SHIELD: Assessing Security-by-Design in Federated Data Spaces Using Attack Graphs

Alessandro Palma[1], Nikolaos Papadakis[2], Georgios Bouloukakis[2], Joaquin Garcia-Alfaro[2], Mattia Sospetti[3], Kostas Magoutis[4,5]
palma@diag.uniroma1.it,{nikolaos.papadakis,georgios.bouloukakis,joaquin.garcia_alfaro}@telecom-sudparis.eu,mattia.sospetti@grottinilab.com,magoutis@ics.forth.gr
[1]Sapienza University of Rome, Department of Computer, Control, and Management Engineering (DIAG), 00185, Italy
[2]Samovar, Télécom SudParis, Institut Polytechnique de Paris, 91120 Palaiseau, France
[3]GrottiniLab S.r.L., Italy
[4]Computer Science Department, University of Crete, Greece
[5]Institute of Computer Science (ICS), Foundation for Research and Technology - Hellas (FORTH), Greece

Abstract

Federated data spaces allow organizations to share and control their own data across various domains, but their exposure to cyber attacks has increased due to a surge in newly discovered vulnerabilities. Existing solutions to secure them focus on messaging protocol protection (e.g., using cryptographic means), but this is not sufficient. Attackers may exploit additional vulnerabilities to cause significant issues (e.g., disrupting the availability of services). To this end, we propose SHIELD, a security-by-design approach for federated data spaces, which leverages attack graphs and trust computation to mitigate the risks of cyber attacks. Mitigation is accomplished by proactively assessing the data spaces' weaknesses and implementing security messaging measures to prevent detrimental attacks. A prototype implementation of SHIELD using publish/subscribe as a messaging mechanism is experimentally evaluated over a real architecture in a V2X (Vehicle-to-Everything) scenario.

CCS Concepts

• **Computer systems organization → Distributed architectures**; • **Networks → Network security**; • **Security and privacy**;

Keywords

Federated Data Spaces, Security By Design, Attack Graph, Trust Management

ACM Reference Format:
Alessandro Palma[1], Nikolaos Papadakis[2], Georgios Bouloukakis[2], Joaquin Garcia-Alfaro[2]„ Mattia Sospetti[3], Kostas Magoutis[4,5]. 2025. SHIELD: Assessing Security-by-Design in Federated Data Spaces Using Attack Graphs. In *The 40th ACM/SIGAPP Symposium on Applied Computing (SAC '25), March 31-April 4, 2025, Catania, Italy*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3672608.3707797

1 Introduction

In today's digital world, the Internet of Things (IoT) is expanding to provide services in interconnected ecosystems of devices, such as those in agriculture, vehicles, and smart cities. This widespread adoption of IoT devices has led to the development of *federated IoT data spaces*, which are collaborative environments where multiple organizations share and access data across different domains while maintaining control over their own information [34]. In federated IoT data spaces, data is primarily collected via IoT devices, with servers and other intermediary components (e.g., brokers) facilitating the aggregation and processing of this data. Such spaces are essential in cultivating *smart communities* that leverage data for enhanced service provision, improved efficiency, and refined decision-making processes. A smart community refers to an urban area that uses digital technologies to enhance the quality of life, improve sustainability, and streamline municipal services [33]. Smart communities predominantly use IoT devices to exchange data between software components through messages. Messages act as carriers of information, ranging from simple status updates to complex datasets. An example of a messaging mechanism commonly employed in federated data spaces is publish/subscribe (pub/-sub) [10] where publishers send messages to a topic and subscribers receive messages from topics they are subscribed to.

Given the importance of the messages exchanged in federated data spaces and their exposure to cyber attacks [40], much effort has been put into securing messaging between devices (e.g., through encryption schemes [23]). While these approaches greatly benefit privacy in federated data spaces, they overlook the presence of additional vulnerabilities due to misconfiguration, bugs in the source code, or resource limitations [13]. An attacker can exploit them to intrude and compromise the data spaces through *multi-step attacks*, where an attacker performs different intrusion steps to damage some target devices [31]. They are particularly relevant in federated data spaces, where an attacker may want to compromise a smart community [33, 41], by exploiting known vulnerabilities (e.g., discovered through vulnerability scanners) bypassing the security of the messaging protocol. Thus, data spaces are exposed to cyber attacks even by employing existing approaches [32].

Security-by-design is the paradigm that prioritizes security at the early stages of system development and its provision in federated data spaces is valuable, but far from trivial. The first problem is that vulnerabilities in the devices can rarely be patched [40], because of several constraints. For example, updating software and firmware components to patch vulnerabilities may impact the Quality of Service (QoS) for the shutdown time necessary for the update or

SAC '25, March 31-April 4, 2025, Catania, Italy

ACM ISBN 979-8-4007-0629-5/25/03
https://doi.org/10.1145/3672608.3707797

the required resources (e.g., additional memory). Moreover, many services are managed by third-party companies, thus hindering the possibility of an external user patching vulnerabilities. Finally, the long lifespan of IoT devices implies that some of them may be old, preventing software upgrades or security patches [40].

Thus, alternatives to vulnerability patches must be found. A common approach to address this challenge is leveraging trust management to block untrusted messages [45]. However, this raises a second problem related to the difficulty of determining trust values by integrating security-by-design metrics with information about federated data spaces (e.g., smart communities). Determining quantitative measures for trust is challenging due to the heterogeneous metrics to take into account and the interconnection among devices [5]. Existing trust management approaches focus mainly on encryption [46], overlooking the combination of data space features (e.g., federated architecture) with security-by-design.

A final emerging problem is the difficulty of securing messaging by abstracting from the specific protocol (e.g., MQTT). This difficulty is due to the different types of attacks, which makes it hard to define a general defense mechanism given the diversity of technical challenges of attacks to federated data spaces [20]. In fact, securing messaging protocols makes researchers and practitioners focus on specific attacks (e.g., Denial of Service, spoofing, time-based attacks) [32]. In contrast, multi-step attacks consider an ensemble of attack strategies at design time, requiring the management of complex threat models. Security-by-design approaches exist to handle this problem by modeling the cyber threats in the data space [6], which require human intervention for their mitigation [16], thus necessitating a long defense time. However, protecting data spaces, and cyber-physical systems in general, without human intervention is still a challenge. The difficulty resides in the fact that **there is no standard solution to reduce exposure to cyber attacks and circumvent vulnerability patches at the same time**. This brings security experts to **identify and assess cyber risks** as well as **define a plan of vulnerabilities to patch**, which is time-consuming and difficult to apply to resource-constrained devices in federated data spaces. Some works in the literature found a promising solution in secure messaging, but they focus on specific protocols [4] or attacks [43], thus lacking the security-by-design principles.

To address these problems, we contribute SHIELD, a security-by-design approach for federated data spaces. It leverages security exposure, commonly collected using state-of-the-art tools such as vulnerability scanners to model the threat through *Attack Graph*. It assesses the possible attacks on the federated data space and we incorporate it into a novel trust computation model. Finally, we design a security messaging mechanism that reduces the attack surface of high-risk communication, thereby enhancing security in federated data spaces. The key contributions of this work can be summarized as follows:

(1) SHIELD, a general approach for security-by-design in federated data spaces leveraging Attack Graph (Sec. 3).
(2) A trust computation model that considers Attack Graph metrics to determine trust in federated data spaces (Sec. 3.3).
(3) A security messaging mechanism to face security constraints of devices in federated data spaces (Sec. 3.4).
(4) A working prototype of SHIELD using publish/subscribe and its application and evaluation to a real V2X scenario (Sec. 4).

Section 2 provides a motivating example and system overview, and Sections 5 and 6 report related work and discussion.

2 Proposed Approach

This section presents a motivating example for the addressed problem and the system overview.

2.1 Motivating Example

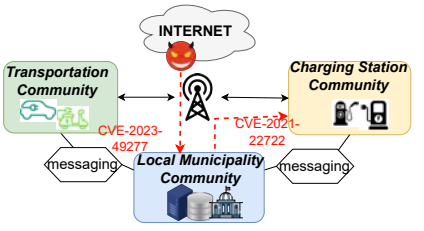


Figure 1: Attack on the V2X motivating scenario.

Vulnerability	Description	V2X device
CVE-2023-49277	A remote user can execute arbitrary code, leading to unauthorized access	Web server (Local Municipality Community)
CVE-2021-22722	Could cause code injection when changing station parameters.	EvLink station (Charging Station Community)

Table 1: Vulnerabilities in the V2X scenario.

Let us consider the federated data space for the V2X (Vehicle-to-Everything) scenario of Fig. 1. It consists of the monitoring and analysis of electric vehicle data to manage power consumption of charging stations [27]. It comprises three smart communities: (i) The transportation community monitors sensors from electric vehicles and sends messages about battery usage to the local municipality; (ii) The local municipality community, acting as the central management entity, comprises storage and management platforms that collect and analyze data to optimize the usage of charging stations; (iii) The charging station community collects real-time data on charging stations, which are monitored by the local municipality.

This scenario reflects the core principles of federated data spaces: smart communities are independent entities and control their own data. However, they must collaborate to enable efficient management of vehicle charging infrastructure. Another example of a federated data space is a smart city port [33], which manages real-time vehicle and ship positions, air quality, and station occupancy. While each community retains control over its data, they selectively share relevant information to enhance transportation operations and respond to emergencies. In such environments attackers do not just exploit a single IoT node, rather they take advantage of the federated environment to propagate multi-step attacks across multiple stakeholders. A proper example is an attacker who injects malicious code to occupy all CPU resources of charging stations and denies their services. Among the others, Denial of Service and malicious code injection attacks are very common in these environments [13]. This is because every device comes with a set of known vulnerabilities, which are collected by the National Institute of Standards and Technology (NIST) through the Common Vulnerabilities and Exposures knowledge base (CVE)[1]. Although security

[1]https://cve.mitre.org/

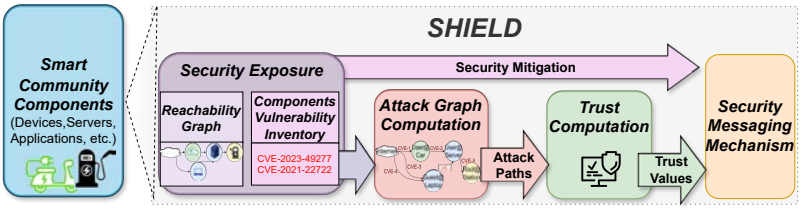


Figure 2: Overview of SHIELD.

mitigation exists to patch these vulnerabilities, they cannot always be applied. Let us consider the vulnerabilities reported in Table 1 for the presented V2X scenario. A remote attacker can compromise the charging station through a multi-step attack (red dashed arrows in Fig. 1). First, the attacker exploits CVE-2023-49277 to get unauthorized access to the server in the local municipality. Then, s/he exploits CVE-2021-22722 in the EVLink station to deny its services.

Two considerations come from this example. The first one is that the attack is independent of the messaging protocol of the data space. While a great part of the literature focuses on securing such protocols (e.g., MQTT) [24], this is not sufficient for securing against cyber attacks in general, and multi-step attacks in particular. For example, encrypting the messages does not prevent the success of the described multi-step attack because the vulnerabilities can be exploited independently of the exchanged messages. Another consideration is that patching the above vulnerabilities is not practical in real-world scenarios [40]. For example, patching the vulnerability in the charging station requires updating its firmware which may be unfeasible given the impact on the Quality of Service (QoS), e.g., different power requirements or long shutdown time for the update. Similarly, mitigating the vulnerability in the server cannot always be done directly because different third-party companies might manage them.

This highlights the presence of vulnerabilities in federated data systems, emphasizing the need for advanced mechanisms to mitigate these risks during the design phase and prevent potential attacks. Existing works on security-by-design for (federated) data spaces mainly focus on the modeling perspective to evaluate the cyber risks [6]. Still, they require human intervention to mitigate vulnerabilities, thus introducing a bottleneck for their defense. To support this bottleneck, other works pay attention to the security of messaging focusing on their protocols [4] or specific attacks [43]. Nonetheless, there is a gap between federated data space messaging and security-by-design. It is less explored and challenging due to the complexity and heterogeneity of both data spaces and their cyber attacks. We fill this gap with SHIELD, whose overview is in the next section.

2.2 Overview of SHIELD

Fig. 2 presents the high-level architecture of SHIELD. The first step of the approach is collecting **Security Exposure** information of the federated data space. It consists of the *Reachability Graph* and the *Vulnerability Inventory*. The former indicates the communication links of the devices in the federated environment and represents the possible routes to access the devices remotely. It is updated when changes in the network occur (e.g., a new device in the data space) by recalculating only the portion of the network affected by the change. Thus, SHIELD ensures that the reachability graph remains up-to-date without excessive computational overhead. The latter is the collection of vulnerabilities in each device, which an attacker may exploit to compromise the data spaces. This information is typically retrievable by running vulnerability scanners (e.g., Nessus and OpenVAS[2]), which are automated tools that identify known vulnerabilities according to the CVE knowledge base. Although vulnerability scanners are run periodically (e.g., every 15/30 days), they do not impact the data spaces' QoS because they are orthogonal to the service provision.

SHIELD uses security exposure information for **Attack Graph Computation**. Attack Graph (AG) is a graph-based representation of the possible ways an attacker can intrude and compromise the federated data space. Nodes in the AG represent attacker states, while the edges represent the vulnerability exploits. Hence, a path into the AG represents a multi-step attack, i.e., a sequence of exploits in different devices that enable the attacker to reach a target. We refer to the paths of the AG as *Attack Paths* and leverage the distributed architecture of federated data spaces to make each device compute the attacks targeting it. An example of an attack path is represented by the red dashed arrows in Fig. 1. According to state-of-the-art risk models [15], we can measure security-by-design metrics (e.g., cyber risk) for each attack path.

We use such security metrics for the **Trust Computation** of each device to evaluate its trustworthiness with the other devices as potential sources of cyber attacks. To this aim, we introduce a novel trust model driven by the cyber risk of attack paths, where the general idea is that messaging with devices highly exposed to cyber attacks should be limited to avoid unreliable data exchange.

Based on trust levels, we design a **Security Messaging Mechanism** with two main objectives, assuming safe trust computation and no attacks on the trust and messaging models: (i) temporarily block high-risk data exchanges that are considered untrusted, and (ii) update the security exposure when a vulnerability is patched. For example, an untrusted device may become trusted after patching a vulnerability, thus it depends on both trust values and security mitigation.

[2]https://www.tenable.com/products/nessus
https://www.openvas.org/

3 System Design and Implementation

3.1 Security Exposure

The inputs of SHIELD are the reachability graph and the vulnerability inventory of each device, which determine roughly how a data space is exposed to cyber attacks (i.e., its security exposure). The former is a directed graph $RG = (D, E)$ where nodes are devices and edges are reachability conditions between them, considering firewall and routing rules. Thus, an edge $e(d_s, d_t) \in E$ indicates that a source device d_s can communicate with a target device d_t through a direct link.

The vulnerability inventory $VI = \{d_1(u_1, u_2, \cdots), \cdots\}$ is the set of vulnerabilities $\{u_1, u_2, \cdots\}$ present in each device d_i. Automatic tools exist to provide this information (e.g., Nessus), that look for system misconfiguration and collect vulnerabilities from the Common Vulnerability Exposure (CVE) knowledge base, which provides details and security metrics of each vulnerability. Typically, network administrators can gain such knowledge through automated tools [6].

However, we need a complete view of the possible exploits that an attacker can perform in federated data spaces through multi-step attacks. Since multi-step attacks are frequent in federated data spaces [41], we employ attack graphs to model such possible threats.

3.2 Attack Graph Computation

An Attack Graph (AG) is a graph-based representation of the possible paths an attacker can exploit to intrude and compromise data spaces. Taking into input the reachability graph and vulnerability inventory, AG represents all the dependencies between devices and vulnerabilities, resulting in multi-step attacks.

Specifically, AG nodes are *security conditions* and represent the attacker access privileges in a specific device. These privileges are commonly considered as *guest*, *user*, and *root* [19]. The edges are *exploit dependencies* and represent the possible movement of the attacker in case of a successful vulnerability exploit. More formally, an Attack Graph $AG = (V, E)$ is a directed multi-graph where V is the set of security condition nodes and E is the set of labeled edges, where an edge $e = (v_1, v_2, u) \in E$ indicates the attacker can move from condition v_1 to condition v_2 by exploiting vulnerability u.

Attack Paths on the AG represent the possible attacks on the data space and are used to estimate their cyber risks. We leverage existing approaches [15], that consider CVSS metrics[3] to estimate the likelihood and impact of an attack path AP and calculate the risk according to its standard definition: $risk(AP) = likelihood(AP) \cdot impact(AP)$. The likelihood is calculated using the CVSS-3.1 exploitability metrics (Attack Vector, Attack Complexity, Privilege Required, and User Interaction), while the impact is determined by CVSS-3.1 impact metrics (more details in [15]). Thus the risk is assessed for the possible multi-step attacks in the data space.

3.3 Trust Computation

The collection of attack paths and their risks provides an overview of the possible attacks on the federated data spaces. Based on this information, we evaluate the trust through a novel trust computation methodology. Let d_s and d_t be two devices in the federated data space. We leverage AG to calculate the set of attack paths $\langle AP_{d_s d_t,1}, \cdots \rangle$ with entry point d_s and having target d_t. We consider the following trust model parameters:

- R is the maximum risk among the possible attack paths from d_s to d_t defined as $R = max(AP_{d_s d_t,1}, \cdots, AP_{d_s d_t,N})$.
- L is the average attack path length from source d_s to target d_t defined as $L = \frac{\sum_{AP_{d_s d_t,i}} len(AP_{d_s d_t,i})}{N}$, where $len(AP_{d_s d_t,i})$ is the length of the i-th attack path. It informs about the number of vulnerabilities the attacker must exploit to perform the attack. Thus lower L corresponds to less effort for the attacker and consequently lower trust. Thus a higher risk (R) corresponds to lower trust.
- O is a boolean value indicating whether the devices d_s and d_t belongs to the same community. The rationale is that two devices within the same community can be considered more trusted than two in different communities [34].
- C is the average number of different communities traversed by the attack paths. Since each community has its own management (data models, policies, access control) [33], more effort is required for the attacker to intrude into a new community. Thus, lower C corresponds to lower trust.

A device d_t evaluates the trust considering the other devices as possible entry points of an attack. In particular, given a pair of devices $\langle d_s, d_t \rangle$, we define the set of the above parameters as $P_{d_s d_t} = [R, L, O, C]$, and the trust of d_t in d_s is the vector:

$$T[d_s, d_t] = [P_{d_s d_t}, W_{d_s d_t}, F], \tag{1}$$

where $W_{d_s d_t}$ is the vector of weights for each parameter $p \in P_{d_s d_t}$ such that $w_i \in [0, 1]$ and $\sum_i w_i = 1 \forall i \in W_{d_s d_t}$. In addition, F is the trust aggregation function such that $F = f(W_{d_s d_t}, P_{d_s d_t}) \rightarrow \{0, 1\}$, where 0 and 1 are untrusted and trusted, respectively. Common trust aggregation functions used in the literature are Bayesian inference, fuzzy logic, and weighted sum [46]. We chose the latter to weight relevant parameters appropriately:

$$F = \begin{cases} 0, & \text{if } w_R R \geq r_{max} \\ 1, & \text{if } w_R R \leq r_{min} \\ round\left(\sum_{p \in \{L,C,O\}} \left(w_p p + w_R(1 - R)\right)\right), & \text{otherwise} \end{cases}$$

The rationale of this function is that the risk is the main driving parameter. If it is over a certain user-defined threshold r_{max}, it indicates that d_s is a too high-risk entry point for attacks targeting d_t, thus d_s is untrusted for d_t. If it is below a certain user-defined threshold r_{min}, it indicates that d_s is a very low-risk entry point for attacks targeting d_t, then d_s is trusted for d_t. Existing approaches consider low risk as below 0.2 (i.e., $r_{min} = 0.2$) and high risk as above 0.85 (i.e., $r_{max} = 0.85$) [15].

In the other cases, the risk is medium and does not provide a significant indication to assess the attacks starting from d_s and targeting d_t. For this reason, we measure the effort of an attacker to perform such attacks. We consider the average attack path length L normalized using feature scaling to measure it in the range [0,1]. Further, we evaluate the percentage of different communities C, normalized in [0,1] by considering the total number of communities in the federated data space, and the boolean parameter indicating whether d_s and d_t belong to the same community (O). We sum these parameters as they are directly proportional to the trust. On the contrary, the risk is inversely proportional to the trust. Hence, we consider $1 - R$. Let us note that all the parameters are weighted

[3] https://www.first.org/cvss/v3.1/specification-document

to assign different priorities to the trust parameters. The result is rounded to 1 or 0 indicating if d_t can or not trust d_s[4].

A point worth discussing is that an attacker may compromise the trust model parameters. As we focus on security-by-design against multi-step attacks, we assume no attacks during the trust computation. On one hand, it is a reasonable assumption since at design time devices may not even be deployed. On the other hand, we believe this is not a strong assumption even in already deployed environments because it can be easily removed with existing solutions for trust management (e.g., privacy-preserving approaches [29, 46]).

3.4 Security Messaging Mechanism

While security mitigation may exist to patch vulnerabilities, they cannot always be implemented [40]. To this aim, we provide a security-by-design messaging mechanism considering both the trust between the devices and the security mitigation that would be necessary to patch vulnerabilities. We define the security mitigation and then map it into the messaging mechanism.

3.4.1 Security Mitigation. Let u be a vulnerability an attacker may exploit on a device d_t. We define the mitigation of such a vulnerability as an action defined as:

$$m(u, d_t) = [id, phase, strategy, cost, applied]. \quad (2)$$

Phase indicates the step of the software development life cycle (SDLC) where the mitigation is applied. It informs about the *impact* of the mitigation: when it is applied to early phases of the SDLC, its changes propagate to the rest of the process, thus more impactful.

Strategy indicates the high-level operation needed to apply the mitigation and patch the vulnerability (e.g., software installation, encryption, and resource limitation). It informs about the *effort* of applying the mitigation, depending on the actions required.

Cost quantifies specific aspects of the mitigation. It may be related to required resources (e.g., new software) or a function of the *phase* and *strategy* attributes to measure the impact-effort trade-off. It is a context-aware function: for example, updating software in IoT devices is more expensive than other less constrained devices.

Applied is a boolean attribute indicating whether the mitigation is applied on device d_t. When mitigation is applied, the exposure to cyber attacks changes and so does the trust between devices. All mitigation actions are initially "not applied" because if vulnerability scanners detect vulnerabilities, then they are present in the device.

We consider mitigations from the Common Weakness Enumeration (CWE) security standard to retrieve this information[5]. Given the difficulty of patching vulnerabilities in federated data spaces, the next goal is to define a security messaging mechanism abstracting the actual security mitigation.

3.4.2 Security Control Messages. Here we focus on temporarily blocking communication until the trust increases to trade-off patching cost and attack surface reduction. While it may seem that turning off communication in high-risk situations could impair the functionality of a federated data space, it is an essential and realistic step in preventing the spread of attacks through compromised devices. Temporarily blocking communication with untrusted devices is based on security-by-design principles [22]. Once vulnerabilities are patched, the trust computation model re-evaluates the device's trustworthiness, restoring communication when it is safe to do so. We define a set of security messaging events according to the common messaging events in federated data spaces. A *security messaging event* exchanges security information to guarantee security-by-design. We refer to the messages during these events as *security control messages*, based on the following events:

(1) Each device d_i is set to *receive* all the security control messages from its connected devices. In this way, it is updated for every change of the security exposure in the environment.
(2) A device d_t *opt-out* the operational messages from untrusted devices because they are considered unreliable. Additionally, since they are high-risk sources of attacks, d_t interrupts the messaging with them.
(3) When a device d_i patches some of its vulnerabilities, it *sends* a security control message to inform the other devices that are set to receive such messages and can update the security exposure and re-assess the trust[6].

To provide this mechanism, we design two types of security control messages. The first one informs about some constraints caused by the vulnerabilities of a device. For example, suppose a vulnerability could be patched by limiting resources, but this is not currently applicable to the vulnerable device. In that case, it informs the other devices about the resource limits, and they may adapt the messaging accordingly. We refer to it as a *mitigation message*.

Mitigation message.

```
sender: <dev_id>
mitigation: <m_1,···,m_N>
value: <v_1,···,v_N>
```

Patching message.

```
sender: <dev_id>
patched: <u_1,···,u_N>
```

When a device d_i retrieves this message, it can adjust the messaging according to the constraints provided by $m_1, \cdots, m_N$ and their corresponding values $v_1, \cdots, v_N$. For example, if the mitigation is limiting the throughput, the value corresponds to the maximum throughput the device can tolerate.

The second security control message adapts the security exposure after some vulnerability patches. We refer to it as a *patching message*. When a device d_i retrieves this message, it can remove the patched vulnerabilities $u_1, \cdots, u_N$ from the AG and recompute its security exposure and trust. Let us note this message does not disclose sensitive information because it only informs about patched vulnerabilities, i.e., the ones the attacker can no longer exploit.

3.4.3 From Security Mitigation to Security Control Messages. The last step is mapping security mitigations to security control messages. It must consider both the security mitigation and the trust values because the security messaging must adapt to the changes in the security exposure of the federated data space measured through trust. We distinguish the cases where security mitigation is applied and not. Note that we assume no attacks on trust and middleware management and thus, all messages are trusted.

Mitigation not applied. When mitigation is not applied, the mapping function corresponds to setting the initial security exposure of

[4] In case of trust is 0.5, we round to 0 to consider the worst-case scenario.
[5] https://cwe.mitre.org/
[6] While IoT devices are difficult to patch, others (e.g., servers) are easier.

the federated data space. We distinguish two categories according to the CWE strategies. One includes all the strategies that impact other devices (e.g. Resource limitation, Firewall rules). For example, "Resource limitation" implies the sender must adapt the messaging rate according to the receiver's capacity. We refer to this category as $C_{between}$ to indicate it affects the interconnection between devices (e.g., in a collaborative environment). The other strategies mainly impact the internal environment of the devices, having only side effects for the others. For example, "Sandbox" strategy implies that the device isolates its running code and is transparent to the other devices. We refer to this category as C_{within} indicating it affects only aspects within a device. In addition, let $m(u, d_t)$ be the security mitigation of a vulnerability u in device d_t and let $T[d_s, d_t]$ be the trust of d_t in d_s. We consider the following cases when mitigation is not applied:

$$\begin{cases} d_t \text{ opt-out } d_s, & \text{if } T[d_s, d_t] = 0 \\ d_t \text{ receives from } d_s, & \text{if } T[d_s, d_t] = 1 \\ d_t \text{ sends } mitigation\_msg, & \text{if } T[d_s, d_t] = 1 \wedge \\ & m(u, d_t) \in C_{between} \\ \text{no action}, & \text{if } T[d_s, d_t] = 1 \wedge \\ & m(u, d_t) \in C_{within} \end{cases} \quad (3)$$

The rationale of the four cases of Eq. 3 is the following:

- If the device d_t does not trust d_s because it is a high-risk entry point for attacks targeting d_t, then d_t opt-out the messages from d_s as it is unreliable.
- If the device d_t trusts d_s, then d_t receives all the messages from d_s to correctly provide the service.
- If the device d_t trusts d_s and has mitigation affecting other devices (i.e., $C_{between}$), then d_t sends to d_s the *mitigation message* to inform about its constraints. Examples of such messages are maximum throughput and device constraints (e.g., software versions).
- If the device d_t trusts d_s and has mitigation for only its internal environment (i.e., C_{within}), then no action is performed.

Mitigation applied. When a device patches a vulnerability, this changes the security posture of the federated data spaces, therefore the risk and trust models must adapt to these changes. Let d_i be a device that applies mitigation $m(u, d_i)$, then d_i *sends a patching message* to inform the other devices it patched vulnerability u. The devices receiving this message can remove u from the attack graph, re-assess the trust, and configure the security messaging accordingly. Let us remind we are assuming no attacks to the trust model and, consequently, patching messages are trusted and reliable.

4 Evaluation

4.1 Prototype Implementation

While the system design is agnostic to the specific messaging mechanism, we developed a prototype for SHIELD using the publish/-subscribe paradigm, widely used for federated data spaces [27]. In particular, topic-based publish/subscribe allows publishers to categorize their messages under specific topics and distinguish operations (e.g., battery usage). Subscribers, on their part, can selectively receive updates by subscribing to these predefined topics, which ensures that they only receive information pertinent to their operational needs. In addition to these operational topics, we associate each device with its own *security topic* to exchange security control messages. Although one can use other messaging mechanisms (e.g., content-based), the modeled prototype separates the security information of each device, opening the opportunity to extend it with heterogeneous message protection (e.g., cryptography). The SHIELD prototype, its implementation, and all the materials for reproducibility are publicly available[7]. It is important to note that SHIELD is flexible and applicable across a wide variety of federated data spaces, adapting to the complexity of each environment. Its modular architecture can be integrated with new and already operating systems, where AG computation and trust management can be added incrementally with minimal disruption. SHIELD is also as aforementioned communication protocol-agnostic and can be adapted to other communication models with minor adjustments. Despite its adaptability, certain constraints may arise such as legacy or resource-constrained devices that may not support the computational demands of SHIELD's analysis (e.g. calculation of AG). In such cases, performance optimizations may be required.

4.2 Experimental Setup

To experimentally evaluate SHIELD and its prototype, we simulate a real V2X network considering devices currently used by an organization providing charging station services[8] (see Table 2). We

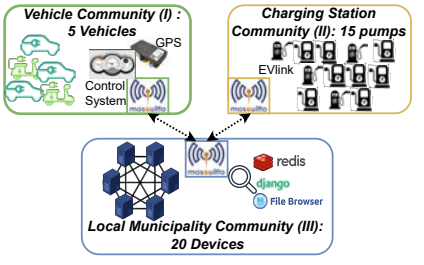


Figure 3: V2X network used for evaluation.

Community	Service
(I) Vehicle	SourceCodester Vehicle Control System v1.0
(I) Vehicle	HQT-401 GPS
(II) Local Municipality	Redis v6.2.6
(II) Local Municipality	Django v3.2
(II) Local Municipality	Filebrowser v2.22
(III) Charging station	EVlink City v3.4.0.1
(III) Charging station	EVlink Smart Wallbox v3.4.0.1
(III) Charging station	EVlink Load Management v4.0.0.13
	Mosquitto Broker with MQTT v5.0

Table 2: Service configuration of the experimental testbed.

created a virtualization of the real network according to the federation topology depicted in Fig. 3 composed of three communities, the Vehicle (I) with 5 vechicles, Charging station (II) with 15 charging stations, and Local Municipality (III), with 20 devices. The devices in the vehicle community use a vehicle control system and GPS to exchange data about power consumption and locations with the management platforms of the local municipality community. The charging stations are from EVlink and run all its software services.

[7] https://github.com/satrai-lab/SHIELD

[8] The V2X network is designed by a national energy provider, which details are not reported to not violate the double-blind review.

They exchange data with the devices in the municipality community. The local municipality community contains devices to monitor and analyze data coming from electric vehicles and charging stations. We leverage topic-based Mosquitto for managing brokers running MQTT v5.0. In this federated data spaces setup, there is one broker associated with each community [33]. We use Nessus and OpenVAS vulnerability scanners to discover CVE vulnerabilities for each device. They reported a total of 83 vulnerabilities in the federated data space that expose it to cross-site scripting, buffer overflow, information disclosure, and distributed denial of service attacks. Detailed vulnerabilities and attacks are available in the open-source repository. The experiments are executed on a PC with an Intel Core i7-11800H 2.3GHz processor and 16 GB memory.

4.3 Security Evaluation

We evaluate the security benefits of SHIELD by considering cyber risk assessment and attack surface reduction. We compare SHIELD with two different scenarios: (i) a naive publish/subscribe system without any security procedure (*Naive*); (ii) a state-of-the-art solution (*SoA*) focused on MQTT security [35]. This latter solution is representative of all works that focus solely on securing the messaging protocol. MQTT is widely used in federated data spaces due to its lightweight and efficient communication model. Our decision to focus on MQTT security in this comparison is to reflect a significant portion of real-world deployments where security measures prioritize securing the communication protocol.

Cyber risk assessment. To assess security-by-design in network systems the cyber risk is commonly used to determine the attack exposure [15]. As such, we measure the average cyber risk in the different communities, reported in Fig. 4a. The bar chart shows that SHIELD outperforms the existing solutions in terms of cyber risk reduction. Without any security component, the cyber risk is 0.4 for all the communities. It is reduced to 0.3 when patching the MQTT vulnerabilities and between 0.1 and 0.3 when using SHIELD. Compared with the naive and SoA approaches, our solution reduces the cyber risk by 60% and 45%, respectively. The main advantage of SHIELD is that it does not focus only on the messaging protocol, but considers all the vulnerabilities in the federated data spaces. This provides a more comprehensive view of the attack exposure, allowing a higher risk reduction, and highlights that existing solutions cover only partially the attack exposure considering attacks to messaging protocols while omitting application and platform vulnerabilities an attacker may exploit. The risk reduction is particularly evident in charging and vehicle communities. This may be due to the fewer links than the local municipality, which is central to the network. The peripheral location of charging and vehicle communities combined with dropping their high-risk links, results in fewer paths to target peripheral communities.

It is interesting to note that in federated data spaces, the primary intent is to reduce the risk of more constrained communities, as they are composed of devices essential to provide business objectives. The rationale of this desiderata is due to the difficulty of patching vulnerabilities in constrained devices. In this sense, SHIELD reduces the risk mostly in charging station and vehicle communities, which are the most critical and constrained ones for the V2X scenario.

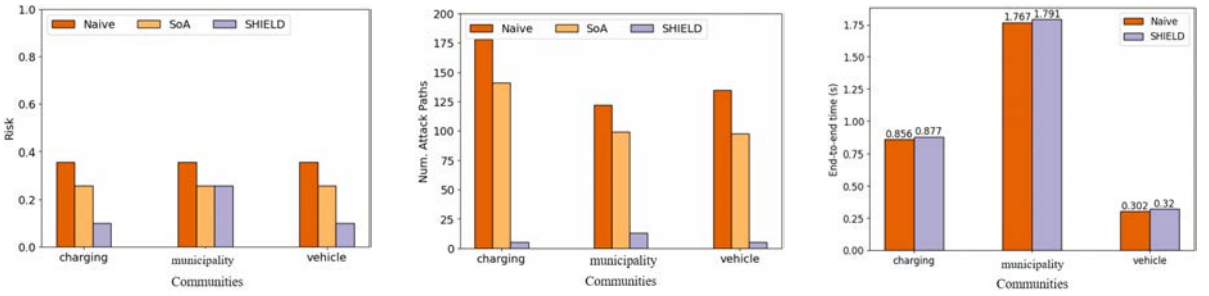
Attack surface reduction. When removing the high-risk messaging links according to the proposed approach, a remote attacker can no longer exploit some vulnerabilities. On the one hand, this slightly impacts the system's performance as we analyze in Fig. 5. On the other hand, it reduces the attack surface in the data space, measured as the number of possible attacks that can be employed, which is the number of paths in the AG (see Section 3.2). To show the attack surface reduction, Fig. 4b reports the number of attack paths targeting each community. The bar chart illustrates that SHIELD notably reduces the count of potential attacks. When there is no security to protect the network (naive solution), the charging station community is the most exposed to cyber attacks, with almost 200 possible attacks targeting it. Differently, the local municipality and vehicle communities are less exposed to cyber attacks with 125 and 140 attacks, respectively. These numbers are reduced by about 20% when considering state-of-the-art approaches, with 140 attacks in the charging community, and 100 in local municipality and vehicle ones. This reduction identifies the impact of securing the messaging protocol (i.e., MQTT). On the contrary, SHIELD considers the exposure of all the vulnerabilities in the federated data spaces, leading to an average reduction of 85% of possible attacks compared to the naive approach and 65% compared to SoA. This indicates that the proposed solution can offer the most significant protection in the attack surface. Note that such protection is against multi-step attacks, meaning that SHIELD reduces the potential strategies of an attacker in the federated data space.

Another interesting aspect from Fig. 4b is that naive and SoA approaches expose the charging community to cyber attacks more than the other communities. In contrast, SHIELD provides higher security for charging and vehicle communities, reducing the number of their attacks to less than 20. The reason behind this result is due to attack graphs, that consider the network topology to model the attacks. In particular, the charging and vehicle communities are more peripheral than the local municipality one. Hence, dropping a few links from them results in a higher reduction of the attack exposure. In fact, network topology should be considered when analyzing attack exposure as it affects the different movements of an attacker. This highlights the rationale of using AGs that, compared with existing approaches, provide more comprehensive attack exposure thanks to its context-awareness.

4.4 Performance and Accuracy Trade-off

To provide security-by-design, SHIELD sends additional messages and drops communication links. This may affect the performance and accuracy of exchanged messages. Thus, we analyze the effects of recomputing AGs and trust when the environment changes dynamically. We compare SHIELD with a naive publish/subscribe system without additional components, i.e., they only exchange messages and do not have any overhead. We keep the same broker infrastructure for the naive and SHIELD simulations.

Overhead of security messaging. We evaluate the performance overhead in terms of response time and number of messages, as they are the main elements affecting the performance of federated data spaces. We run 100 simulations of the message exchange according to the experimental setup (see Section 4.2) and Fig. 4c reports



(a) Cyber risk. (b) Number of attack paths. (c) Average end-to-end response time.

Figure 4: Results of (a) cyber risk, (b) attack surface, and (c) response time for the different communities.

the average end-to-end response time per community. The chart shows that SHIELD introduces an overhead for the response time compared to the naive solution. However, it only marginally affects the messaging performance because the response time overhead is between 0.024 and 0.018 seconds, corresponding to only 1.5% of the total time on average. It is a result we can expect because SHIELD is designed with the rationale of introducing minimum overhead. This is guaranteed by the small number of security control messages exchanged to set up security-by-design, that is one per device. Let us note that such overhead can be tolerated in federated data spaces. In fact, communication shutdowns are common in security-critical environments [22], and fallback modes or minimal operational functions can ensure continued service even during restricted communication.

To prove and validate the impact of dropped messages, we evaluate the messaging accuracy based on the number of messages retrieved by the devices in the different communities:

- True Positives (TP) are the messages commonly exchanged in the data space and exchanged also with SHIELD.
- False Positives (FP) are the messages exchanged with SHIELD, but they should not be exchanged in the data space.
- False Negatives (FN) are the messages commonly exchanged in the data space, but they are not exchanged using SHIELD.
- True Negatives (TN) are the messages that are neither commonly exchanged in the data space nor with SHIELD.

These metrics are collected by emulating traffic in the standard environment as ground truth and repeating the same emulation with SHIELD. Fig. 5 reports the corresponding confusion matrices for each community. As expected, there are no true negatives because they correspond to undetectable messages. The percentage of

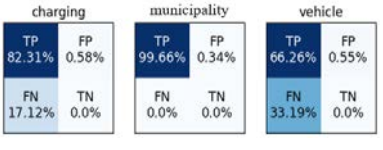


Figure 5: Confusion matrix of exchanged messages.

false positives is negligible between 0.34% in the local municipality and 0.58% in the charging station community. They correspond to the security control messages to set up the security-by-design configuration, which is just one message per device. In contrast, it is interesting to look at the true positives and false negatives. Fig. 5 shows that in the charging and local municipality communities, most of the messages (82.31% and 99.66%, respectively) are correctly delivered and not dropped out by the security unsubscriptions. In the charging community, 17.12% of the messages are not correctly delivered because they are assessed as untrusted and at high risk of corruption. Looking at the vehicle community, the percentage of correct messages is 66.26%, while 33.19% of them are not correctly delivered. Therefore, we can observe a degradation of the accuracy in such a community. The presence of false negative messages in charging and vehicle communities indicates that the approach tends to be more conservative in protecting the corresponding devices. This is a promising result since they represent the devices providing the main business objectives in the proposed V2X scenario, and thus more protected. It is worth noting that the average accuracy of the whole network is 0.82, with precision equal to 0.99 and recall 0.83. This result quantitatively defines the trade-off between performance and security: dropping 18% of the messages is the price to pay to guarantee security-by-design, corresponding to communications at high risk of cyber attacks. We believe the results are promising since the overall accuracy is high and the approach does not significantly drop messaging links nor affect provided services.

Dynamic vulnerability patching. Another aspect to consider for evaluating the performance overhead is the scenario where a device patches some vulnerabilities, thus updating the security exposure. In this case, the main source of overhead is the time necessary to recompute the AG and trust values after a patch. We consider different situations of this type by randomly removing vulnerabilities to simulate their patching. We simulate from 10 to 50 vulnerability patches and measure the time to re-assess the security exposure, including AG and trust computation. We run 100 simulations and report the results in Fig. 6, where the solid line is the average computation time and the alpha blended area shows lower and upper quantiles. Fig. 6 shows the decreasing trend of the computation time with the increasing number of patched vulnerabilities. This is an expected result since decreasing the number of vulnerabilities in the network results in a decreased size of the resulting AG and so

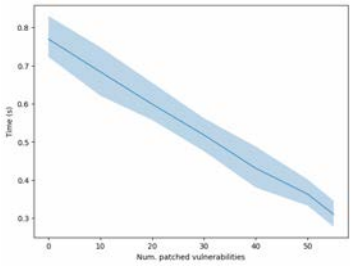


Figure 6: Security exposure recomputation time.

does the computation time. The interesting data is that even in the worst-case scenario, i.e., when no vulnerability is patched, the computation time is relatively small. In security-by-design systems, one can tolerate a higher computation time for the initial assessment because most of the computation is at the beginning and there are not frequent changes due to vulnerability patching [40]. In contrast, we observe a maximum computation time of 0.8 seconds in the worst-case scenario, in line with federated data spaces QoS (and the V2X context too) [27]. Given the short reconfiguration time wrt the response time (see Fig. 4c), an opportunity is to apply SHIELD in real-time response systems [15].

5 Related Work

Security-by-design in data spaces. The state-of-the-art approaches for security-by-design in data spaces are based on the assessment of cyber risks of devices [9]. For example, Vajpayee et al. [44] propose an anomaly detection assessment to assign cyber risks to IoT devices according to a mathematical model. Similarly, Li et al. [25] assess the overload rate of power system components to identify critical nodes, while Alguliyev et al. [1] introduce a multi-criteria decision-making method to rank and identify vulnerable components. These works focus on cyber risk for the detection of critical components but do not address their protection as we proposed.

Different works found a promising solution in AG modeling to support security-by-design. Salayma [39] uses AGs to capture the dynamic changes of IoT networks and assess their impact on security, while Arat et al. [2] employ them to identify critical threats in IoT devices and, similarly, Kushan Sudheera et al. [42] use a machine learning approach to identify attack steps from intrusion alerts. Focusing on the protection perspective, Gressl et al. [16] present a framework that leverages AG to select appropriate secure key placement, and Dar et al. [8] provide security checks in IoT environments. Bhardwaj et al. [3] introduce an AG-based approach to assess cyber risks and suggest rekeying, key discarding, and network node rebooting measures, while Galenbe et al. [12] use reinforcement learning to process the alerts for an attack detector to re-route sensitive traffic away from compromised network paths.

Although these works advance the protection of data space at design time, they still delegate their implementation to human experts or keep the focus on cryptographic measures, which we showed are not sufficient to guarantee comprehensive security-by-design in data spaces. We advance the literature by combining trust management with AG and enabling a novel controlling messaging paradigm for security-by-design in the middleware of data spaces.

Secure messaging. Existing secure messaging mechanisms mainly focus on confidentiality and integrity. Gilles et al. [14] present an approach to secure out-premise authentication for confidential data exchange, and Razouk et al. [37] propose security middleware as a smart gateway to pre-process data at the edge of the network. Given the weaknesses of unencrypted messaging protocols [4], other approaches find a solution for security in messaging by considering encryption for CPU resource limitation [7] or for message tampering and mutual authentication [28]. Similarly, Mukherjee et al. [30] leverage a session resumption algorithm to reuse encrypted sessions and describe an optimal end-to-end security scheme matching device constraints. Finally, other works proposed blockchain technologies as Huang et al. [17] who present a decentralized communication model for multi-tenant cloud, and Zhao et al. [47] propose a blockchain fair payment in publish/subscribe systems, where subscribers specify interest by making a deposit. Existing works focus on the confidentiality and integrity of messaging protocols while overlooking the possibility of attacking the systems bypassing them. In this paper, we filled this gap by proposing a security-by-design solution for federated data spaces by leveraging trust computation.

Trust management. Among the approaches of trust management in data spaces, Pascoal et al. [36] leverage TEE to determine a safe subset of components in a federated environment, while Li et al. [26] propose a security model for deep neural networks by reducing the traversal space through a prejudgment mechanism. With a similar goal, Kalkan et al. [18] introduce a framework for trusted messaging during service discovery in a decentralized P2P network based on a Distributed Hash Table. Other works addressed trust management for QoS like Wang et al. [45] who introduce a game-theoretic mechanism based on the dynamic black-and-white list in edge computing systems, while Khan et al. [21] propose a trust-based approach for managing the reputation based on the Routing Protocol for Low power and Lossy networks. Similarly, Fernandez-Gago et al. [11] introduce a framework to include trust in IoT scenarios based on dynamicity, identity management, and privacy, while Sadique et al. [38] design a hierarchical architecture to model trust as a combination of security and privacy. The relation between trust and security proposed by these approaches focuses mainly on authentication and cryptography. In contrast, we proposed trust computation as a combination of attack graphs and data space features, enabling security-by-design for a comprehensive (i.e., beyond cryptography) attack exposure, otherwise not measurable with existing solutions.

6 Discussion and concluding remarks

This paper proposed SHIELD, a security-by-design solution to mitigate the risks of multi-step attacks in federated data spaces. We have addressed security constraints in IoT devices such as difficulties in patching vulnerabilities through a novel messaging mechanism. We leverage Attack Graphs as a valuable tool to provide comprehensive attack exposure and more informed risk assessment compared with other solutions. A prototype implementation of our approach using publish/subscribe has been experimentally validated in a real V2X scenario. The results show the capability of SHIELD to reduce the risk and attack surfaces up to 65% more than existing approaches, without any significant impact on the messaging performance.

As a security-by-design approach, SHIELD prevents multi-step attacks, which are the ones where the attacker exploits a series of vulnerabilities to reach a certain target. Its main objective is reducing cyber risk through the proposed messaging mechanism traded off with the QoS. As a threat to validity, we are assuming safe trust computation and no attacks on the trust model. The messaging mechanism may introduce vulnerabilities (e.g., fake patching messages) that we investigate in future work by using control checking and digital signatures.

Acknowledgments

This work is supported by the Horizon Europe project DI-Hydro under grant agreement number 101122311 and by the *Avvio alla Ricerca* Sapienza grant AR2241902B426269. We acknowledge as well support from the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No. 101007820. The article reflects only the authors' view and the REA (Research Executive Agency) is not responsible for any use that may be made of the information it contains.

References

[1] R. Alguliyev, R. Aliguliyev, and L. Sukhostat. 2025. An approach for assessing the functional vulnerabilities criticality of CPS components. *Cyber Security and Applications* 3 (2025), 100058.

[2] F. Arat and S. Akleylek. 2023. Modified graph-based algorithm to analyze security threats in IoT. *PeerJ Computer Science* 9 (2023), e1743.

[3] S. Bhardwaj and M. Dave. 2024. Attack detection and mitigation using Intelligent attack graph model for Forensic in IoT Networks. *Telecom. Sys.* 85, 4 (2024).

[4] T. K. Boppana and P. Bagade. 2022. Security risks in MQTT-based Industrial IoT Applications. In *2022 IEEE International Conference on Omni-layer Intelligent Systems (COINS)*. 1–5. https://doi.org/10.1109/COINS54846.2022.9854993

[5] B. Buhnova, D. Halasz, D. Iqbal, and H. Bangui. 2023. Survey on Trust in Software Engineering for Autonomous Dynamic Ecosystems. In *Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing* (Tallinn, Estonia) *(SAC '23)*.

[6] V. Casola, A. De Benedictis, M. Rak, and U. Villano. 2019. Toward the automation of threat modeling and risk assessment in IoT systems. *Internet of Things* 7 (2019).

[7] J. Choi, J. Kim, C. Lim, S. Lee, J. Lee, D. Song, and Y. Kim. 2022. GuardiaNN: Fast and Secure On-Device Inference in TrustZone Using Embedded SRAM and Cryptographic Hardware. In *23rd ACM/IFIP International Middleware Conference*.

[8] A. A. Dar, F. A. Reegu, S. Ahmed, and G. Hussain. 2024. Strategic Security Audit Protocol: Safeguarding Smart Home IoT Devices against Vulnerabilities. In *2024 11th International Conference on Computing for Sustainable Global Development*.

[9] P. Dewitte, K. Wuyts, L. Sion, D. Van Landuyt, I. Emanuilov, P. Valcke, and W. Joosen. 2019. A comparison of system description models for data protection by design. In *34th ACM/SIGAPP Symposium on Applied Computing (SAC '19)*.

[10] P. Eugster, P. Felber, R. Guerraoui, and A.-M. Kermarrec. 2003. The Many Faces of Publish/Subscribe. *ACM Comput. Surv.* 35 (06 2003).

[11] C. Fernandez-Gago, F. Moyano, and J. Lopez. 2017. Modelling trust dynamics in the Internet of Things. *Information Sciences* 396 (Aug. 2017).

[12] E. Gelenbe, P. Fröhlich, M. Nowak, S. Papadopoulos, A. Protogerou, A. Drosou, and D. Tzovaras. 2020. IoT Network Attack Detection and Mitigation. In *9th Mediterranean Conference on Embedded Computing (MECO)*. 1–6.

[13] A. Ghosal and M. Conti. 2020. Security issues and challenges in V2X: A Survey. *Computer Networks* 169 (2020), 107093.

[14] O. Gilles, D. Gracia Pérez, P. A. Brameret, and V. Lacroix. 2023. Securing IIoT communications using OPC UA PubSub and Trusted Platform Modules. *Journal of Systems Architecture* 134 (Jan. 2023), 102797.

[15] G. Gonzalez-Granadillo, S. Dubus, A. Motzek, J. Garcia-Alfaro, E. Alvarez, M. Merialdo, S. Papillon, and H. Debar. 2018. Dynamic risk management response system to handle cyber threats. *Future Generation Computer Systems* 83 (2018).

[16] L. Gressl, A. Rech, C. Steger, A. Sinnhofer, and R. Weissnegger. 2020. Security based design space exploration for CPS. In *Proceedings of the 35th Annual ACM Symposium on Applied Computing* (Brno, Czech Republic) *(SAC '20)*. 3 pages.

[17] B. Huang, R. Zhang, Z. Lu, Y. Zhang, J. Wu, L. Zhan, and P. C. K. Hung. 2020. BPS: A reliable and efficient pub/sub communication model with blockchain-enhanced paradigm in multi-tenant edge cloud. *J. Parallel and Distrib. Comput.* 143 (2020).

[18] K. Kalkan and K. Rasmussen. 2020. TruSD: Trust framework for service discovery among IoT devices. *Computer Networks* 178 (Sept. 2020), 107318.

[19] K. Kaynar. 2016. A taxonomy for attack graph generation and usage in network security. *Journal of Information Security and Applications* 29 (Aug. 2016), 27–56.

[20] M. Kern, F. Skopik, M. Landauer, and E. Weippl. 2022. Strategic selection of data sources for cyber attack detection in enterprise networks: a survey and approach. In *37th ACM/SIGAPP Symposium on Applied Computing (SAC '22)*.

[21] Z. A. Khan and P. Herrmann. 2017. A Trust Based Distributed Intrusion Detection Mechanism for Internet of Things. In *31st International Conference on Advanced Information Networking and Applications (AINA)*. IEEE, Taipei, Taiwan.

[22] H. A. Kholidy. 2021. Autonomous mitigation of cyber risks in the Cyber–Physical Systems. *Future Generation Computer Systems* 115 (2021), 171–187.

[23] S. Kumar, K. Abhishek, R. Jhaveri, A. Alabdulatif, and R. Gaur. 2023. An Efficient Dual Encryption of IoMT data Using Lightweight Security Scheme for Cloud Based IoT Environment. In *Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing (SAC '23)*. 7 pages.

[24] S. Lakshminarayana, A. Praseed, and P. S. Thilagam. 2024. Securing the IoT Application Layer from an MQTT Protocol Perspective: Challenges and Research Prospects. *IEEE Communications Surveys & Tutorials* (2024), 1–1.

[25] J. Li, Y. Lin, and Q. Su. 2024. Identifying critical nodes of cyber–physical power systems based on improved adaptive differential evolution. *Electric Power Systems Research* 229 (2024), 110112.

[26] L. Li, X. Zhao, J. Fan, F. Liu, N. Liu, and H. Zhao. 2024. A trustworthy security model for IIoT attacks on industrial robots. *Future Gen. Comp. Sys.* 153 (2024).

[27] K. L. Lim, J. Whitehead, D. Jia, and Z. Zheng. 2021. State of data platforms for connected vehicles and infrastructures. *Comm. in transportation res.* 1 (2021).

[28] L. Malina, G. Srivastava, P. Dzurenda, J. Hajny, and R. Fujdiak. 2019. A Secure Publish/Subscribe Protocol for Internet of Things. In *14th International Conference on Availability, Reliability and Security (ARES '19)*.

[29] I. Messadi, M. H. Becker, K. Bleeke, L. Jehl, S. B. Mokhtar, and R. Kapitza. 2022. SplitBFT: Improving Byzantine Fault Tolerance Safety Using Trusted Compartments. In *Proceedings of the 23rd ACM/IFIP International Middleware Conference* (, Quebec, QC, Canada,) *(Middleware '22)*. 13 pages.

[30] B. Mukherjee, S. Wang, W. Lu, R. L. Neupane, D. Dunn, Y. Ren, Q. Su, and P. Calyam. 2018. Flexible IoT security middleware for end-to-end cloud–fog communication. *Future Generation Computer Systems* 87 (Oct. 2018), 688–703.

[31] J. Navarro, A. Deruyver, and P. Parrend. 2018. A systematic survey on multi-step attack detection. *Computers & Security* 76 (2018), 214–249.

[32] N. Neshenko, E. Bou-Harb, J. Crichigno, G. Kaddoum, and N. Ghani. 2019. Demystifying IoT Security: An Exhaustive Survey on IoT Vulnerabilities and a First Empirical Look on Internet-Scale IoT Exploitations. *IEEE Communications Surveys & Tutorials* 21, 3 (2019), 2702–2733.

[33] N. Papadakis, G. Bouloukakis, and K. Magoutis. 2023. ComDeX: A Context-aware Federated Platform for IoT-enhanced Communities. In *17th ACM International Conference on Distributed and Event-Based Systems (DEBS '23)*. 12 pages.

[34] N. Papadakis, G. Bouloukakis, and K. Magoutis. 2024. CCDUIT: a software overlay for cross-federation collaboration between data spaces. In *21st IEEE International Conference on Software Architecture (ICSA)*. IEEE, Charminar, Hyderabad, India.

[35] C.-S. Park and H.-M. Nam. 2020. Security Architecture and Protocols for Secure MQTT-SN. *IEEE Access* 8 (2020), 226422–226436.

[36] T. Pascoal, J. Decouchant, and M. Völp. 2022. Secure and distributed assessment of privacy-preserving GWAS releases. In *Proceedings of the 23rd ACM/IFIP International Middleware Conference*. ACM, Quebec QC Canada.

[37] W. Razouk, D. Sgandurra, and K. Sakurai. 2017. A new security middleware architecture based on fog computing and cloud to support IoT constrained devices. In *1st International Conference on Internet of Things and Machine Learning*.

[38] K. M. Sadique, R. Rahmani, and P. Johannesson. 2018. Trust in Internet of Things: An architecture for the future IoT network. In *International Conference on Innovation in Engineering and Technology (ICIET)*. 1–5.

[39] M. Salayma. 2024. Threat modelling in Internet of Things (IoT) environments using dynamic attack graphs. *Frontiers in The Internet of Things* 3 (2024).

[40] T. Sasi, A. H. Lashkari, R. Lu, P. Xiong, and S. Iqbal. 2023. A Comprehensive Survey on IoT Attacks: Taxonomy, Detection Mechanisms and Challenges. *Journal of Information and Intelligence* (2023).

[41] J. B. F. Sequeiros, F. T. Chimuco, T. M. C. Simões, M. M. Freire, and P. R. M. Inácio. 2024. An Approach to Attack Modeling for the IoT: Creating Attack Trees from System Descriptions. In *Adv. Inf. Net. and App.* Springer Nature Switzerland.

[42] K. L. K. Sudheera, D. M. Divakaran, R. P. Singh, and M. Gurusamy. 2021. ADEPT: Detection and Identification of Correlated Attack Stages in IoT Networks. *IEEE Internet of Things Journal* 8, 8 (2021), 6591–6607.

[43] S. N. Swamy, D. Jadhav, and N. Kulkarni. 2017. Security threats in the application layer in IOT applications. In *International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*. 477–480.

[44] P. Vajpayee and G. Hossain. 2024. Risk Assessment of Cybersecurity IoT Anomalies Through Cyber Value at Risk (CVaR). In *IEEE World AI IoT Congress (AIIoT)*.

[45] B. Wang, M. Li, X. Jin, and C. Guo. 2020. A Reliable IoT Edge Computing Trust Management Mechanism for Smart Cities. *IEEE Access* 8 (2020).

[46] L. Wei, Y. Yang, J. Wu, C. Long, and B. Li. 2022. Trust Management for Internet of Things: A Comprehensive Study. *IEEE Internet of Things Journal* 9, 10 (2022).

[47] Y. Zhao, Y. Li, Q. Mu, B. Yang, and Y. Yu. 2018. Secure Pub-Sub: Blockchain-Based Fair Payment With Reputation for Reliable Cyber Physical Systems. *IEEE Access* 6 (2018), 12295–12303.