n}{2 \cdot w \cdot (N-n) + N \cdot n} & \text{if } 0 < n < N \\ 1 & \text{if } n \geq N \end{cases} \tag{3}$$

where w is a normalizing value we set to 1, and N is a user-set threshold number of evidence needed to achieve a desired level of significance of the estimate (in the sense of a confidence interval). It is easy to see from Eq. (3) that the value of c_e increases with increasing number of evidence n, as intended.

We set the Bayesian *a priori* expectation f regarding the initial trustworthiness of an entity, by utilizing a proof-of-burn (PoB) bootstrapping mechanism[11]. The function $f^* : y \in \mathbb{R}^+ \rightarrow f \in [0, 1]$, relating the amount of cryptocurrency units burned, and the resulting baseline trust, where $y = \frac{x}{c}$ the fraction of the amount of cryptocurrency burned and c the baseline value for achieving initial trust of 0.5, should satisfy the following properties:

$$f^*(0) = 0,\ f^*(1) = \frac{1}{2},\ \lim_{r \to \infty} f^*(y) = 1,\ f^{*'}(y) > 0,\ f^{*''}(y) < 0 \tag{4}$$

After considering different functions, we arrive at the definition of $f^* \equiv f_1$, with:

$$f_1(y) = 1 - \frac{1}{1 + log_2(y + 1)} \tag{5}$$

The definition of f_1 satisfies the properties of Eq. (4), and additionally, its growth rate is suitable to our needs. In Figure 3, both f_1 and $f_2 = 1 - (\frac{1}{2})^y$ (proposed in [49]), are depicted. We see that f_1 rises faster up to the value of 0.5, and then its growth rate drops significantly, especially with respect to f_2. By adopting f_1, users will be able to acquire a baseline trust value of 0.5 by spending a reasonable fee, but to acquire significantly larger initial trust, they will have to pay exponentially more. Even by paying 20 times the default fee, their initial trust would not be much higher than 0.8.

3.4 Privacy discussion

TRIDEnT, by design, offers increased privacy assurances for participants by allowing the formation of P2P overlays based on trust, rather than indiscreetly publishing TI data. Indicators are assumed to be sanitized according to best practices (e.g.,[25]), limiting the exposure of private information to collaborators. Furthermore, data sanitization could be executed depending on the trust relation between the collaborators, allowing for less sanitized – more valuable TI to flow, as trust between collaborators grows. Overall, our design offers a solid foundation for data exchange, able to support existing and upcoming privacy-enhancing techniques based on TI sanitization, bloom filters etc., which are not the focus of this paper.

[11] According to our current implementation, the PoB operation is performed at registration (see line 3 of Smart Contract 1).

Apart from the TI data themselves, another important source of privacy leakage is the marketplace metadata (e.g. who is buying from whom). Although participants may use pseudonyms in the marketplace, the metadata may need to be protected with techniques similar to [37]. This is left for future work.

3.5 Attacks and defenses

Apart from the usual attacks and corresponding defenses against trust / reputation systems in the marketplace context (see e.g. [19]), there are some specific notable issues that *TRIDEnT* is faced with, that may not be obvious on first sight.

Target bad-mouthing attack: In this scenario, the adversary attempts to artificially lower the reputation of the target party, so that he can discredit her reporting competence before attacking her. This way, the adversary would be able to lower the chance of the attack evidence being spread among the defenders. The most common way of performing such an attack is creating multiple identities and giving a lot of negative ratings. Since joining the system comes at a non-negligible cost and submitting a rating requires starting a stream, which incurs respectable costs, the attack is expensive for an adversary to perform, and not scalable. Remember, that the principal aim of *TRIDEnT* is to mitigate mass attacks, not targeted ones.

Stream reselling: In this attack, an adversary resells TI he has bought from sellers, to other parties. Apart from this being a problem for the incentives mechanism (the reseller can practically free-ride), this attack can also compromise the security of the system, as TI may contain sensitive information and are intended for use by the designated recipient. Enforcing watermarking on security TI seems almost impossible, as the distinctive information would be easy to locate and remove. A solution with the use of fake TI as watermarking would also reduce the value of the TI as a whole, which goes against the goals of the system. Therefore, defense against reselling attacks relies on the same mechanism as defense against low-quality data, i.e. the trust management system already described in the previous sections. Hence, sellers also face risk and should assess their trust in their buyers. That being said, studying the reselling attack in isolation is a challenging open problem for the community. For example, mechanisms that would allow honest buyers to discern resellers could be valuable for the system. One could envision a solution where honest participants continuously and randomly check a subset of a node's incoming and outgoing streams to detect resells. This check could be performed in a privacy-preserving manner via secure multiparty computation. These solutions could of course find applications in other data sharing applications.

After presenting the basic functionality of *TRIDEnT*, we proceed to validate the basic principles behind it with a formal probabilistic model.

4 PROTOTYPE AND EVALUATION

In this section, we present our prototype implementation of *TRIDEnT*. Specifically, we present *eth-TRIDEnT*, running on Ethereum. The code can straightforwardly be modified to run on permissioned platforms, e.g., Hyperledger Fabric.

Smart contract pseudocode: In Smart Contract 1 we provide a pseudocode description of the most notable parts of the backbone smart contract $\mathbb{B}_{\text{market}}$, which provides the basic functionality of *TRIDEnT*. We loosely follow the notation of [23], with \$ prepended to variables associated to some form of currency (tokens), and *CLT* standing for the client (e.g. Ethereum wallet address) who calls the function. The initialization of the data structures is omitted for space reasons. The logic of the contract is not trivial, and therefore we proceed to provide a brief overview.

The *register* function allows clients to register in the system with a public key pk and initializes their balance (ledger) and ratings set. This initial balance provided in Ether cannot be retrieved from the contract and is therefore considered a form of coin burning even though the value invested is useful inside the system[12] . Only registered clients can call any of the following functions. The *advertise* function allows clients to post advertisements accompanied by tags (see line 10) in the marketplace. The function *rmAdvert* allows clients that have posted an advertisement to remove it from the marketplace, while returning fees and deposits held by the contract to their rightful recipients. The *mkOffer* function adds an offer to a specific advertisement already existing in the marketplace. A fee (initial payment to the seller if the offer is accepted) and a deposit (incentive to provide rating) are withheld by the contract. The *delOffer* function allows the offer maker or the advertiser to delete (take back or turn down respectively) an offer and send the tokens withheld by the contract to the offer maker. Moving on, the *accOffer* function allows clients that have made advertisements to accept offers for them, claiming the initial fee and providing the encrypted (ip:port) information needed to establish the stream (see $\mathcal{SE}$ protocol of Figure 2). The *unsubscribe* function allows clients that have either posted an advertisement or are subscribed to one, to remove their subscription. Finally, *rate* allows clients to provide a rating about a client that made an advertisement about a stream they are subscribed to. Upon their first rating for a given subscription, they receive their deposit back.

eth-TRIDEnT: We implemented our smart contracts in Solidity v0.4.25. We created two contracts, namely `AlertExchange.sol` and `Token.sol`. The former is the heart of the prototype, implementing the functions described in pseudocode above; the latter implements a simple ERC-20 token used as the system's currency. In order to mint the token, a node has to burn ETH. Micro-transaction channels were implemented by calls to the already deployed μ-Raiden contract. When it comes to evaluating smart contracts, transaction costs (in Ethereum: Gas fees), i.e. fees paid by users to miners, are an important metric and the usual way of arguing about the feasibility of an approach.

In Table 5, we show the transaction costs of the functions of the Alert Exchange contract, as calculated after deployment to the Rinkeby test network. The smart contracts can be found at the following addresses on Rinkeby:

–**Alert Exchange:** 0x682528b9cc9b74ca00efb004a4619fabd6f5cd69
–**Token:** 0x380ab69f3230c1990c9bbf4ecdcfefc6bd26501c
–**Raiden Channels:** 0x83aeb45854e1ac54f5d9fa42fd7a79b398aa50cf

The code of the main contract (`AlertExchange`) was verified and

[12] Traditional "wasteful" burning can naturally also be employed.

Smart Contract 1: The Backbone contract $\mathbb{B}_{\text{market}}$

```
1  Function register:
2      Upon receiving (register, $payment, pk) from CLT:
3      ledger[CLT] := $payment
4      ratings[CLT] := ∅
5      key[CLT] := pk
6      parties := parties ∪ (CLT)
7  Function advertise:
8      Upon receiving (advertise, tags) from CLT:
9      ASSERT CLT ∈ parties
       /* This assertion is repeated in all subsequent funtions,
          but omitted for clarity                                  */
10     adv = (tags, CLT, {∅}, {∅})
11     adverts := adverts ∪ adv
12 Function rmAdvert:
13     Upon receiving (rmAdv, adv) from CLT:
14     ASSERT adv ∈ adverts, CLT = adv[1]              // see line 10
15     adverts := adverts \ adv
16     Return pending offer fees and deposits to offer makers
17 Function mkOffer:
18     Upon receiving (makeOffer, adv, $fee, $deposit) from CLT:
19     ASSERT adv ∈ adverts
20     ledger[CLT] := ledger[CLT] - $fee - $deposit
21     ledger[this] := ledger[this] + $fee + $deposit
       /* tokens transferred to the contract itself (this)      */
22     offer = (CLT, fee)            // offer[0] = CLT, offer[1] = fee
23     adv[2] := adv[2] ∪ offer
24 Function delOffer:
25     Upon receiving (dlOffer, offer) from CLT:
26     ASSERT ∃ adv ∈ adverts: offer ∈ adv[2], (CLT = offer[0] OR CLT =
         adv[1])          // CLT is advert owner or has made the offer
27     adv[2] = adv[2] \ offer
28     Return $fee and $deposit to offer maker
29 Function accOffer:
30     Upon receiving (acceptOffer, offer, c) from CLT:
31     ASSSERT ∃ adv ∈ adverts: offer ∈ adv[2], CLT = adv[1]
32     sub = (offer[0], false, offer[1], c)              // c = Enc(ip:port)
33     adv[3] = adv[3] ∪ sub
34     ledger[CLT] := ledger[CLT] + offer[1]     // claim initial $fee
35     ledger[this] := ledger[this] - offer[1]
36     delOffer(offer)
37 Function unsubscribe:
38     Upon receiving (unsubscribe, sub) from CLT:
39     ASSERT ∃ adv ∈ adverts: sub ∈ adv[3], (CLT = adv[1] OR CLT =
         sub[0])                  // CLT is advert owner or subscriber
40     adv[3] = adv[3] \ sub
41 Function rate:
42     Upon receiving (rate, sub, rating) from CLT:
43     ASSERT ∃ adv ∈ adverts: sub ∈ adv[3] CLT = sub[0],    // CLT is a
         subscriber
44     ratings[adv[1]] := ratings[adv[1]] ∪ rating
       /* if rating for the first time                          */
45     if sub[1] = false then
46         ledger[CLT] := ledger[CLT] + deposit
47         ledger[this] := ledger[this] - deposit
48     sub[1] := true
```

can be easily inspected at Etherscan[13]. For the calculation of the transaction costs, the gas price was set to 4 Gwei ($4 \cdot 10^{-9}$ ETH) and the ETH/EUR exchange rate to 180.00, according to observations at the time of writing[14]. Among the transaction costs of the methods, the cost for accepting an offer stands out. This is because the method

[13] https://rinkeby.etherscan.io/address/0x682528b9cc9b74ca00efb004a4619fabd6f5cd69

[14] According to https://ethgasstation.info/. September 2018.

Function	Cost			
	Gas	*Gwei*	*EUR*	
		current	*current*	*peak*
deploy	3 994 723	15 978 892	2.88	99.68
register	54 672	218 688	0.04	1.36
advertise	173 279	693 116	0.12	4.32
rmAdvert	41 257	165 028	0.03	1.03
mkOffer	194 381	777 524	0.14	4.85
delOffer	25 820	103 280	0.02	0.64
accOffer	756 014	3 024 056	0.54	18.86
unsubscribe	34 139	136 556	0.02	0.85
rate	46 663	186 652	0.03	1.16

Table 5: Trans. costs at the time of testing (Sep'18) and the peak of Ethereum price and network load (Jan'18) (last col.).

contains the deletion of an offer, as well as the creation of a subscription. Subscriptions are the contracts' biggest data structures, which makes them relatively expensive to save. As an example, the establishment of a stream would cost a total of under 1 EUR at the time of writing and under 30 EUR considering the worst-ever price and network traffic[15]. As these streams are projected to be relatively long-lived, these costs are expected to be negligible for firms. Note that the contract deployment cost is a one-time cost for the whole lifetime of the system. The implementation of the `AlertExchange` contract consists of 423 lines of Solidity code and a detailed report on the engineering obstacles we encountered will follow in an extended version of the paper.

Client-side application: In order to make interaction with the deployed smart contracts user-friendly, we also developed a client-side command-line application consisting of around 2 000 lines of JavaScript code. The application calls the smart contract methods using the web3 JavaScript API[16]. It enables all necessary functionalities for viewing, advertising, subscribing, and rating TI streams, as well as calculating trust values for participants. Furthermore, the client-application allows parties to provide an endpoint from which subscribers can download TI, authenticates the subscribers and validates that the TI batches are paid for.

5 RELATED WORK

In this section we focus on the related work with regard to incentives and trust for CIDSs, as well as on collaborative platforms that exist nowadays. For a summary of CIDSs the reader can refer to Section 2.

Gal-Or and Ghose [16] propose a game-theoretic model for information-sharing in cyber-security. They conclude that although information sharing is beneficial to firms, with no additional incentives and anti-free-riding mechanisms, they may not be encouraged to participate truthfully. This work acted as a motivation for *TRIDEnT*. Despite the significant work that has been done in the areas of collaborative security [26] and CIDSs [41, 47], only a very small portion of it deals with incentives and trust among the participants. That is, the majority of the related work assumes that all the participants are honest, trustworthy and willing to collaborate. Duma et al. [12], were one of the first to propose a simple trust model

[15] 15th January 2018. ETH/EUR: 1134.20, Average gas price: 22 Gwei.

[16] https://github.com/ethereum/wiki/wiki/JavaScript-API

for CIDSs. However, their model is prone to insider attacks (e.g., betrayal attacks). Fung et al. also examined this field, proposing more advanced mechanisms in their incremental works [14, 15], and attempted to deal with the aforementioned insider attack challenge.

Zhu et al. [48], touch the topic of incentives in CIDSs and propose a game-theoretic model that ensures that peers of the system contribute equally (in terms of computational power). This work is very relevant for us, however it is limited only to incentives that are connected to the computational power of the overall system as well as to the resource allocation problem. Hence, this work does not take into account the bigger picture of collaborative security; e.g., the incentives for an organization to join a collaborative ecosystem, the economics of such a system, etc. Similarly, Guo et al. investigate incentives for CIDSs in mobile ad-hoc networks [17]. They propose an auction process using virtual credits and model cooperative detection as an evolutionary game. Although they also use a form of virtual currency, their setting is substantially different than ours (completely different assumptions) and they do not consider trust relations among collaborators. Jin et al. [20] propose a privacy-protection mechanism for CIDS where participants are trustworthy and misreport (share false information) with a known probability to protect their privacy. In our view, misreporting is not an option for real-world systems, and privacy must be preserved by proper data sanitization and forming collaboration overlays with trustworthy partners, as enabled by *TRIDEnT*. Finally, the idea of using blockchain technology and smart contracts to enable collaboration in the security domain has been pitched in several interesting recent works [1, 27, 44]. However, to the best of our knowledge, no concrete design and instantiation for network TI sharing has been proposed before. In the start-up world, *PolySwarm* [31] is creating a prediction marketplace for the classification of malware. This is an interesting idea and could be integrated with *TRIDEnT* on a single platform.

Finally, besides the aforesaid (mostly academic) work, a number of practical real-world TI sharing platforms exist. The Malware Information Sharing Platform (MISP) is one such example of a collaborative platform that has been used by several organizations [43]. Initially MISP, as its original name implies, mainly focused on malware exchange; over the years though, it has been practically extended to more indicator types. However, MISP does not tackle the incentives topic, nor the trust and TI quality aspect. In addition, due to its structure (closed system that requires a formal process to enroll) attacks on it (internal or external) have not been extensively investigated. Other similar systems that have been proposed include IBM's X-Force Exchange[17], Facebook's ThreatExchange[18] and DShield[19]. These platforms are in principle closed systems that focus on a one-to-many or many-to-many exchange of low level (e.g., raw) data between entities [24]. In many cases, the diversity of shared data is very limited; for instance, the Facebook ThreatExchange is only dealing with Facebook indicators to assist Facebook API developers. Note that we do not view *TRIDEnT* as a competitor to existing sharing platforms, which have solved a lot of practical problems concerning the structure and mechanics of the actual data transmission and correlation, but rather as an underlying important incentives layer.

[17] https://exchange.xforce.ibmcloud.com
[18] https://developers.facebook.com/programs/threatexchange
[19] https://www.dshield.org

6 DISCUSSION AND CONCLUSIONS

In this section, we discuss how our system can generalize to other types of shared data, as well as what limitations exist.

Generalization In essence, *TRIDEnT* is a generic streaming marketplace, and thus could be used for sharing any kind of information, and not just security TI. However, the importance, numerous challenges, as well as the unique risk incurred by both buying and selling TI, make the functionalities offered by *TRIDEnT* necessary. Despite its limitations (see below), we consider *TRIDEnT* to be a considerable step towards increased security collaboration in practice; not only among big companies and organizations, but open to anyone. Also, the format of *TRIDEnT*'s advertisement and TI is naturally not limited to Snort-style alerts. The system can be used to establish streams where e.g. bloom filters or pre-trained machine learning models are exchanged, allowing greater real-time detection capabilities. The stream establishment functionality could even be employed in order to run privacy-preserving CIDS on raw data using secure multiparty computation (SMC) in the future. The main challenges of incentives, TI quality and trust would still be present to some extent in any case. We consider the field to be important and ripe for research progress after a stagnant period.

Why a decentralized TI marketplace Although financial incentives and reputation systems can be implemented in centralized sharing architectures, there are reasons to pursue a decentralized solution. It may be impossible for all parties interesting in exchanging data to agree on a single trusted arbitrator and platform operator (for example when belonging to different jurisdictions). With a decentralized implementation with smart contracts, no platform operator is needed.

Limitations Considering our system's limitations, an important one is that it has not been tested in the wild, since the financial cost of developing and supporting production-ready software (and more so security-critical software) greatly exceeds our research budget. Additionally, the stream reselling attack (see Section 3.5) may require new mitigation mechanisms.

Conclusions and future work With the design and prototype implementation of *TRIDEnT* we show that fully decentralized threat indicator exchanges built with smart contracts are feasible. Further investigation into the trust establishment mechanisms (e.g. investigating the entry cost vs. security trade-off of different proof-of-burn parameters, or incorporating forms of financial insurance in the trust calculation), as well as empirical validation of the design principles are areas of future research.

REFERENCES

[1] Nikolaos Alexopoulos, Emmanouil Vasilomanolakis, Natália Réka Ivánkó, and Max Mühlhäuser. 2017. Towards Blockchain-Based Collaborative Intrusion Detection Systems. In *Critical Information Infrastructures Security - 12th International Conference, CRITIS 2017, Lucca, Italy, October 8-13, 2017, Revised Selected Papers*. 107–118. https://doi.org/10.1007/978-3-319-99843-5_10

[2] Luca Allodi, Fabio Massacci, and Julian Williams. 2017. The work-averse cyber attacker model: Theory and evidence from two million attack signatures. In *16th Annual Workshop on the Economics of Information Security, WEIS 2017, San Diego, USA, 26-27 June, 2017.*

[3] Ross Anderson and Tyler Moore. 2006. The economics of information security. *Science* 314, 5799 (2006), 610–613.

[4] Ross J. Anderson. 2001. Why Information Security is Hard-An Economic Perspective. In *17th Annual Computer Security Applications Conference (ACSAC 2001), 11-14 December 2001, New Orleans, Louisiana, USA*. 358–365. https://doi.org/10.1109/ACSAC.2001.991552

[5] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J. Alex Halderman, Luca Invernizzi, Michalis Kallitsis, Deepak Kumar, Chaz Lever, Zane Ma, Joshua Mason, Damian Menscher, Chad Seaman, Nick Sullivan, Kurt Thomas, and Yi Zhou. 2017. Understanding the Mirai Botnet. In *26th USENIX Security Symposium, USENIX Security 2017, Vancouver, BC, Canada, August 16-18, 2017*. 1093–1110.

[6] Leon Böck, Emmanouil Vasilomanolakis, Max Mühlhäuser, and Shankar Karuppayah. 2018. Next Generation P2P Botnets: Monitoring Under Adverse Conditions. In *Research in Attacks, Intrusions, and Defenses - 21st International Symposium, RAID 2018, Heraklion, Crete, Greece, September 10-12, 2018, Proceedings*. 511–531.

[7] Samuel Burke. 2016. http://money.cnn.com/2016/10/22/technology/cyberattack-dyn-ddos/index.html

[8] Christian Cachin. 2016. Architecture of the Hyperledger blockchain fabric. In *Workshop on Distributed Cryptocurrencies and Consensus Ledgers*.

[9] Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly detection: A survey. *ACM Comput. Surv.* 41, 3 (2009), 15:1–15:58. https://doi.org/10.1145/1541880.1541882

[10] Frédéric Cuppens and Alexandre Miège. 2002. Alert Correlation in a Cooperative Intrusion Detection Framework. In *2002 IEEE Symposium on Security and Privacy, Berkeley, California, USA, May 12-15, 2002*. 202–215. https://doi.org/10.1109/SECPRI.2002.1004372

[11] Roman Danyliw. 2016. *RFC 7970. The Incident Object Description Exchange Format Version 2.* Technical Report.

[12] Claudiu Duma, Martin Karresand, Nahid Shahmehri, and Germano Caronni. 2006. A Trust-Aware, P2P-Based Overlay for Intrusion Detection. In *International Conference on Database and Expert Systems Applications (DEXA'06)*. IEEE, 692–697.

[13] Jesse M Ehrenfeld. 2017. WannaCry, Cybersecurity and Health Information Technology: A Time to Act. *Journal of Medical Systems* 41, 7 (2017), 104.

[14] Carol Fung, Olga Baysal, Jie Zhang, Issam Aib, and Raouf Boutaba. 2008. Trust management for host-based collaborative intrusion detection. *Managing Large-Scale Service Deployment* 5273 (2008), 109–122.

[15] Carol J Fung, Jie Zhang, Issam Aib, and Raouf Boutaba. 2011. Dirichlet-based trust management for effective collaborative intrusion detection networks. *IEEE Transactions on Network and Service Management* 8, 2 (2011), 79–91.

[16] Esther Gal-Or and Anindya Ghose. 2005. The economic incentives for sharing security information. *Information Systems Research* 16, 2 (2005), 186–208.

[17] Yunchuan Guo, Han Zhang, Lingcui Zhang, Liang Fang, and Fenghua Li. 2018. Incentive Mechanism for Cooperative Intrusion Detection: An Evolutionary Game Approach. In *International Conference on Computational Science*. Springer, 83–97.

[18] Sascha Hauke. 2015. *On the Statistics of Trustworthiness Prediction.* Ph.D. Dissertation. Technische Universität Darmstadt.

[19] Kevin Hoffman, David Zage, and Cristina Nita-Rotaru. 2009. A survey of attack and defense techniques for reputation systems. *ACM Computing Surveys (CSUR)* 42, 1 (2009), 1.

[20] Richeng Jin, Xiaofan He, and Huaiyu Dai. 2017. On the tradeoff between privacy and utility in collaborative intrusion detection systems-a game theoretical approach. In *Proceedings of the Hot Topics in Science of Security: Symposium and Bootcamp*. ACM, 45–51.

[21] Radu Jurca and Boi Faltings. 2003. An incentive compatible reputation mechanism. In *E-Commerce, 2003. CEC 2003. IEEE International Conference on*. IEEE, 285–292.

[22] Kaspersky Lab. 2013. https://securelist.com/red-october-diplomatic-cyber-attacks-investigation/36740/

[23] Ahmed E. Kosba, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou. 2016. Hawk: The Blockchain Model of Cryptography and Privacy-Preserving Smart Contracts. In *IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016*. 839–858. https://doi.org/10.1109/SP.2016.55

[24] Stefan Laube and Rainer Böhme. 2017. Strategic Aspects of Cyber Risk Information Sharing. *ACM Computing Surveys (CSUR)* 50, 5 (2017), 77.

[25] Patrick Lincoln, Phillip A. Porras, and Vitaly Shmatikov. 2004. Privacy-Preserving Sharing and Correlation of Security Alerts. In *Proceedings of the 13th USENIX Security Symposium, August 9-13, 2004, San Diego, CA, USA*. 239–254.

[26] Guozhu Meng, Yang Liu, Jie Zhang, Alexander Pokluda, and Raouf Boutaba. 2015. Collaborative security: A survey and taxonomy. *ACM Computing Surveys (CSUR)* 48, 1 (2015), 1.

[27] Weizhi Meng, Elmar Wolfgang Tischhauser, Qingju Wang, Yu Wang, and Jinguang Han. 2018. When intrusion detection meets blockchain technology: a review. *IEEE Access* 6 (2018), 10179–10188.

[28] Florian Menges and Günther Pernul. 2018. A comparative analysis of incident reporting formats. *Computers & Security* (2018).

[29] OASIS Cyber Threat Intelligence (CTI) TC. 2017. STIX 2. Structured Threat Infromation Expression. https://oasis-open.github.io/cti-documentation/

[30] Vern Paxson. 1999. Bro: a system for detecting network intruders in real-time. *Computer networks* 31, 23-24 (1999), 2435–2463.

[31] Polyswarm. 2018. *A decentralized cyber threat intelligence market.* Technical Report. https://polyswarm.io/polyswarm-whitepaper.pdf

[32] Ponemon Institute. 2018. *Third Annual Study On Exchanging Cyber Threat Intelligence: There Has to Be a Better Way.* Technical Report. Ponemon.

[33] protective 2018. PROTECTIVE: Proactive Risk Management. https://protective-h2020.eu/.

[34] Sebastian Ries. 2007. Certain trust: a trust model for users and agents. In *Proceedings of the 2007 ACM Symposium on Applied Computing (SAC), Seoul, Korea, March 11-15, 2007*. 1599–1604. https://doi.org/10.1145/1244002.1244342

[35] Sebastian Ries, Sheikh Mahbub Habib, Max Mühlhäuser, and Vijay Varadharajan. 2011. CertainLogic: A Logic for Modeling Trust and Uncertainty - (Short Paper). In *Trust and Trustworthy Computing - 4th International Conference, TRUST 2011, Pittsburgh, PA, USA, June 22-24, 2011. Proceedings*. 254–261. https://doi.org/10.1007/978-3-642-21599-5_19

[36] Martin Roesch. 1999. Snort-lightweight intrusion detection for networks. In *USENIX conference on System administration*. 229–238.

[37] Alexander Schaub, Rémi Bazin, Omar Hasan, and Lionel Brunie. 2016. A Trustless Privacy-Preserving Reputation System. In *ICT Systems Security and Privacy Protection - 31st IFIP TC 11 International Conference, SEC 2016, Ghent, Belgium, May 30 - June 1, 2016, Proceedings*. 398–411. https://doi.org/10.1007/978-3-319-33630-5_27

[38] Vitaly Shmatikov and Ming-Hsiu Wang. 2007. Security against probe-response attacks in collaborative intrusion detection. In *Proceedings of the 2007 workshop on Large scale attack defense*. ACM, 129–136.

[39] Deepak K Tosh, Shamik Sengupta, Sankar Mukhopadhyay, Charles A Kamhoua, and Kevin A Kwiat. 2015. Game theoretic modeling to enforce security information sharing among firms. In *Cyber Security and Cloud Computing (CSCloud), 2015 IEEE 2nd International Conference on*. IEEE, 7–12.

[40] Johannes Ullrich. 2000. Dshield Internet Storm Center. https://www.dshield.org/.

[41] Emmanouil Vasilomanolakis, Shankar Karuppayah, Max Mühlhäuser, and Mathias Fischer. 2015. Taxonomy and Survey of Collaborative Intrusion Detection. *Comput. Surveys* 47, 4 (2015), 33. https://doi.org/10.1145/2716260

[42] Emmanouil Vasilomanolakis, Matthias Krügl, Carlos Garcia Cordero, Max Mühlhäuser, and Mathias Fischer. 2015. SkipMon: A Locality-Aware Collaborative Intrusion Detection System. In *Computing and Communications Conference (IPCCC), IEEE 34th International Performance*. IEEE, 1–8.

[43] Cynthia Wagner, Alexandre Dulaunoy, Gérard Wagener, and Andras Iklody. 2016. MISP: The Design and Implementation of a Collaborative Threat Intelligence Sharing Platform. In *Proceedings of the 2016 ACM on Workshop on Information Sharing and Collaborative Security*. ACM, 49–56.

[44] George D Webster, Ryan L Harris, Zachary D Hanif, Bruce A Hembree, Jens Grossklags, and Claudia Eckert. 2018. Sharing is Caring: Collaborative Analysis and Real-time Enquiry for Security Analytics. In *IEEE International Symposium on Recent Advances on Blockchain and Its Applications (BlockchainApp)*.

[45] Gavin Wood. 2014. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper* 151 (2014).

[46] Mark Wood and Michael Erlinger. 2007. *RFC 4766. Intrusion detection message exchange requirements.* Technical Report.

[47] Chenfeng Vincent Zhou, Christopher Leckie, and Shanika Karunasekera. 2010. A Survey of Coordinated Attacks and Collaborative Intrusion Detection. *Computers & Security* 29, 1 (feb 2010), 124–140.

[48] Quanyan Zhu, Carol Fung, Raouf Boutaba, and Tamer Basar. 2012. GUIDEX: A game-theoretic incentive-based mechanism for intrusion detection networks. *IEEE Journal on Selected Areas in Communications* 30, 11 (2012), 2220–2230.

[49] Dionysis Zindros. 2014. A pseudonymous trust system for a decentralized anonymous marketplace. https://gist.github.com/dionyziz/e3b296861175e0ebea4b