DOI: 10.1561/9781680836837.ch14

Chapter 14

Leveraging Interledger Technologies in IoT Security Risk Management

By Dmitrij Lagutin, Yki Kortesniemi, Vasilios A. Siris, Nikos Fotiou, George C. Polyzos and Lei Wu

There are security vulnerabilities in all technological systems but particularly in many Internet of Things (IoT) solutions. *Responsible disclosure* has been the established approach for the security community to deal with the discovered vulnerabilities, but this approach does not fare well in the modern fast-paced world and, in particular, in the low-cost, often highly constrained, long-expected usable lifetime, yet highly volatile IoT space. This chapter proposes a Distributed Ledger Technology (DLT) and interledger-based *Automated Responsible Disclosure* (ARD) solution that provides stronger incentives to the involved parties to address the vulnerabilities in a timely manner, better accountability of all parties, and, as a result, better security for the public at large.

14.1 Introduction

Responsible disclosure is a model to disclose security vulnerabilities in a way that allows the vendor some time to create a fix before the vulnerability is publicly revealed. Responsible disclosure is best known and associated with software products and bugs in software, but it applies to both software and hardware, and more generally any products with security features. Most security experts use this model to report vulnerabilities [1].

This model is of particular interest in the context of the Internet of Things (IoT) for several reasons. First, there have been many significant security vulnerabilities in IoT devices. Not only are many IoT devices relatively unsophisticated and often very limited in capabilities, but they are also often more exposed yet less frequently or even completely unmonitored during their operation for extended periods of time. Furthermore, because of the relatively low price and margin of IoT devices, and often fast progress in the domain, many manufactures are disinterested in addressing vulnerabilities publicised in their products. Many of the products have only short market lifetimes and all compete for very fast time to market, leading to an abundance of security issues in products. Finally, because of all these problems, IoT devices have been exploited and have caused recently highly publicized security problems with significant impact [2, 3], and these problems are unlikely to significantly diminish in the near future.

The idea behind the responsible disclosure is sound, but unfortunately it often does not work that well in practice. In many situations, there are no clear and enforced time limits after which the vulnerability will be disclosed, and the vendors behind vulnerable products often request extensions to the disclosure if they are not able to provide fixes in time. Also, there have been several cases where the security experts have been pressured, threatened, or even sued by vendors to not release the vulnerability within a reasonable time frame [4, 5]. As a recent example from August 2019, a security researcher found a vulnerability in Valve's Steam gaming client, but as the vendor was not interested in fixing the issue, the researcher then wanted to publish the findings, and the vendor promptly banned the researcher from the HackerOne[1] bug bounty platform [6]. Furthermore, due to the lack of transparency to the details of the vulnerability, the vendor may release a fix that does not properly address the vulnerability merely to appear as security conscious. And finally, it is impossible for the customers and the public in general to know how many vulnerabilities in each vendor's products have been identified and how many of them remain unfixed.

Even if most vendors put emphasis on security in their advertising, in reality vendors often do not have clear incentives to create secure products or fix their vulnerabilities quickly, which is apparent from a simple cost-benefit analysis:

1. Vendors are typically companies that aim to maximize their profits, which can be expressed as: revenue-expenses.
2. It is not possible for a vendor to generate significant extra revenue by improving the security of its products, since proving or measuring security of the product is practically impossible (it cannot be practically proved that a product is secure, only that it is insecure).

1. https://www.hackerone.com/

3. Similarly, insecure products will not make the vendor lose a significant amount of revenue, since the current Responsible Disclosure scheme does not reveal the number of vulnerabilities to the general public and allows the vendor ample time to fix its products. Since most of vulnerabilities are released to the general public only after a long period of time, most of customers do not suffer too much or too often from using insecure products, and therefore, they do not have a clear incentive to switch to another vendor.
4. Creating more secure products is expensive (it requires more development time, more experienced developers, better processes, more time for testing, etc.); the vendor will often rather minimise the costs by making less secure products.

Per 2 and 3 above, vendor revenue will not be adversely affected by insecure products, yet per 4, creating less secure products reduces expenses. Therefore, per 1, providing less secure products tends to maximise the vendor's profits.

Now, if the vulnerabilities would be disclosed in more strict and transparent manner, the situation would change. Vendors behind insecure products would lose reputation and would have a harder time selling products. Customers could, for instance, demand agreements from vendors with bad security history that in the case a serious vulnerability is found in the product, the vendor would pay compensation to the customer, etc. As a result, vendors would have strong incentives to improve the security of their products, and overall, there would be financial incentives to provide more secure products and fix vulnerabilities quickly, which in turn would lead to more secure products in the market.

Transparency of vulnerabilities is also important because even if the vulnerability has been found and responsibly disclosed by a one expert, it does not mean that no one else in the world knows about it. In fact, it is very likely that the same vulnerability has already been found by other actors, such as criminal organisations, which do not have any incentives to disclose it to the vendor and commonly trade vulnerabilities on black markets [7]. From the customers' point of view, it is categorically better to know that a vulnerability exists (even if the whole world knows about it), compared to the situation where the vulnerability exists but the customers do not know about it. At least in the former case the customers can take actions to mitigate the risk (stop using the vulnerable product temporarily, install additional safeguards, etc.), while in the latter case the customers cannot predict or mitigate attacks in any way since they are not aware of the possibility of the attack. Therefore, a long time between the vulnerability discovery and the related public disclosure increases the possibility of security breaches and decreases the overall security.

To succeed, any practical implementation of Responsible Disclosure also has to be able to deal with the scale of vulnerabilities, so because of the huge number of IoT devices with diverse capabilities and operations, a fully—or almost

fully—automated set of procedures for recording vulnerabilities and their corresponding patches in a transparent and non-repudiated way is necessary to ensure the security and ultimately the success of IoT deployments.

Distributed Ledger Technologies (DLTs) and interledger mechanisms can be leveraged to immutably record when a vulnerability is disclosed to authorities and/or vendor, and after a certain time period (which should be the same regardless of the vendor in question) the vulnerability is then automatically revealed to the public (regardless of wherever it has been fixed or not), or at least the metadata would be revealed (that vendor X did not fix the vulnerability within a certain timeframe). On a technical level, this would be implemented by putting the vulnerability disclosure, *V*, to a private ledger for security authorities and vendor or vendors, and recording the hash of the vulnerability along with the vendor, product, and time information on a public ledger. As a result, the vendors would have stronger incentives to improve the security of their products.

This chapter is organised as follows: Section 14.2 provides background on DLTs, smart contracts and chaincode, the corresponding functionality in permissioned blockchains, interledger technologies, and Decentralized Identifiers (DIDs). Section 14.3 then briefly presents the history of and the state of the art in vulnerability disclosure approaches. Section 14.4 gives an overview of the proposed Automated Responsible Disclosure (ARD) approach of IoT risk management through interledger-enabled disclosure with accountability. Section 14.5 provides an evaluation of the design, and Section 14.6 presents the conclusions and proposed future work in order to establish, promote, and practically evaluate the approach in the real world.

14.2 Background

The Automated Responsible Disclosure (ARD) solution proposed in this chapter builds on four key technologies: distributed ledger technologies, smart contracts and chaincode, interledger functionality, and Decentralised Identifiers.

14.2.1 Distributed Ledger Technologies (DLTs)

Distributed Ledger Technologies (DLTs), such as blockchains, offer decentralised solutions for collaboration, interoperability, and trust. One of the main features of DLTs is the *immutability* of data: ledgers are append-only databases where existing data cannot be modified and only new data can be added. Another major feature of DLTs is a distributed *consensus mechanism* [8], which controls what and how data are added to the ledger. Finally, DLTs also *replicate* data to participating nodes thus improving availability. Because of these three properties, DLTs avoid single points

of failure and offer resilience against many attacks. It is relatively easy to determine if any of the participating nodes in the DLT are misbehaving and even in an extreme case, where an attacker manages to control the majority of the DLT's resources, the attacker still cannot modify the existing data, only control the addition of new data.

DLTs can be implemented with different levels of openness. They can be fully *open* (permissionless), which means that anyone can join the DLT and propose transactions; most well-known DLTs such as Bitcoin and Ethereum are based on this principle. However, DLTs can also be permissioned, which makes them either *semi-open*, in which case read access is open to everyone but write access is restricted, or *closed*, in which case both read and write access are restricted.

The main practical innovation of DLTs is the enablement of distributed trust. While there have been multiple proposals for distributed databases in the past, they have mostly concentrated on the distributed implementation, while the trust model has remained firmly centralised. In contrast, DLTs allow various entities that may not fully trust each other, such as individuals, organisations, and companies, to collaborate in a safe and transparent manner, with only a low risk of being cheated by others.

14.2.2 Smart Contracts and Chaincode

Smart contracts [9] are another important feature provided by several DLTs: they are distributed applications that are executed on the ledger. Whenever an entity interacts with a smart contract, these operations are executed by all (full) nodes in the DLT network in a deterministic and reliable way; one of these nodes is then selected to store the contract's execution outcome (if any) in the ledger. Smart contracts can verify DLT identities and digital signatures, perform general purpose computations, and invoke other smart contracts. The code of the smart contract is immutable and cannot be modified even by its owner. Moreover, since all transactions sent to a contract are recorded in the DLT, it is possible to obtain all historical values of the contract. Smart contracts typically refer to code running on the Ethereum (in which case they are Turing-complete), but similar functionality is available in other DLTs. In particular, in the permissioned Hyperledger Fabric, similar functionality is called *chaincode*, and simpler, more constrained scripts can also be run on Bitcoin. Smart contracts or similar functionality is critical for automating processes and will be exploited in the techniques described later.

14.2.3 Interledger Functionality

There exists a large number of DLTs each offering different trade-offs in terms of latency, throughput, consensus algorithm, functionality, etc., thus rendering them suitable for different types of applications. For example, a DLT can focus on

cryptocurrency payments, recording of IoT events, or access authorisation. In complex systems, it is therefore often not feasible of efficient to use only a single DLT for everything; hence, the *interledger* approach that allows different DLTs to exchange data with each other is required in many situations. Using multiple ledgers is also beneficial for privacy reasons: participants within a DLT need to be able to access all data stored in that DLT to independently verify its integrity, which encourages the participants to use private ledgers and store only a subset of the data to the main ledger used for collaboration with others. A concrete example of the use of interledger is the following: data are stored in a closed ledger and simultaneously related information (e.g., a hash of the original data) is stored to a semi-open or open ledger. At a later time, the original data are revealed and the hash in the (more) open ledger guarantees that the original data have not been modified.

Multiple ledgers are also necessary for crypto-agility, as cryptographic algorithms used by DLTs (such as SHA-256) will not stay safe forever; thus, it is necessary to have a mechanism to transfer data from one ledger to another. Siris *et al.* [10] present a review of interledger approaches, which differ in their support for transferring and/or trading value between ledgers, whether they support the transfer of information in addition to payments across ledgers, the balance between decentralized trust and cost (which can include both transaction cost and delay), the level of privacy, and their overall scalability and functionality that can facilitate the innovation of the DLT ecosystem. Finally, interledger functionality may also utilize *hash-locks*, which are cryptographic locks that can be unlocked by revealing a secret whose hash matches with the value configured in the hash-lock. By using the same secret value on two or more ledgers, it is possible to achieve cross-ledger atomic operations, where the first transaction reveals the secret, which in turn is used to unlock the transaction on the other ledger(s). If the secret is never revealed, none of the transactions will succeed.

Interledger mechanisms, which in the above example involve the hash of data stored in a private ledger, allow securely linking information and/or transactions stored on multiple ledgers. Such mechanisms can be used to create a dependence relation that allows the execution of smart contracts functions on one ledger only if some data has been recorded or transactions have been performed on another ledger. As discussed in Section 14.4 that describes the ARD approach, the above feature ensures that the time dependency and order of events on the public and private ledgers are automatically and transparently enforced.

14.2.4 Decentralized Identifiers (DIDs)

Currently, an identity technology receiving much attention is the *Decentralized Identifiers (DIDs)*. A key aspect of DIDs is that they are designed not to

be dependent on a central issuing party (Identity Provider or IdP) that creates and controls the identity. Instead, DIDs are managed by the identity owner (or a guardian on the owner's behalf), an approach known as *self-sovereign identity* [11].

There are several different DID technologies in development [12], some of the most prominent being Sovrin, uPort, and Veres One. These technologies started with similar but individual goals in mind, but lately many of them have adopted the approach and format of the W3C DID specification [13] thus rendering them more and more interoperable. The specification defines a DID as a random string, often derived from the public key used with the identity. If a new DID is allocated for every party one operates/communicates with, correlating one's activities with different parties would be significantly harder to achieve. This property can be further enhanced by replacing existing DIDs with new ones at suitable intervals, even after just a single use.

Yet, DIDs alone do not suffice, as some means of distributing the related public keys, any later changes to the keys, or other identity-related information is required. To this end, many of the DID solutions rely on a DLT for public DIDs (used by parties that want to be known publicly), whereas for private DIDs (e.g., used by individuals or their personal IoT devices) an application specific channel is used to distribute the information. Some DID technologies, e.g., Sovrin and Veres One, are launching their own permissioned DLTs, while others rely on existing blockchains (e.g., uPort is built on top of Ethereum). All three example technologies originally intended to use DLTs/blockchains for distributing information about private DIDs, but the emergence of the General Data Protection Regulation (GDPR) [14] in the EU and other similar changes have made storing personally identifiable information on a non-mutable platform such as a DLT/blockchain problematic. For this reason, Sovrin and Veres One have already excluded individuals DIDs from the ledger, and similar treatment may face the DIDs of IoT devices if they reveal personal information about their owner.

In ARD, DIDs are used to protect the privacy of security experts discovering the vulnerabilities to prevent any undue pressure to suppress the findings.

14.3 Previous Work

Disclosure of vulnerabilities has long been an important challenge in software industry, because security vulnerabilities are among the major reasons for security breaches [15], and vulnerabilities have been extensively exploited in many successful attacks [16]. Since IoT devices interact with the real world, there are potentially high risks with severe impacts (in particular with actuation directly leading

to safety concerns). At the same time, the growing number of IoT devices introduced in recent years and the new types of data coming with them have increased the security risks both for the industry [17] and the in consumer IoT domain [18]. As an example, a case study on baby monitors [19] details how many baby monitors ship with static credentials and other vulnerabilities allowing access to the confidential information provided by the devices. Together, the above-mentioned risks in software systems and the flaws in the IoT systems can be highly dangerous and cause significant economic losses [20], which draws considerable discussion on how such risks can be controlled and how related vulnerabilities should be disclosed [21].

Over time, many different approaches have been proposed for the mechanisms of vulnerabilities disclosure. Traditionally, the so-called *full vendor disclosure* has been used; there the vendor has full control over how and when a vulnerability is disclosed, which leaves many vulnerabilities totally unfixed or fixed only after a long delay [22, 23]. In order to solve the problem of full vendor disclosure with public at risk, the *immediate public disclosure* has been proposed, which introduces a direct and strong incentive for the related vendor to fix the released issue as fast as possible [24]. Even though early disclosure of the vulnerabilities can drive faster fixing and uptake of mitigating measures by related parties to reduce the risk, the attackers can also exploit the released vulnerability to attack already before a patch is delivered by the vendor [25]. To address the conflicts between vendor and users, a hybrid disclosure approach, *responsible disclosure*, was introduced: the vendor is allowed some time to develop a patch before the vulnerability is disclosed by the finder [26]. This way, the vendor is given the opportunity to fix the vulnerability in time without the dangers of immediate exposure, but the public can still learn about the existence of the vulnerability in case the vendor fails to deliver a patch.

To coordinate the process between vendors, finders, and users, often a Computer Emergency Response Team/Coordination Center (CERT/CC) is used as a trusted third party. The economics of the vulnerability disclosure process has been studied [27] to shed light on the interaction between participating parties. There was a trend of using market strength to achieve the optimization of social benefit in the vulnerability disclosure process [28], though many [29, 30] criticize that the market solution performs no better than existing solutions.

Different vulnerability disclosure approaches have been compared and evaluated in their efficiency [31], and with the corresponding cost, benefits and risks provided in [32]. The work in [33] offers another angle for the life cycles of software vulnerabilities, which is based on analysis in seven dimensions for more than forty thousand vulnerabilities reported in the past few decades.

14.4 Automated Responsible Disclosure (ARD)

The parties to responsible disclosure, and the proposed *Automated Responsible Disclosure (ARD)* mechanism, can be divided into the following categories, as shown in Figure 14.1.:

- *Vendors* provide software and hardware-based (IoT) solutions to customers.
- *(Consortia of) Authorities* are in charge of managing security vulnerabilities; there may be multiple such authorities in a single country or region.
- *Security Experts* discover new vulnerabilities.
- *General public*, press, etc. who may be customers of the Vendors, or otherwise interested in the security of available services.

To improve responsible disclosure, the ARD solution has been designed with the following goals and assumptions:

1. The Security Experts cannot be intimidated to prevent the disclosure of vulnerabilities.
2. The Authorities can validate the reported vulnerabilities to prevent false reports.

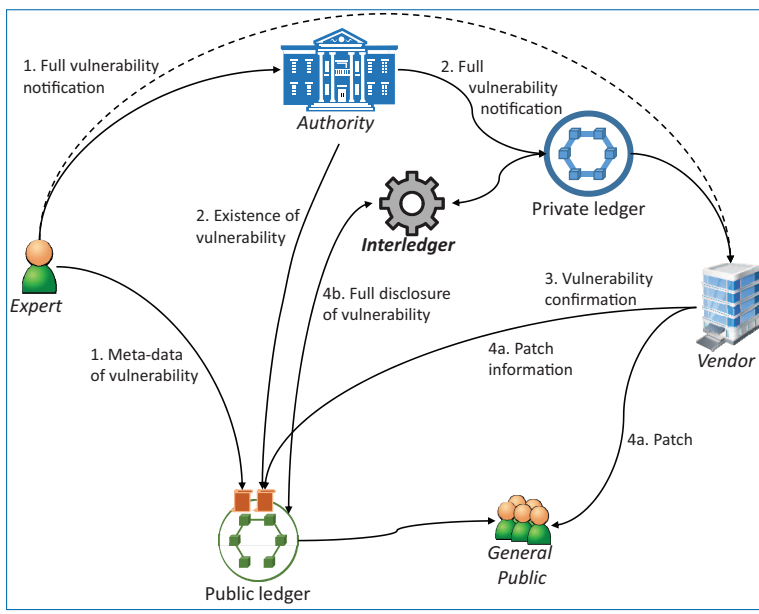

Figure 14.1. Entities of the interledger-enabled ARD solution.

3. The Vendor is clearly notified about each vulnerability and has to agree to fix it within the grace period; otherwise, the vulnerability is immediately disclosed.
4. Releasing a fix results in automatic disclosure of the vulnerability.
5. If a fix has not been released before the grace period has expired, the vulnerability is automatically disclosed.
6. Authorities are trusted to try to operate correctly, e.g., to prevent unintentional leaks of vulnerability information, correctly screen reported vulnerabilities, etc., but disclosure cannot be prevented by the Authority.

The solution is designed to utilize two distributed ledgers, a private one maintained by the Authority and used for storing the details of the vulnerability during the disclosure process, and a public one for first disclosing the existence of a vulnerability and later the details of the vulnerability. A key element is then the interledger functionality, which facilitates the automatic disclosure of the information from the private ledger to the public one once the conditions for disclosure have been met. In the basic case, the interledger functionality is run by the Authority or some other trusted party, but for additional security and resilience, the interledger functionality can be implemented in a distributed manner over a consortium of parties.

Technically, the public and private ledgers can utilize any suitable ledger technology (permissioned for the latter, and permissionless or permissioned for the former). The public ledger is readable by all, while some of the write operations are restricted. Anyone has a permission to store metadata of the vulnerability information (including its hash), which can be achieved, e.g., by a smart contract owned by the Authority. Additionally, write operations related to confirmation of vulnerability, notification of patching the vulnerability, and full disclosure of vulnerability information are limited to the Authority, Vendors, interledger, and the Security Expert who reported the vulnerability.

The private ledger can be implemented, for example, using Hyperledger Fabric, a permissioned DLTs, and a separate *chaincode* (which corresponds to a smart contract in Fabric) per Vendor. In Fabric, the vulnerability information is stored as a *private data collection* and the chaincode controls the access to the vulnerability using a hash-lock (see Siris *et al.* [10]). The hash-lock's secret is generated by the Security Expert and also known by both the Authority and Vendor and is revealed as necessary to the interledger, after which the interledger functionality can retrieve the vulnerability information from the private ledger and write it to the public ledger. The same secret is then used to lock the functionality Vendor is using for notifying of fixing the vulnerability, so Vendor's notification would reveal the secret, which would then allow the interledger functionality to retrieve and publish the full vulnerability information.

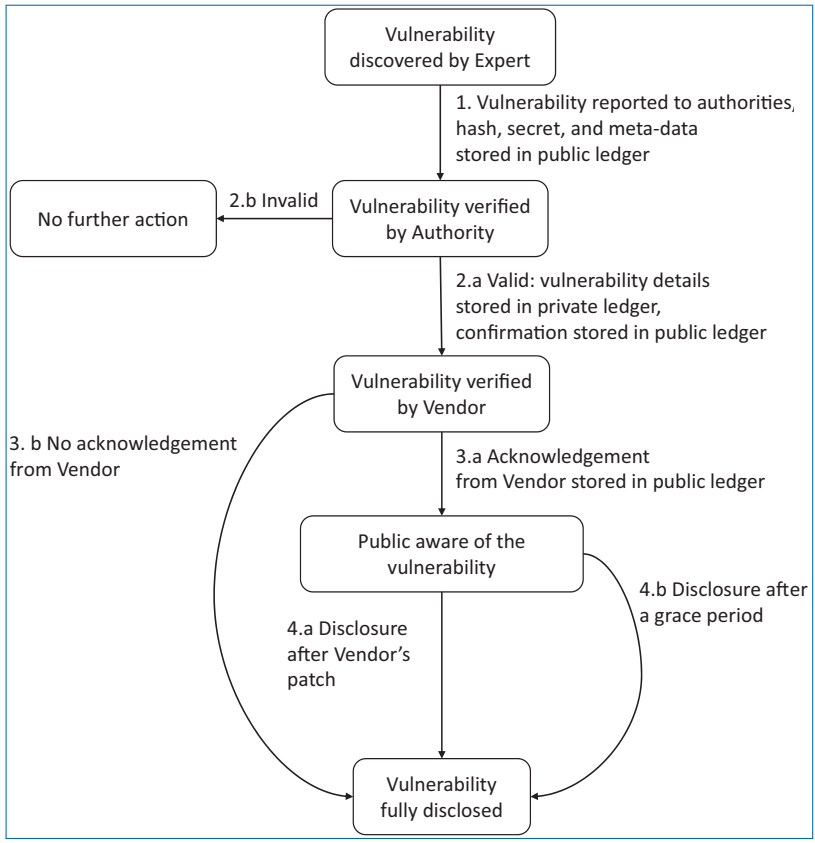

Figure 14.2. State diagram of the ADR mechanism.

If there is a bounty mechanism for discovering vulnerabilities, the Security Experts can be identified using ephemeral decentralized identifiers (DIDs) or similar anonymous identifiers thus allowing them to claim the bounty while preventing the vendor from intimidating the Experts. The other parties in the solution can utilize longer term public identifiers including public DIDs.

ARD functions as follows, as described in Figure 14.2.:

1. The Security Expert, who has discovered a vulnerability V, generates the associated secret s and reports the vulnerability (together with the secret s) with a sufficient detail to the Authority, and optionally to the Vendor. The Security Expert also calculates the hash over the vulnerability information and stores this hash, $H(V)$, along with hash over the secret, $H(s)$, and other relevant metadata (Vendor's name, product name, affected version, etc.) to the public ledger, using a previously published mechanism, e.g., a smart contract.
2. The Authority, once they have ascertained the report is not bogus, stores the details of the vulnerability in the private ledger. The interledger functionality

then commits a transaction to public ledger stating that the Authority confirms that the previously reported vulnerability (with hash $H(V)$) is valid. If the report is a duplicate of a previous report, this is noted in the confirmation and the disclosure will follow the timeline of the original report.

3. The Vendor is notified of the new vulnerability. They now have a fixed time period (e.g., two weeks) to fetch the details of the vulnerability from the private ledger and either:
   a. confirm and agree to fix the vulnerability by storing their agreement to the public ledger, in which case they are accorded the grace period of, e.g., six months,[2] or
   b. deny/ignore the vulnerability and do nothing, in which case the Authority will reveal the secret s[3] to the public ledger and the interledger functionality will automatically disclose the vulnerability from the private ledger.
4. If the vendor has agreed to fix the vulnerability, and then
   a. releases a fix and stores a corresponding notice on the public ledger to indicate the vulnerability is now fixed, after which the interledger functionality will publish the full vulnerability information, which in turn will mark the vulnerability resolved. If the vendor wants its customers to have some extra time to update their systems, it can first indicate the fix is upcoming, and then after some time period marks the vulnerability as fixed. If, however the grace period runs out with no fix,
   b. the interledger functionality automatically discloses the vulnerability as above.
5. After the vulnerability has been disclosed, the security expert can claim the vulnerability report for credit and any bounty offered.

In Steps 1 and 2, the vulnerability report has to identify the Vendor (e.g., with their DID) and the product (e.g., with a vendor-specific identifier provided by the vendor). The same vendor and product ids are then included in the notice on the public ledger, which allows general public to know the number of known but not yet fixed vulnerabilities per vendor and per product.

In Step 3, if the vendor acknowledges the vulnerability, the vendor signs a proof of delivery (with their private key, tied to the public DID) containing the timestamp, the hash of vulnerability disclosure *H(V)*, and the relevant metadata (affected product name and version, etc.). This acknowledgement is stored in the publicly accessible ledger.

2. The Vendor can also mark the vulnerability as a duplicate of the previously reported one, in which case it will be disclosed simultaneously with the original vulnerability, as mentioned above.

3. If the Authority does not reveal the secret s in a timely manner, the Security Expert can also do the same.

Therefore, in the solution the vulnerability is always automatically disclosed by the interledger functionality in one of the three possible moments depending on the actions of the vendor:

- Immediately (if the vendor does not acknowledge it)
- After a fix has been released
- After the grace period if no fix has been released

The periods for acknowledging the vulnerability, after a fix has been released, and the grace period can be customized by the authority based on country, necessary security level, etc. For instance, for critical products, the period can be shorter.

Similarly, the bounty system can be customized: it can, e.g., contain a fixed compensation by the authority and an additional compensation by the vendor or by third parties for select products.

14.5 Analysis

The solution meets the five requirements listed in the beginning of Section 14.3:

1. *The security experts cannot be intimidated to prevent disclosure.*
   All vulnerabilities are reported under ephemeral anonymous identifiers thus maintaining the anonymity of the security expert throughout the process and preventing any intimidation.
2. *The authorities can validate the reported vulnerabilities to prevent false reports.*
   The authority vets all vulnerabilities before they are accepted to a suitable level to weed out bogus reports.
3. *The vendor is clearly notified about each vulnerability and has to agree to fix it to gain the grace period; otherwise, the vulnerability is immediately released.*
   The existence of a new vulnerability related to a product of the Vendor is clearly published on the public ledger to provide the vendor sufficient time to agree to or deny the vulnerability. If the Vendor does not agree to fix the vulnerability, the Authority publishes the secret s on the public ledger after which the vulnerability is automatically disclosed.
4. *Releasing a fix results in an automatic disclosure of the vulnerability.*
   Publishing the fix on the public ledger triggers the automatic disclosure after a short wait period.
5. *If a fix has not been released before the grace period has expired, the vulnerability is automatically disclosed.*
   The Authority (or the Security Expert) will reveal the secret s after the grace period has run out, after which the interledger functionality will automatically disclose the vulnerability.

To further evaluate the solution, the potential dishonesty of each party has to be considered.

If the Security Expert reports a non-valid vulnerability. There is the potential risk that the system is flooded with irrelevant information, e.g., to make the product appear to contain more vulnerabilities than it actually has. To prevent that, the Authority vets all reports, which can help prevent the obvious bogus reports, but non-trivial fakes may pass depending on the level of scrutiny. However, in this respect, the solution is not worse than existing solutions.

Duplicate reports of the same vulnerability. Valid but duplicate reports (either intentional or unintentional) of the same vulnerability can make the product appear of lower quality due to the increased number of reported vulnerabilities. To avoid this, duplicate reports can be marked by either the Authority or Vendor as duplicates of a previous report. Thus, they will not affect the number of individual vulnerabilities discovered, but each report will be eventually disclosed using the same timeline as the original report.

The Authority does not accept valid reports. This is always a potential issue as one can either have a mechanism to vet the reports to prevent spamming or have guaranteed acceptance of reports with a related spam issue. To reduce the risk of valid reports not being accepted, in addition to reporting the vulnerability to the Authority and optionally to the Vendor, the Security Expert stores the hash of vulnerability with associated metadata in the public ledger. If the Authority has not accepted a valid report, the Security Expert can then publicly reveal the vulnerability,[4] which together with the previously stored hash will make it obvious that the Authority did not handle the reported vulnerability properly.

The Authority does not disclose the vulnerability after the grace period has expired. In this case, both the Security Expert and general public will know that the vulnerability information has not been disclosed properly by the Authority, and the disclosure process can also be triggered by the Security Expert, who also possesses the secret needed.

The Vendor denies the existence of a valid vulnerability. In that case, the vulnerability is automatically disclosed thus providing a clear incentive to remain truthful or risk lose customers' trust.

The Vendor claims the fix is effective when it's not. Releasing a fix triggers a disclosure, which would reveal the fix is not effective thus providing a clear incentive to remain truthful.

4. This can be performed either using the same smart contract used for reporting vulnerability metadata, or any other channel.

The Vendor tries to extend the grace period. All time periods are automatically enforced preventing any extensions.

Furthermore, it is important to note that the actions performed by all parties, including the actions of the entity providing the interledger functionality, are transparently and immutably recorded on the public ledger. Hence, the transparency is increased and any attempts for misbehavior can be identified in a non-repudiable manner. The general public will also know the number of found, confirmed, but unpatched vulnerabilities per Vendor and product. This provides incentives for Vendors to patch vulnerabilities quickly and create more secure products.

14.6 Conclusions and Future Work

This chapter has introduced the Automated Responsible Disclosure (ARD) approach to automate the process of vulnerability disclosure for products and systems, designed a mechanism to implement it, and presented an initial analysis confirming the achievement of the design. This approach is motivated by, and addresses in particular, IoT systems risk management.

The solution is an interledger-enabled automated responsible disclosure providing accountability and appropriate incentives to the involved parties. The solution relies on DLTs, smart contracts and chaincode, interledger technologies, and decentralized identifiers, and extends the state of the art in responsible disclosure. The analysis of the design shows that the goals set are achieved automatically, even with external forces trying to disrupt the normal operations. Immutable recordings of actions of the involved parties on a public blockchain provide additional assurances (and incentives for the involved parties) in the case the design assumptions do not hold (or in case of various failures).

The designed mechanism does not include any mechanisms to automatically compensate the expert(s) who found and reported the vulnerability. However, DLTs and interledger mechanisms are very appropriate for providing this functionality as well. It was omitted from this discussion in order to simplify the design and avoid obscuring the key functionality and properties of the solution. This would be an expected extension after the approach has been accepted in the community, but it could also be an added feature and advantage for the approach to become accepted.

Further work is required in order to establish, promote, and practically evaluate the approach in the real world. Relevant authorities, responsible for public security and safety should probably push for the establishment of such a solution and the ecosystem around it.

Acknowledgments

The research reported here has been undertaken in the context of SOFIE (Secure Open Federation for Internet Everywhere) project, which has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No. 779984.

References

[1] J. Trull. Responsible Disclosure: Cyber Security Ethics. CSO Online, 2015. Available at: https://www.csoonline.com/article/2889357/responsible-disclosure-cyber-security-ethics.html (Accessed 5.2.2020).

[2] L. O'Donnell. 2 Million IoT Devices Vulnerable to Complete Takeover. Threatpost, 2019. Available at: https://threatpost.com/iot-devices-vulnerable-takeover/144167/ (Accessed 17.2.2020).

[3] D. Voolf, S. Boddy, and R. Cohen. Gafgyt Targeting Huawei and Asus Routers and Killing Off Rival IoT Botnets. F5 Labs, 2019. Available at: https://www.f5.com/labs/articles/threat-intelligence/gafgyt-targeting-huawei-and-asus-routers-and-killing-off-rival-iot-botnets (Accessed 17.2.2020).

[4] T. Spring. The Vulnerability Disclosure Process: Still Broken. Threatpost, 2018. Available at: https://threatpost.com/the-vulnerability-disclosure-process-still-broken/137180/ (Accessed 5.2.2020).

[5] K. Zetter. A Bizarre Twist in the Debate Over Vulnerability Disclosures. Wired, 2015. https://www.wired.com/2015/09/fireeye-enrw-injunction-bizarre-twist-in-the-debate-over-vulnerability-disclosures/ (Accessed 5.2.2020).

[6] C. Cimpanu. Researcher publishes second Steam zero day after getting banned on Valve's bug bounty program. ZDNet, 2019. https://www.zdnet.com/article/researcher-publishes-second-steam-zero-day-after-getting-banned-on-valves-bug-bounty-program/ (Accessed 5.2.2020).

[7] A. M. Algarni and Y. K. Malaiya. Software Vulnerability Markets: Discoverers and Buyers. Journal of Computer, Information Science and Engineering **8**, no. 3 (2014), 71–81.

[8] N. Stifter, A. Judmayer, P. Schindler, A. Zamyatin, and E. Weippl. Agreement with Satoshi – On the Formalization of Nakamoto Consensus. Cryptology ePrint Archive, Report 2018/400, 2018.

[9] N. Fotiou and G. C. Polyzos. Smart Contracts for the Internet of Things: Opportunities and Challenges. European Conference on Networks and Communications (EuCNC), Ljubljana, Slovenia, June 2018.

[10] V. A. Siris, P. Nikander, S. Voulgaris, N. Fotiou, D. Lagutin, and G. C. Polyzos. Interledger Approaches. IEEE Access **7** (2019), 89948–89966.
[11] C. Allen. The Path to Self-Sovereign Identity. April 2016. Available at: http://www.lifewithalacrity.com/2016/04/the-path-to-self-soverereignidentity.html (Accessed 18.12.2018).
[12] Blockchain and Identity: Projects/companies working on blockchain and identity. Available at: https://github.com/peacekeeper/blockchainidentity (Accessed 7.11.2018).
[13] D. Reed *et al.* Decentralized Identifiers (DIDs) v1.00 – Core Data Model and Syntaxes. W3C Working Draft 09 December 2019. Available at: https://www.w3.org/TR/did-core/ (Accessed 12.2.2020).
[14] Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation).
[15] H. Cavusoglu, H. Cavusoglu, and S. Raghunathan. Emerging Issues in Responsible Vulnerability Disclosure. WEIS. 2005.
[16] A. Arora, R. Telang, and H. Xu. Optimal policy for software vulnerability disclosure. Management Science **54**, no. 4, (2008), 642–656.
[17] L. J. Trautman and P. C. Ormerod. Industrial cyber vulnerabilities: Lessons from Stuxnet and the Internet of Things. U. Miami L. Rev. 72 (2017), 761.
[18] A. Nakajima, *et al.* A Pilot Study on Consumer IoT Device Vulnerability Disclosure and Patch Release in Japan and the United States. Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security. 2019.
[19] M. Stanislav and T. Beardsley. Hacking iot: A case study on baby monitor exposures and vulnerabilities. Rapid7 Report (2015).
[20] A. Arora and R. Telang. Economics of software vulnerability disclosure. IEEE Security & Privacy **3**, no. 1, (2005), 20–25.
[21] H. Cavusoglu and S. Raghunathan. Efficiency of Vulnerability Disclosure Mechanisms to Disseminate Vulnerability Knowledge. IEEE Transactions on Software Engineering **33**, no. 3, (2017), 171–185.
[22] J. Schiller. Responsible vulnerability disclosure: a hard problem. Secure Business Quarterly **2**, no. 1–5 (2002).
[23] M. McQueen, J. L. Wright, and L. Wellman. Are Vulnerability Disclosure Deadlines Justified? Third International Workshop on Security Measurements and Metrics, Banff, AB, (2011) 96–101.
[24] W. Pond. Do security holes demand full disclosure?. eWeek (2000).

[25] S. Ransbotham and S. Mitra. The impact of immediate disclosure on attack diffusion and volume. Economics of Information Security and Privacy III, Springer (2013), 1–12.
[26] A. Stone. Software flaws, to tell or not to tell?. IEEE Software **20**, no. 1, (2003), 70–73.
[27] J. Bollinger. "Economies of disclosure." ACM SIGCAS Computers and Society **34**, no. 3, (2004): 1–1.
[28] A. Ozment. Bug auctions: Vulnerability markets reconsidered. Third Workshop on the Economics of Information Security (2004).
[29] K. Kannan and Rahul Telang. Market for software vulnerabilities? Think again. Management Science **51**, no. 5, (2005), 726–740.
[30] S. Ransbotham, Sam, S. Mitra, and J. Ramsey. Are markets for vulnerabilities effective? Mis Quarterly (2012), 43–64.
[31] H. Cavusoglu, H. Cavusoglu and S. Raghunathan. Efficiency of Vulnerability Disclosure Mechanisms to Disseminate Vulnerability Knowledge. IEEE Transactions on Software Engineering, **33**, no. 3, (2007), 171–185.
[32] A. Cencini, K. Yu, and T. Chan. Software vulnerabilities: full-, responsible-, and non-disclosure. (2005). Available from: https://courses.cs.washington.edu/courses/csep590/05au/whitepaper_turnin/software_vulnerabilities_by_cencini_yu_chan.pdf (Accessed 17.2.2020).
[33] M. Shahzad, M. Z. Shafiq, and A. X. Liu. A large scale exploratory analysis of software vulnerability life cycles. Proceedings of the 34th International Conference on Software Engineering (ICSE), Zurich, (2012), 771–781.