Artículos de investigación

Semiotics: An Approach to Model Security Scenarios for IoT-Based Agriculture Software

Semiótica: un enfoque para modelar escenarios de seguridad para software de agricultura basado en IoT

Notas de autor

adrianagomezr@utp.edu.co

Julio Ariel Hurtado
Universidad del Cauca, Colombia
ahurtado@unicauca.edu.co
https://orcid.org/0000-0002-2508-0962
Leandro Antonelli
Universidad Nacional de La Plata, Argentina
lanto2004@gmail.com
https://orcid.org/0000-0003-1388-0337
Santiago López
Universidad del Cauca, Colombia
santiagolopez94@unicauca.edu.co
https://orcid.org/0000-0002-8588-8365
Adriana Gómez
Universidad Tecnológica de Pereira, Colombia
adrianagomezr@utp.edu.co
https://orcid.org/0009-0008-5686-6408
Juliana Delle Ville
Universidad Nacional de La Plata, Argentina
jdelleville@lifia.info.unlp.edu.ar
https://orcid.org/0009-0007-7888-7544
Giuliana Maltempo
Universidad Nacional de La Plata, Argentina
gmaltempo@lifia.info.unlp.edu.ar
https://orcid.org/0009-0005-7441-5828
Frey Giovanny Zambrano
Universidad del Cauca, Colombia
freyzambrano@unicauca.edu.co
https://orcid.org/0009-0007-9629-4072
Andrés Solis
Corporación Universitaria Comfacauca, Colombia
asolis@unicomfacauca.edu.co
https://orcid.org/0000-0003-3342-0776
Marta Cecilia Camacho
Institución Universitaria Colegio Mayor del Cauca, Colombia
cecamacho@unimayor.edu.co
https://orcid.org/0000-0003-1973-3063
Miguel Solinas
Universidad Nacional de Córdoba, Colombia
miguel.solinas@unc.edu.ar
https://orcid.org/0000-0002-7550-1067
Gladys Kaplan
Universidad Nacional de La Matanza, Argentina
gladyskaplan@gmail.com
https://orcid.org/0000-0003-1800-7342
Freddy Muñoz
Fundación Universitaria de Popayán, Colombia
lfreddyms@gmail.com
https://orcid.org/0000-0002-8172-0530

Recepción: 14 Noviembre 2023
Aprobación: 12 Marzo 2024
Publicación: 17 Abril 2024

Abstract

Agriculture is a vital human activity that contributes to sustainable development. A few decades ago, the agricultural sector adopted the Internet of Things (IoT), which has played a relevant role in precision and smart farming. The IoT developments in agriculture require that numerous connected devices work cooperatively. This increases the vulnerability of IoT devices, mainly because they lack the necessary built-in security because of their context and computational capacity. Other security threats to these devices are related to data storage and processing connected to edge or cloud servers. To ensure that IoT-based solutions meet functional and non-functional requirements, particularly those concerning security, software companies should adopt a security-focused approach to their software requirements specification. This paper proposes a method for specifying security scenarios, integrating requirements and architecture viewpoints into the context of IoT for agricultural solutions. The method comprises four steps: (i) describe scenarios for the intended software, (ii) describe scenarios with incorrect uses of the system, (iii) translate these scenarios into security scenarios using a set of rules, and (iv) improve the security scenarios. This paper also describes a prototype application that employs the proposed algorithm to strengthen the incorrect use scenario based on the correct use scenario. Then, the expert can complete the information for the analysis and subsequent derivation of the security scenario. In addition, this paper describes a preliminary validation of our approach. The results show that the proposed approach enables software engineers to define and analyze security scenarios in the IoT and agricultural contexts with good results. A survey administered to five security experts found that the proposed security scenario method is generally useful for specifying agricultural IoT solutions but needs improvement in different areas.

Keywords: IoT, Quality Scenario, IoT Requirements, Smart Farming, Industry 4.0, Intelligent Systems.

Resumen

La agricultura es una actividad humana vital que contribuye al desarrollo sostenible. Hace unas décadas, el sector agrícola introdujo el Internet de las Cosas (IoT), desempeñando un papel relevante en la agricultura de precisión e inteligente. Los desarrollos IoT en agricultura requieren colaboración entre múltiples dispositivos, lo que incrementa su vulnerabilidad, debido principalmente a la falta de seguridad integrada por restricciones del contexto. Otras amenazas a estos dispositivos conciernen el almacenamiento y procesamiento de datos conectados a servidores periféricos o en nube. Para garantizar que las soluciones IoT cumplen los requisitos funcionales y no funcionales, especialmente los de seguridad, las empresas de software deberían adoptar un enfoque centrado en la seguridad para su especificación de requerimientos de software. El objetivo del artículo consistió en proponer un método ligero para especificar escenarios de seguridad integrando los puntos de vista de requisitos y arquitectura en el contexto del IoT en soluciones agrícolas. El método comprende cuatro actividades: (i) crear escenarios de buen uso, (ii) crear escenarios de uso incorrecto, (iii) traducir el escenario anterior en escenario de seguridad aplicando reglas y (iv) refinar el escenario de seguridad resultante. También se describe un prototipo de herramienta que utiliza el algoritmo propuesto para ayudar a reforzar el escenario de uso incorrecto basado en el escenario de uso correcto, dando al experto la posibilidad de completar la información para el análisis y posterior derivación del escenario de seguridad. Por último, se proporciona una evaluación preliminar del método propuesto. Los resultados de mostraron que el enfoque propuesto permite a los ingenieros de software definir y analizar escenarios de seguridad en los contextos de IoT y agricultura con buenos resultados. La encuesta, aplicada a cinco expertos en seguridad, encontró que el método de escenario de seguridad propuesto es generalmente útil, pero necesita mejoras en diferentes áreas.

Palabras clave: IoT, Escenario de Calidad, Requerimientos de IoT, Agricultura Inteligente, Industria 4.0, Sistemas Inteligentes.

Highlights

- Security Scenarios for Robust Ag-IoT Architecture
- Uncovers Misuse Cases for Countermeasures
- Lightweight, 4-Step Security Scenario Method
- Expert Validation: Useful for Ag-IoT Security
- Improves Agile Development for Smart Farms

Highlights

- Escenarios de seguridad para una arquitectura robusta de Ag-IoT
- Descubre casos de uso indebido de contramedidas
- Método de escenario de seguridad ligero de 4 pasos
- Validación de expertos: útil para la seguridad Ag-IoT
- Mejora el desarrollo ágil para granjas inteligentes

1. INTRODUCTION

The International Telecommunication Union (ITU) defines IoT (Internet of Things) as a "global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving interoperable information and communication technologies" [1]. According to [2], as an interconnected network, the IoT contributes to making decisions based on the information collected, and its interaction does not require human intervention. This definition includes the concept of a cyber-physical system, which is a complex abstraction that requires a conceptual map [3] rather than a simple definition to explain the concept.

In software development, requirements analysis is a critical activity to define software functionalities, attributes, and quality properties. This process has distinctive characteristics when the software is constructed using emergent technologies like the IoT. Therefore, traditional software development practices must be adapted to these new technologies and business contexts [2]. Requirements engineering involves collaboration between clients and development teams in order to incorporate the right features into the finished product [4]. Inconsistencies between initial requirements and the final product could lead to reengineering processes, increasing the project's scope and cost of the project [5]. Requirements engineering works with two types of knowledge: explicit and tacit [6]. Tacit knowledge is difficult to communicate because experts and development teams often have different backgrounds and use different terminologies [7]–[9], making it challenging to obtain information from stakeholders.

Software products are defined by a set of functional and non-functional requirements. The latter determine the product's quality and are most frequently considered when an IoT system is developed according to its specific application domain [2]. One way of specifying software requirements is to describe *use scenarios*employing storytelling techniques. The effectiveness of this approach lies in the possibility of incorporating details that are essential to achieve a rich consolidation of knowledge. Scenarios use natural language, allowing experts to use them without complex formalisms. This makes them highly effective in promoting communication and collaboration among diverse groups of experts [10], [11].

In the development of IoT products, the main challenges for non-functional requirements are limited processing and storage capacity, performance reliability, availability, accessibility, interoperability, security, privacy, scalability flexibility, and context awareness [2], [12]. It follows that security is a highly relevant aspect in IoT-based software because it concerns the protection of resources such as modules, code, and others from

unauthorized access [12]. Using scenarios, experts from different domains can describe various situations and work together to improve them, learning from one another in the process. This can be especially valuable when dealing with complex problems that require input from multiple perspectives. Overall, scenario-based design can be a powerful tool for fostering cooperation and achieving better outcomes in a wide range of domains.

A software architect should consider incorporating security into a whole system as soon as stakeholders identify security concerns rather than adding security technologies in an ad-hoc manner [13]. As Bruce Schneier points out [14], security is a process and a chain that is only as strong as its weakest link. Therefore, software providers should adopt a security-centric approach to designing and developing IoT-based solutions that meet functional and non-functional requirements like security [15].

The agricultural sector now requires data collection and advanced technologies to improve production while using limited resources. Sustainable agriculture can help to preserve nature without compromising the needs of future generations [16], [17]. The Food and Agriculture Organization (FAO) of the United Nations has identified population growth, resource scarcity, and degradation as key future challenges. There is a need to increase the efficiency, productivity, and quality of agrifood systems while protecting the environment [18]. To achieve this, new developments and technologies must be introduced to automate traditional farming methods and make farm labor more efficient. The Internet of Things (IoT) seems to be able to transform these conventional processes [16], [17].

Compared with traditional IT systems, their IoT counterparts face distinct security challenges that are primarily due to the presence of resource-constrained devices. Currently, the of IoT platforms are not adequately structured to handle different threats and attacks in an organized manner. These limitations make IoT systems more susceptible to a wide range of attack vectors, posing potential threats to their security [19]. Furthermore, traditional protection schemes used in the conventional internet and IoT are not as useful for agricultural systems, which creates opportunities and research gaps [20]–[22]. Therefore, an accurate identification and understanding of their specific security requirements is crucial to develop such IoT-based agricultural systems.

This paper proposes a scenario-based method for specifying security aspects. The method is composed of four essential steps: (i) describe scenarios for the intended software application, (ii) describe scenarios related to the previous ones but in which the application is used incorrectly, (iii) apply a set of rules to map attributes from the previous scenarios into architectural scenarios, and (iv) describe the architectural scenarios in more detail. Additionally, this paper describes a preliminary evaluation of the proposed approach. Considering the security challenges that agricultural IoT faces today, this paper addresses the following research question: *How can we adequately elicit security requirements for smart IoT-based agricultural solutions?*

The proposed method includes an application prototype called Requirement Healer, which uses natural language processing techniques. Its aim is to make the information contained in a scenario more robust by applying natural language processing techniques to extend the scenarios with precise information extracted from catalogs designed for this prototype. Our prototype provides support for the four steps in the proposed method. The prototype provides the user with a form to write about scenarios for a software application, including incorrect use scenarios. It allows the user to sort all the scenarios by name and, after selecting one incorrect use scenario, it derivates security scenarios by applying the mapping rules proposed here. Then, the user only needs to edit the security scenarios generated by the prototype in order to fine-tune their description.

The paper is organized in the following way. Section 2 describes some background to the scenarios. Then, Section 3 reviews related works. Section 4 details our contribution, namely the proposed approach. Section 5 presents a preliminary evaluation. Section 6 introduces the supporting prototype that complements the information about the scenarios. Finally, Section 7 discusses our conclusions.

2. BACKGROUND

This section describes two types of scenarios: scenarios that focus on the functionality of a software application and those that focus on architectural security concerns.

2.1 Scenarios describing functionality

A scenario [10], [11] is an artifact that describes situations (within the application or the software domain) using natural language. It describes a specific situation that may arise in a certain context to achieve some goal. The scenario includes a set of steps (episodes) to reach that goal. In the episodes, active agents (actors) use materials, tools, and data (resources) to perform some specific action. Although there are many templates to describe scenarios, this paper will use the one proposed by [23]. Table 1 summarizes the template.

Table 1.

Template for describing scenarios that focus on functionality.

Attribute	Description
Scenario title	ID
Goal	Objective
Context	Starting point (time, place, activities previously achieved)
Actors	Active agents
Resources	Passive elements (tools, materials, data)
Episodes	List of actions, simple breakdown with no conditions, no iterations

Source: Own work.

Let's consider the following example scenario describing how an irrigation system is activated. This task could be done in different ways depending on the technological infrastructure of the farm. For example, an operator could manually start the irrigation by physically entering the machine room where the pumps are. In this situation, no IoT software application is involved. Instead, this paper will focus on another kind of scenario where it is a software application that activates the pumps. Now, the example goes as follows: An expert in agriculture evaluates the field conditions to determine whether irrigation is necessary and provides the information to the farm supervisor. Then, the supervisor activates the irrigation pipe using an IoT-based web application. Table 2 summarizes the situation.

The previous scenario describes an authorized person's legitimate use of the software application to activate the irrigation system. This scenario could be similar to a use case or user story [7]–[9]. However, the software system could be vulnerable to hacking attacks, where a malicious user intends to break into the web software application to start the irrigation system either just for fun or to destroy the crop. This incorrect and harmful utilization of the software application is regarded as a misuse case [24].

Table 2.
Authorized attempt to start the irrigation system.

Attribute	Description
Scenario title	Attempt to access the water irrigation infrastructure by an authorized person.
Goal	Protect access to the water irrigation system to ensure that water is used responsibly.
Context	The farm has irrigation infrastructure (pipes, tanks, pumps, and valves) to water (irrigate) the field.
Actors	Expert, Supervisor.
Resources	Checklist to determine if it is necessary to irrigate the field. Security protocol to have access to and operate the pump and valves.
Episodes	· An expert evaluates the conditions of the field to determine if it is necessary to irrigate. · The expert writes a report to the supervisor with the recommendation to irrigate. · The supervisor logs in to the IoT web application. · The supervisor starts the pump and opens the valves.

Source: Own work.

2.2 Scenarios describing architectural security concerns

Software architecture is the process of designing a system's fundamental structure and organization to achieve specific quality attributes. Quality Attributes (QA) are critical non-functional characteristics that determine the system's overall effectiveness. QAs are specified using quality scenarios, which define how the system should behave under various conditions. A Quality Attribute scenario is a specific, testable scenario that demonstrates how a QA requirement is satisfied. A QA scenario is typically structured with an ID, a stimulus that triggers the interaction with the software application, the environment where the interaction occurs, the affected artifact, the response, and some quantitative description of the response. Table 3 summarizes a template for this kind of scenarios.

Table 3.
Security scenario template.

Attribute	Description
Scenario ID	Unique Some identification of the scenario.
Source of the Stimulus	Some human, system or any other actor generates a stimulus to the system.
Stimulus	It is an input condition that generates a response from the system.
Environment	The stimulus occurs under a certain context. The system may have an overload context, normal operation, or some other relevant state. For many systems, "normal" operation can refer to one of a number of modes. For these kinds of systems, the environment should specify in which mode the system is executing
Artifact	The stimulated artifact. This may be an ecosystem, a whole system, a component, or some set of components.
Response	It is the response as the result of the arrival of the stimulus.
Response Measure	The response should be measurable so that the requirement can be tested.

Source: Own work.

Security refers to a system's capability to defend itself against danger, ensure its safe-ty, and protect system data from unauthorized disclosure, modification, or destruction. Security involves protecting computer systems using technical and administrative safeguards. Additionally, security refers to the degree to which a particular security policy is enforced with some level of assurance. The three fundamental types of security concerns are confidentiality, integrity, and availability. Confidentiality refers to the protection of data and processes from unauthorized disclosure or access by individuals or entities that are not authorized to access them. Integrity refers to protecting data and processes from unauthorized modification, whether intentional or accidental. It includes ensuring that data are not tampered with or corrupted during storage, processing, or transmission. And availability refers to the protection of data and processes from denial-of-service attacks or other forms of disruption that could prevent authorized users from accessing or using them. This includes ensuring that systems are available and responsive when needed and that they can handle high levels of traffic or activity without becoming over-loaded or crashing. Table 4 is an example of a security scenario that refers to the same situation as the requirement scenario described in Table 2.

Table 4.

Security scenario example.

Attribute	Description
ID	S01
Source of stimulus	An unauthorized individual attempts to access the water irrigation system through an IoT-connected device.
Stimulus	The individual attempts to gain access to sensitive data (sensor measurements) or to manipulate the system's functionality (change the valve behavior).
Environment	Normal execution. The system has IoT-connected devices that are used to access the functionality of the solution, such as sensors, actuators, and processors.
Artifact	Security protocol and access control subsystem.
Response	The security protocols detect the unauthorized access attempt and ban the individual from the access control subsystem.
Response measure	Ensure the security of the system. Attacks should be detected quickly, ideally within 0.5 seconds. Additionally, the system must have a high rate of success in resisting attack attempts, with a target success rate above 95 %.

Source: Own work.

3. RELATED WORK

The complexity of IoT software applications is a concern that has been identified by several researchers. Thus, some proposals to deal with this complexity are available [25] proposed FRASAD, a model-driven software development framework to manage the complexity of IoT applications. [26] proposed another approach to deal with this complexity. Their approach includes activities such as requirements development, domain-specific design, verification, simulation, analysis, calibration, deployment, code generation, and execution. However, none of these proposals considers security, which is our main concern.

Some other approaches have indeed considered the security issue, but in terms of implementation, whereas our proposal considers security in terms of requirements specification. [27] proposed a process and a tool to apply formal methods in IoT applications using the Unified Modeling Language (UML). They developed a plug-in tool that validates UML software models to design secure software applications. [28] presented a

taxonomy of security requirements that should be considered when a software application is designed and imp lemented. [19] proposed a security architecture to provide security enabled IoT services and a baseline for security deployment. Their architectural solution plays a crucial role in their study because it addresses the security requirements of IoT systems. These security requirements are useful components of our security scenarios proposal. By focusing on these requirements, we can effectively establish a robust security framework at the requirements level. [29] established security requirements for IoT systems and focused on enhancing the security of smart home applications. The requirements they identified complement our proposal because they introduce significant vocabulary for describing security scenarios in the context of the IoT and smart farming. By incorporating these elicited requirements, we can effectively address the specific security challenges and other considerations associated with IoT and smart farming environments.

Multiple approaches in the literature have considered security among software requirements, but they have not addressed how to specify security requirements precisely. In [30] was presented a comprehensive literature review of IoT security requirements, but they did not include any references on how to specify them. The proposal by [31] deals with different non-functional requirements: security, scalability, and performance. They tried to balance different requirements or decide which one to satisfy when there is a conflict. [32] also dealt with conflicts, but her approach involves non-functional requirements.

Finally, [33] presented an approach to specify security requirements for IoT applications. They combined a framework for requirement elicitation with automated reasoning to provide secure IoT for vulnerable users in healthcare scenarios. They mapped technical system requirements using high-level logical modelling. Then, they performed an attack tree analysis and a security protocol analysis. Their work concentrated on the attack tree analysis to identify the situation, while our approach focuses on how to describe security requirements precisely.

4. OUR APPROACH

This section describes our general approach, followed by a detailed explanation of each step.

4.1 Our approach in a nutshell

Our proposed approach consists of several steps. First, we describe scenarios that outline the intended use of the software. Next, we create scenarios that describe incorrect use of the application in an attempt to exploit any vulnerabilities. We then convert these scenarios into security scenarios by applying a set of preestablished rules. Lastly, we refine and improve the security scenarios. Figure 1 summarizes our approach.

Figure 1.
Our approach in a nutshell.
Source: Own work.

4.2 Description of scenarios with correct use of the indented software application

The first step is the description of scenarios that focus on the correct use of the software application regarding security concerns. This step should be completed by a requirement engineer or analyst (or a group of them), who should interact with domain experts (clients, users, and stakeholders in general) to capture the requirements for the software application and specify scenarios. They should describe the functionality of the intended software and also consider security concerns. Therefore, the analyst eliciting and defining scenarios

should have some background knowledge of non-functional security requirements in order to include this concern in the specifications. The result of this step is a set of scenarios that describe the functionality, as illustrated in Table 2.

4.3 Description of scenarios with incorrect use of the indented software application

The second step is analyzing the scenarios that were described in the previous step to find security issues. Issues that exploit the problems and compromise the security of the software application are described. Ideally, this step should be completed by the same requirements engineer (or group of them) that participated in the previous tasks. They should analyze every scenario in detail and consider guidelines such as those proposed by [34], [21]. Then, they describe scenarios of incorrect use of the software application. Basically, they should describe scenarios that exploit possible vulnerabilities. For example, considering the scenario of correct use of the software application to activate the irrigation system (Table 2), the requirements engineer may determine that the access to the system (and therefore the access to the activation of the pumps) constitutes a security breach. Hence, they describe a scenario where an unauthorized person gains access to the software application and, consequently, to the irrigation infrastructure. Table 5 describes this scenario of unauthorized access.

Table 5.

Unauthorized attempt to start the irrigation system.

Scenario title	Attempt to access the water irrigation infrastructure by an unauthorized person.
Goal	Protect access to the water irrigation system to ensure that water is used responsibly.
Context	The farm has irrigation infrastructure (pipes, tanks, pumps, and valves) to water (irrigate) the field.
Actors	Unauthorized person.
Resources	Checklist to determine if it is necessary to irrigate the field. Security protocol to access and operate the pump and valves.
Episodes	•An unauthorized person gains access to the IoT web application. •An unauthorized person starts the pump and opens the valves.

Source: Own work.

4.4 Derivation of security scenarios

In this step, a set of rules is described to map the information contained in an incorrect use scenario. The goal is to obtain a first draft of a scenario describing security concerns. It is worth mentioning that the incorrect use scenario will not provide enough information for a complete security scenario. The rules proposed here use only four attributes from the incorrect use scenario (title, context, actors, and resources) to fill out four attributes of the security scenario (stimulus, environment, source of the stimulus, and artifact).

With this information, the following step is to refine the security scenario. Table 6 summarizes the mapping between attributes of the two types of scenarios. Following the example of the incorrect use scenario described in Table 5, the scenario obtained by applying the proposed rules is shown in Table 7. This preliminary scenario is still far from being a full-fledged security scenario such as that described in Table 4. It needs to be refined in the following step.

Table 6.
Mapping rules between attributes of the incorrect use and security scenarios.

Attribute of the incorrect use scenario	Attribute of the security scenario
Title	Stimulus
Context	Environment + Source of the stimulus
Actors	Sources of stimulus
Resources	Artifact

Source: Own work

Table 7.
Result of applying the mapping rules between attributes of the incorrect use and security scenarios.

Attribute	Description
Stimulus	Attempt to access the water irrigation infrastructure by an unauthorized person.
Environment + Source of stimulus	The farm has an irrigation infrastructure (pipes, tanks, pumps) to water (irrigate) the field. An unauthorized person attempts to operate the pump and the valve to irrigate the field.
Source of stimulus	Unauthorized person.
Artifact	Checklist to determine if it is necessary to irrigate the field. Security protocol to access and operate the pump and valves.

Source: Own work.

4.5 Refinement of security scenarios

Some adjustments and improvements should be made to the scenarios derived from the mapping in the previous step. Some new information should be added, and some should be rephrased. The requirements engineer should use their experience and knowledge to provide further information and paraphrase other based on the elicitation meeting and their expertise in the field. First, the security scenario should be assigned an ID to identify it in the software development process; this is a minor task related to documentation definitions. Afterwards, the *stimulus, environment, source of stimulus, and artifact* attributes should be rephrased to adapt the information obtained in the previous step. Both the *environment* and *source of stimulus* attributes capture data from a single attribute in the incorrect use scenario (i.e., *context*). Therefore, in the security scenario, the information in *context* should be split into two attributes. Finally, the *response* and *response measure* attributes should be defined from scratch. Although the mapping rules do not provide information about these two attributes, the descriptions found in the rest of the scenario provide the context that requirements engineers need to define them. Requirements engineers should bear in mind that the *response measure* attribute, in particular, should be described with quantitative measures. The tool described in the following section can support this task.

Table 8 summarizes the necessary refinements in this step. The scenario described at the beginning of this paper in Table 4 is an example of the kind of scenario that this approach aims to obtain.

Table 8.
Refinement to the security scenarios.

1. An *ID* must be assigned.	2. *Stimulus* must be rephrased.	3. *Context* must be split in two attributes (i.e., *environment* and *source of stimulus*).
4. *Source of stimulus* must be rephrased.	5. *Artifact* must be rephrased.	6. *Response* and *response measure* must be added

Source: Own work.

Security scenarios in smart farms and IoT require a specific vocabulary so that they are accurately described [29] argue, there are several concerns to take into account as part of these scenarios. One such concern is technology-dependent security for IoT devices (*artifact*), which refers to the security measures required in the IoT context (*environment*). Another important aspect is the authentication of IoT objects and individuals (*sources of stimulus*) using various mechanisms to prevent or detect attacks (*responses*). These responses to potential security threats have several limits (*response measure*). Requirements engineers could use this vocabulary as specialized terminology and a semiotic tool.

5. ASSESSMENT OF THE APPROACH

5.1 Assessment Design

Next, we assessed the acceptance of our approach by security experts in the field of IoT-based smart agriculture, using the Technology Acceptance Model (TAM) [35], [36] to guide our evaluation. Specifically, we were interested in understanding to what extent our approach was accepted by its target audience. To evaluate the usefulness and ease of use of our approach, we adopted two well-known and widely used metrics-Perceived Usefulness and Perceived Ease of Use-as defined in Fred D. Davis's work [35]. To this end, we designed and administered a survey to a group of security professionals who are representative of our target audience and have experience in eliciting security requirements. Conducting the survey with this group provided us with valuable feedback and insights that helped us to identify strengths and weaknesses in our approach and ultimately improve its overall acceptance.

5.2 Survey Application and Data Collection

We conducted a survey with a group of five experts in the field of software and network security. Prior to the survey, we presented our methodology to the group and spent approximately 40 minutes discussing and addressing any questions they had. Once we presented our approach, we administered a survey that included 17 close-ended and three open-ended questions. The survey aimed to gather insights from the experts on the perceived ease-of-use and usefulness of our method. Most of the experts found the proposed security scenario method to be a useful tool for specifying the requirements of agricultural IoT solutions. Half of them think that the proposed method simplifies the process of specifying security requirements, resulting in better quality and control of the specification.

The experts noted that the proposed method is well-defined, easy to understand, and flexible, making it ideal for defining scenarios. Additionally, the evaluation revealed that most (over 60 %) found it to be clear, well-structured, and interactive in its.

5.3 Results and Analysis

While the method was generally perceived as useful and easy to use for developing security scenarios, it was suggested that it needs to be more specific to determine its usefulness in practice. The experts suggested that the method could be enhanced to include specific aspects of cybersecurity, as well as development and implementation elements that are essential to ensuring the security of agricultural IoT systems. This would allow for a complete specification of the security requirements of such systems. Furthermore, it was noted that users need to interact with the method to remember its steps. During the evaluation, the experts identified some areas for improvement, such as incorporating vulnerabilities and risks commonly found in IoT systems, considering different types of users and adversaries, and taking into account various attack vectors.

By applying these suggestions, the proposed method could be further refined to better meet the needs of users and enhance the security of agricultural IoT systems, particularly adding this information to the terminology of the field.

6. PROTOTYPE OF THE SUPPORT TOOL

A computer tool (software application) was prototyped to support the approach proposed in this article. This prototype aims to make the incorrect use scenarios more robust to aid the subsequent derivation of security scenarios. Therefore, the prototype described in this section plays a fundamental role between steps (ii) and (iii) in our method, that is, after the creation of scenarios describing incorrect uses of an application but before the derivation of security scenarios.

The tool was prototyped as an extension of Requirements Healer. It was implemented in Python [37] using libraries such as spaCy [38], an NLP library, and textblob [39].

Requirements Healer is a web application that can be run on desktop computers as well as mobile phones. It currently manages different projects and supports different kinds of artifacts written in natural language. Scenarios are one kind of artifact, but the application can be easily extended to support other artifacts such as user stories, use cases, etc.

Our prototype supports the different steps in our approach. It provides users with an edition form where they can write about scenarios for a software application, including incorrect use scenarios. Figure 2 and Figure 3 show the forms for a correct use scenario and an incorrect use scenario, respectively. The prototype allows the user to sort the scenarios by name and, after selecting one incorrect use scenario, it performs the derivation of security scenarios by applying the mapping rules proposed here. Then, the user can edit the security scenarios to improve their description.

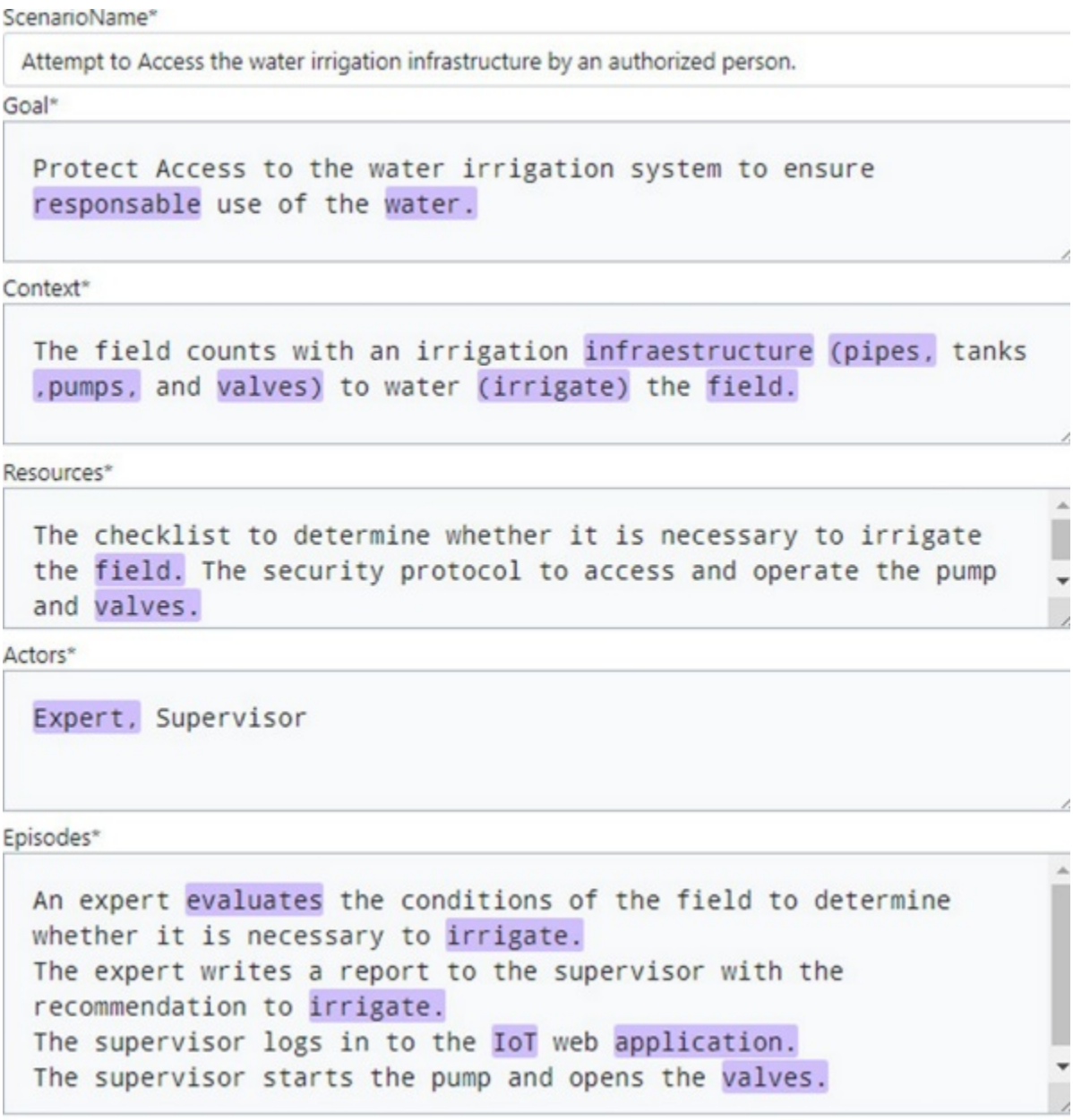

Figure 2.
Correct use scenario.
Source: Own work.

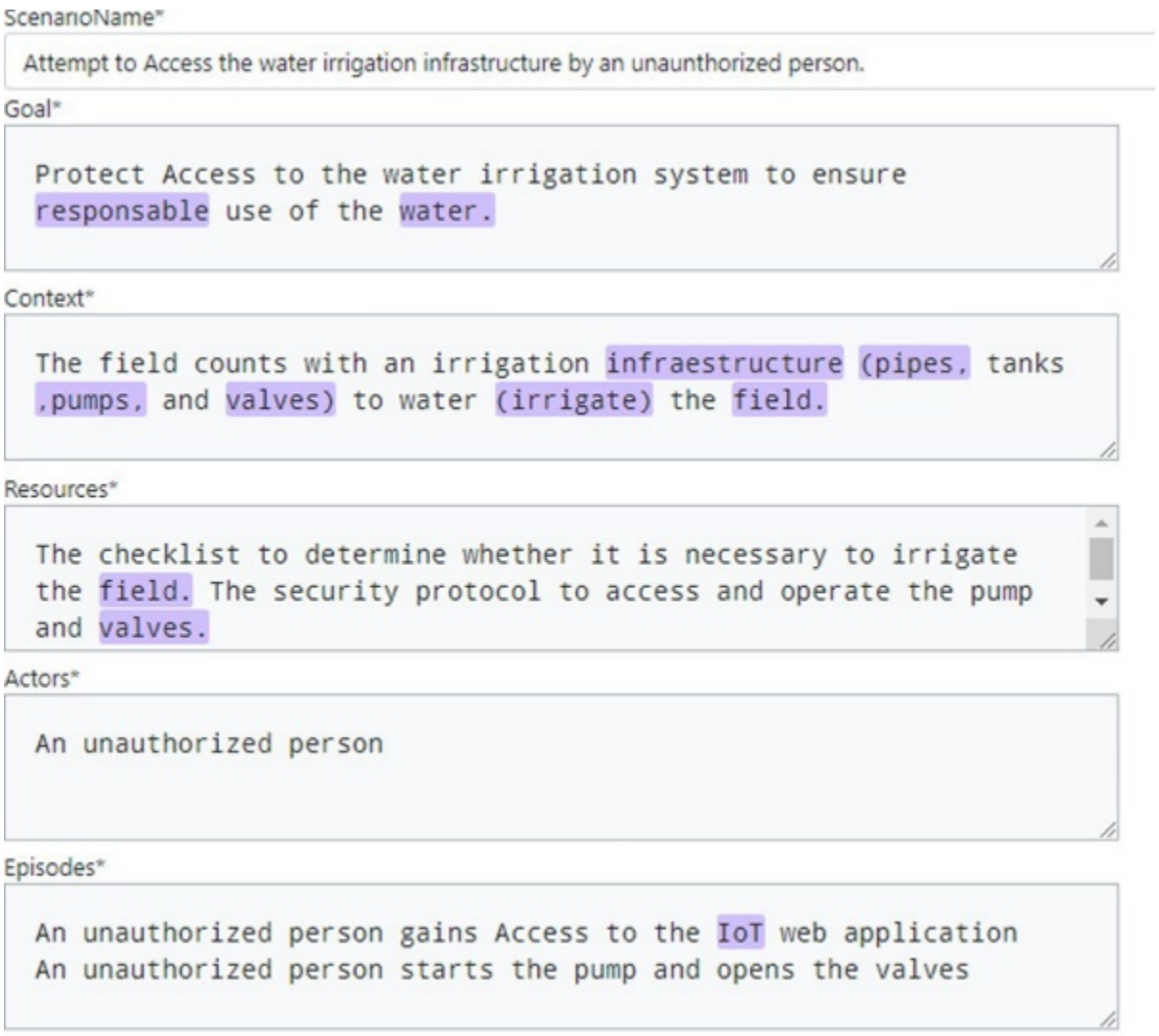

Figure 3.
Incorrect use scenario to process the keywords.
Source: Own work.

The prototype uses natural language processing tools to assist the requirements engineer in the description of security scenarios. For example, using lemmatization and stemming techniques, the prototype can verify whether certain terms or expressions listed in a glossary have been used in the scenario. Assessing the presence of certain types of expressions within the fields of an incorrect use scenario will allow us to find the most appropriate technique for coping with the issue described in the scenario. This procedure (explained in more detail below) is vital to make the incorrect use scenarios more robust.

6.1 Catalogs

The aim of the prototype is to make the information contained in a scenario more robust, using natural language processing techniques to extend the scenarios with precise information contained in catalogs that have been specifically designed for this prototype [40]. The literature was reviewed to obtain relevant information about the most common attacks that threaten IoT-based agricultural solutions [12], [20]–[22], [41]–[45]. Special attention was paid to identify vulnerabilities and specific attacks, the Quality Attribute

(QA), and the architectural layer affected by each type of attack, as well as the corresponding mechanisms to mitigate them. The information thus obtained was transferred to the two following catalogs:

a- General Security Aspects. This catalog includes general information classified under the following headings:

QA (Security): the quality attribute affected by an attack, e.g., privacy.
Example attacks: some concrete examples of the type of attack.
Consequences for the agricultural industry: a description of the impact and consequences the attack may have on the agricultural industry.
Architectural layer involved: a list of the architectural layers that may be affected by the attack.
Layer definition: a description of the affected architectural layer.
Common problems: an explanation of some common problems caused by the attack.
Resources: a list of all the resources that may be involved in the attack.

b- Specific Attacks to QAs. This catalog includes more in-depth descriptions of specific attacks classified under the following headings:

QA (Security): the quality attribute(s) that may be affected by the attack.
Threats and attacks (1): a list of concrete examples of attacks whose targets are the same as for the main attacks.
Threats and attacks (2): a list of exploits related to the type of attack.
Description: a detailed description of the attack.
Mitigation mechanism: a description of the recommended mitigation protocols or algorithms to mitigate or counter-attack the former attack.
Keywords: a list of words that best describe the attack.
Comments: comments about the specific attack, such as alternative classifications and extra information related to the attack.

These catalogs are organized as tables, each heading corresponding to a column. Each row of the General Security Aspects catalog contains information about one quality attribute (e.g., privacy, confidentiality, etc.). Each row of the Specific Attacks catalog contains information about one specific type of related attack. Figure 4 and Figure 5 show screenshots of the General Security Aspects catalog and the Specific Attacks catalog, respectively.

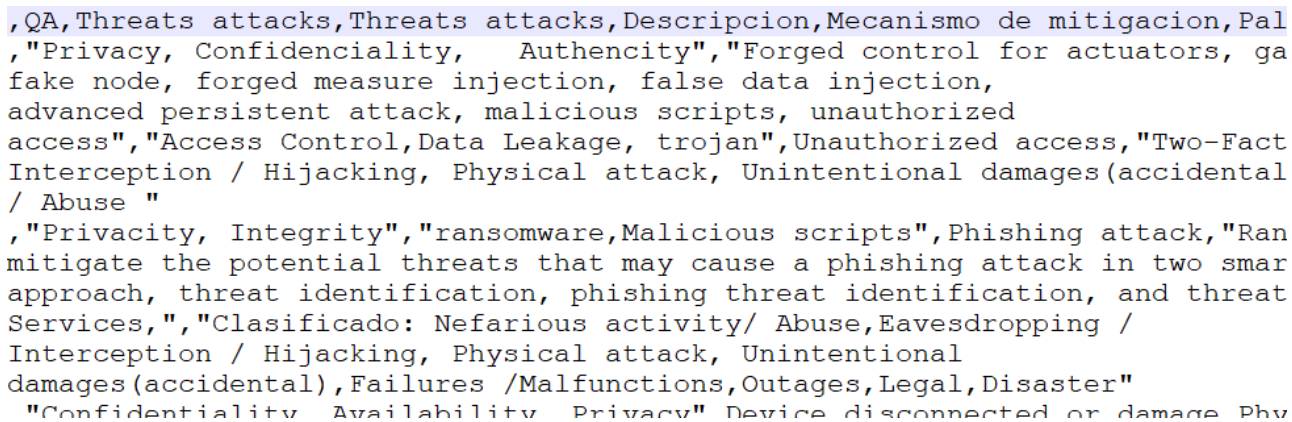

Figure 4.

General Security Aspects catalog (screenshot).

Source: Own work.

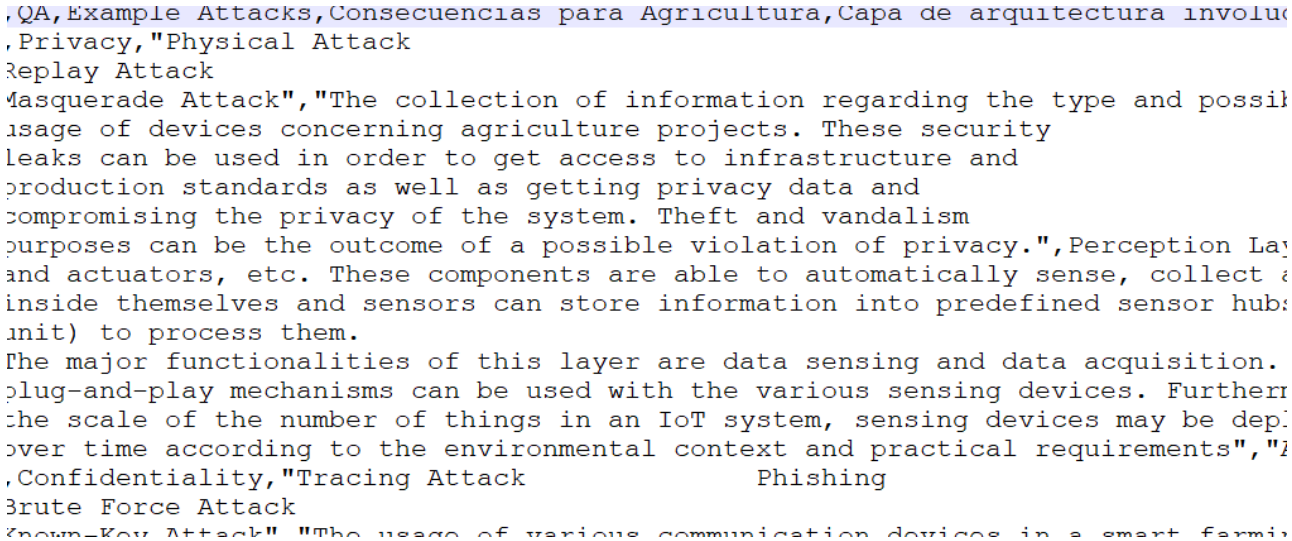

Figure 5.

Specific Attacks catalog (screenshot).

Source: Own work.

6.2 Scenario processing

The operation of our prototype can be summarized as follows. First, keywords related to specific attacks are identified in existing scenarios (both correct and incorrect use scenarios). The catalogs are then searched for the keywords in order to locate the row containing an occurrence of the specific attack. Once the relevant row is identified in both catalogs, the following information is extracted: affected security QA, attack involved, mitigation mechanism, and consequences for the agricultural industry. The algorithm concatenates the information extracted from the catalogs and attaches it to a security scenario in a new field labelled *threats*. The user can use this information to derive more robust and precise security scenarios.

We expect that our prototype will help requirements engineers to complement correct and incorrect use scenarios for software applications. The following is a detailed description of the algorithm:

Step 0: Preprocessing (Tokenization and POS tagging). This step is carried out using spaCy's open-source libraries. spaCy's pre-trained language models are used to tokenize the document and assign POS (part-of-speech) tags to each token.
Step 1: Process scenarios looking for nouns and verbs. First, the document is processed to extract all nouns and verbs (the words concentrating the most important information). The nouns and verbs are lemmatized to obtain their root form. Also, the main subject is captured.
Step 2: Process catalogs looking for ADJ + NOUN sequences. First, keywords from the catalogs are captured using an external source. Then, the catalogs in CSV format are processed using spaCy's libraries. spaCy's matcher is used to extract ADJ + NOUN sequences from the catalogs. Matched sequences are filtered. Thus, only the most important matches are kept.
Step 3: Compare the outputs of Steps 1 and 2. Using the output of Step 2, we look for specific words in the output of Step 1 that have the same syntactic structure. That is, the words extracted from a scenario are checked against the words obtained from the CSV catalogs. The number of rows in the catalog where a match is found is counted. This count is stored in tuples (row, no. of matches).
Step 4: Find the row with most matches. The tuples are processed with a max function.
Step 5: Get the relevant information from the catalogs and fill in the scenario. The relevant information is extracted from the row with most matches and tagged as follows: affected QA, threats and attacks, affected layer, layer details, mitigation mechanisms, and impact. This information is then added to a draft of the security scenario, in a field labelled *threats*. The results of this process are shown in Figure 6.

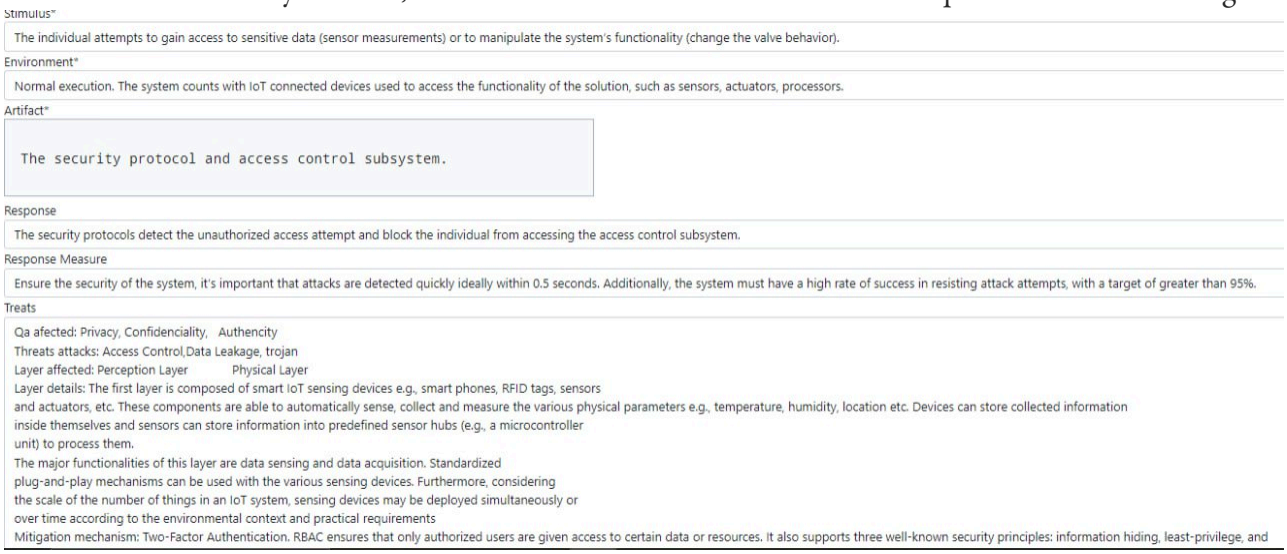

Figure 6.
App output.
Source: Own work.

The algorithm strengthens the incorrect use scenario (which is based on the correct use scenario), enabling the expert to complete the information for the analysis and subsequent derivation of the security scenario.

7. DISCUSSION

[25] proposed FRASAD, a model-driven software development framework to manage the complexity of IoT applications. [26] proposed another approach to deal with said complexity. Their approach includes activities such as requirements development, domain-specific design, verification, simulation, analysis, calibration, deployment, code generation, and execution [28] presented a taxonomy of security requirements to be considered when a software application is designed and implemented.[19] proposed a security

architecture to provide security-enabled IoT services and a baseline for security deployment. [29] established security requirements for IoT systems and focused on enhancing the security of smart home applications.[30] presented a comprehensive literature review of IoT security requirements [33] presented an approach to specify security requirements for IoT applications. They combined a framework for requirement elicitation with automated reasoning to provide secure IoT for vulnerable users in healthcare scenarios. They mapped technical system requirements using high-level logical modelling.

The proposals mentioned in the *Related Work* section enrich our proposal and complement it in different areas such as the following. (i) The architectural solution outlined in [19] plays a crucial role in this study because it addresses the security requirements of IoT systems. These security requirements are useful components of our security scenarios proposal. By focusing on these requirements, we can effectively establish a robust security framework at the requirements level. (ii) The requirements identified in [29] complement our proposal as they introduce significant terminology for describing security scenarios in the context of IoT and smart farming. by incorporating these elicited requirements, we can effectively address the specific security challenges and other considerations associated with IoT and smart farming environments.

Although the related works contributed to our research, there are general and specific differences between our study and the proposals mentioned above. (i) None of these proposals has considered security, which is our main concern [25], [26]. (ii) Some other approaches have indeed considered the security issue, but in terms of implementation, whereas our proposal considers security in terms of requirements specification. (iii) A number of approaches have considered security among software requirements, but they have not addressed how to specify security requirements precisely. (iv) [30] presented a comprehensive literature review of IoT security requirements, but they did not include any references on how to specify them. (v) [33] presented an approach to specify security requirements for IoT applications, but their work concentrated on attack tree analysis to identify the situation. In contrast, our approach focuses on how to describe security requirements precisely.

Considering the previous findings that support our proposal and the differences found in related works, we present a lightweight approach to requirement specifications that begins with a description of functional requirements. The misuse of the application is specified in order to design countermeasures to deal with it. This study also describes a prototype tool that helps to apply the proposed approach. Finally, a preliminary assessment is provided. In the survey administered to five security experts, it was found that the proposed security scenario method is generally useful for specifying agricultural IoT solutions but needs improvement in different areas.

The experts commented that the approach still needs to be more specific and interactive for users to remember its steps. They also suggested incorporating more specific and accurate cybersecurity aspects, vulnerabilities, and risks commonly found in IoT systems, as well as different types of common and malicious users. These results provided valuable feedback for refining and improving the method in order to fulfil user needs and enhance security aspects.

Currently, the most widely used software development methodology is agile development. However, we propose a different, complementary, and lightweight technique made specifically for IoT applications in smart farming. The prototype tool and the algorithm described in this paper can strengthen and refine incorrect use scenarios based on correct use scenarios, enabling experts to add more information for the analysis and subsequent derivation of the security scenario.

8. CONCLUSIONS

This paper proposed a novel approach to describing security scenarios that can be used to design robust software architectures for IoT technologies in the agricultural field. Developers of IoT applications should be concerned about security (and some other non-functional requirements) since the risk of exposing physical

artifacts to intruders is considerable in this area. Moreover, it is difficult to identify the threat and design a countermeasure. Generally, these issues are identified when it is too late, when some intruder exploits the vulnerability.

Therefore, this paper presented a lightweight approach that begins with a description of the functional requirements. The misuse of the application is then identified in order to design countermeasures to deal with it. This paper also described a prototype tool to help apply the proposed approach. The method is composed of four essential steps: (i) describe scenarios for the intended software application, (ii) describe scenarios related to the previous ones but referring to an incorrect use of the application, (iii) apply a set of rules to map attributes from the previous scenarios to the architectural scenarios, and (iv) describe the architectural scenarios in more detail. Additionally, a preliminary assessment of this method was also conducted.

The survey applied to five security experts found that the proposed security scenario method is generally useful for specifying agricultural IoT solutions but needs improvement in certain areas. Experts suggested incorporating specific cybersecurity aspects, vulnerabilities, and risks commonly found in IoT systems, as well as different types of users and adversaries. They also noted that the method needs to be more specific and interactive for users to remember its steps. The results provided valuable insights for refining and improving the method in order to meet user needs and enhance security.

Currently, the most widely used software development methodology is agile development. However, we propose a complementary and lightweight technique specifically for IoT applications in smart farming. In future studies, we aim to enrich our proposal with additional guidelines for writing scenarios at each stage. Additionally, further experimentation is necessary before we make the approach more complex. Nevertheless, we firmly believe that our approach can be improved and made more robust.

The tool and this algorithm can strengthen incorrect use scenarios (which are based on correct use scenarios), enabling experts to complete the information for the analysis and subsequent derivation of security scenarios.

References

[1] ITU-T. "Overview of internet of things." 2012. [Online]. Available: https://www.itu.int/rec/T-REC-Y.2060/en

[2] K. Ojo-Gonzalez, and B. Bonilla-Morales, "Requerimientos no funcionales para sistemas basados en el internet de las cosas (IoT): Una revisión," *I+D Tecnológico*, vol. 17, no. 2, Jul. 2021. https://doi.org/10.33412/idt.v17.2.3303

[3] Berkeley CPS Publications. "*Cyber-Physical Systems (CPS).*" Berkeley.edu. Accessed: Sep. 20, 2023. [Online]. Available: https://ptolemy.berkeley.edu/projects/cps/

[4] P. Shankar, B. Morkos, D. Yadav, and J. D. Summers, "Towards the formalization of non-functional requirements in conceptual design," *Res. Eng. Des.*, vol. 31, no. 4, pp. 449–469, Oct. 2020. https://doi.org/10.1007/s00163-020-00345-6

[5] E. Serna M., and A. Serna A., "Process and progress of requirement formalization in software engineering," Ingeniare, Rev. Chil. Ing., vol. 28, no. 3, pp. 411–423, Sep. 2020. https://doi.org/10.4067/S0718-33052020000300411

[6] U. Ahmed, "A review on khowledge management in requirements engineering," in *International Conference on Engineering and Emerging Technologies (ICEET), Lahore, Pakistan*, 2018, pp. 1-5. https://doi.org/10.1109/ICEET1.2018.8338650

[7] C. Potts, "Using schematic scenarios to understand user needs," in *Proceedings of the conference on Designing interactive systems processes, practices, methods, & techniques - DIS '95*, New York, Aug. 1995, pp. 247–256. https://doi.org/10.1145/225434.225462

[8] J. Patton, and P. *Economy, User Story Mapping: Discover the Whole Story, Build the Right Product*, 1st Ed. Sebastopol, CA, United States of America: O'Reilly Media, 2014.

[9] J. R. Price, *Write a Use Case: Gathering Requirements that Users Understand, The Communication Circle*, 2020.

[10] J. M. Carroll, "Five reasons for scenario-based design," in *Proceedings of the 32nd Annual Hawaii International Conference on Systems Sciences. 1999. HICSS-32. Abstracts and CD-ROM of Full Papers*, Maui, HI, USA, Jan. 1999, pp. 11. https://doi.org/10.1109/hicss.1999.772890

[11] S. Hofer, and H. Schwentner, *Domain Storytelling: A Collaborative, Visual, and Agile Way to Build Domain-Driven Software (Addison-Wesley Signature Series (Vernon))*, 1st Ed. Massachusetts, United States Of America: Addison-Wesley Professional, 2021.

[12] S. Pal, M. Hitchens, T. Rabehaja, and S. Mukhopadhyay, "Security requirements for the internet of things: A systematic approach," *Sensors*, vol. 20, no. 20, p. 5897, Oct. 2020. https://doi.org/10.3390/s20205897

[13] S. Myagmar, A. J. Lee, and W. Yurcik, "Threat Modeling as a Basis for Security Requirements," *ResearchGate*, Aug. 2005. [Online]. Available: https://www.researchgate.net/publication/228634178_Threat_Modeling_as_a_Basis_for_Security_Requirements

[14] B. Schneier, "Cryptography Is Harder than It Looks," *IEEE Secur. Priv.*, vol. 14, no. 1, pp. 87–88, Jan.-Feb. 2016. https://doi.org/10.1109/MSP.2016.7

[15] T. Martin, D. Geneiatakis, I. Kounelis, S. Kerckhof, and I. N. Fovino, "Towards a formal lot security model," *Symmetry*, vol. 12, no. 8, p. 1305, Aug. 2020.https://doi.org/10.3390/sym12081305

[16] M. Dhanaraju, P. Chenniappan, K. Ramalingam, S. Pazhanivelan, and R. Kaliaperumal, "Smart Farming: Internet of Things (IoT)-Based Sustainable Agriculture," *Agriculture*, vol. 12, no. 10, p. 1745, Oct. 2022. https://doi.org/10.3390/agriculture12101745

[17] N. Khan, R. L. Ray, G. R. Sargani, M. Ihtisham, M. Khayyam, and S. Ismail, "Current progress and future prospects of agriculture technology: Gateway to sustainable agriculture," *Sustainability*, vol. 13, no. 9, p. 4883, Apr. 2021. https://doi.org/10.3390/su13094883

[18] D. C. Rose, R. Wheeler, M. Winter, M. Lobley, and C. Charlotte-Anne, "Agriculture 4.0: Making it work for people, production, and the planet," *Land use policy*, vol. 100, p. 104933, Jan. 2021. https://doi.org/10.1016/j.landusepol.2020.104933

[19] S. El-Gendy, and M. A. Azer, "Security Framework for Internet of Things (IoT)," in *2020 15th International Conference on Computer Engineering and Systems (ICCES), Cairo, Egypt*, 2020, pp. 1-6. https://doi.org/10.1109/ICCES51560.2020.9334589

[20] A. Rettore de Araujo Zanella, E. da Silva, and L. C. Pessoa Albini, "Security challenges to smart agriculture: Current state, key issues, and future directions," *Array*, vol. 8, p. 100048, Dec. 2020. https://doi.org/10.1016/j.array.2020.100048

[21] A. Yazdinejad et al., "A review on security of smart farming and precision agriculture: Security aspects, attacks, threats and countermeasures," *Applied Sciences*, vol. 11, no. 16, Aug. 2021. https://doi.org/10.3390/app11167518

[22] K. Demestichas, N. Peppes, and T. Alexakis, "Survey on Security Threats in Agricultural IoT and Smart Farming," sensors, vol. 20, no. 22, p. 6458, Nov. 2020. https://doi.org/10.3390/s20226458

[23] J. C. Sampaio Do Prado Leite, G. D. S. Hadad, J. H. Doorn, and G. N. Kaplan, "A scenario construction process," *Requir. Eng.*, vol. 5, no. 1, pp. 38–61, Jul. 2000. https://doi.org/10.1007/pl00010342

[24] S. Khamaiseh, and D. Xu, "Software security testing via misuse case modeling," in *2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress*, *Orlando, FL, USA, 2017*, pp. 534-541. https://doi.org/10.1109/DASC-PICom-DataCom-CyberSciTec.2017.98

[25] X. T. Nguyen, H. T. Tran, H. Baraki, and K. Geihs, "Frasad: A Framework for Model-driven IoT Application Development Xuan," in *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT), Milan, Italy, 2015*, pp. 387-392. https://doi.org/10.1109/WF-IoT.2015.7389085

[26] B. Karaduman, S. Mustafiz, and M. Challenger, "FTG+PM for the Model-Driven Development of Wireless Sensor Network based IoT Systems," in *2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C), Fukuoka, Japan, 2021*, pp. 306-316. https://doi.org/10.1109/MODELS-C53483.2021.00052

[27] H. Cardenas, R. Zimmerman, A. R. Viesca, M. Al Lail, and A. J. Perez, "Formal UML-based Modeling and Analysis for Securing Location-based IoT Applications," in *2022 IEEE 19th International Conference on Mobile Ad Hoc and Smart Systems (MASS), Denver, CO, USA, 2022*, pp. 722-723. https://doi.org/10.1109/MASS56207.2022.00109

[28] K. Slovenec, M. Vuković, D. Salopek, and M. Mikuc, "Securing IoT Services Based on Security Requirement Categories," in *2022 International Conference on Software, Telecommunications and Computer Networks (SoftCOM), Split, Croatia, 2022*, pp. 1-6. https://doi.org/10.23919/SoftCOM55329.2022.9911319

[29] S. Sotoudeh, S. Hashemi, and H. G. Garakani, Security Framework of IoT-Based Smart Home," in *2020 10th International Symposium on Telecommunications (IST), Tehran, Iran, 2020*, pp. 251-256. https://doi.org/10.1109/IST50524.2020.9345886

[30] W. Iqbal, H. Abbas, M. Daneshmand, B. Rauf, and Y. A. Bangash, "An In-Depth Analysis of IoT Security Requirements, Challenges, and Their Countermeasures via Software-Defined Security," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 10250–10276, Oct. 2020. https://doi.org/10.1109/JIOT.2020.2997651

[31] Ö. Özkaya, and B. Örs, "Model based node design methodology for secure IoT applications," in *2018 26th Signal Processing and Communications Applications Conference (SIU), Izmir, Turkey*, 2018, pp. 1-4. https://doi.org/10.1109/SIU.2018.8404490

[32] R. M. Carvalho, "Dealing with Conflicts Between Non-functional Requirements of UbiComp and IoT Applications," in *2017 IEEE 25th International Requirements Engineering Conference (RE), Lisbon, Portugal, 2017*, pp. 544-549. https://doi.org/10.1109/RE.2017.51

[33] F. Kammuller, J. C. Augusto, and S. Jones, "Security and privacy requirements engineering for human centric IoT systems using eFRIEND and Isabelle," in *2017 IEEE 15th International Conference on Software Engineering Research, Management and Applications (SERA), London, UK, 2017*, pp. 401-406. https://doi.org/10.1109/SERA.2017.7965758

[34] M. Gupta, M. Abdelsalam, S. Khorsandroo, and S. Mittal, "Security and Privacy in Smart Farming: Challenges and Opportunities," *IEEE Access*, vol. 8, pp. 34564–34584, Feb. 2020. https://doi.org/10.1109/ACCESS.2020.2975142

[35] F. Davis, "*User Acceptance of Information Systems: Technology acceptance model (TAM)*," University of Michigan, Ann Arbor, Michigan. [Online]. Available: https://deepblue.lib.umich.edu/bitstream/handle/2027.42/35547/b1409190.0001.001.pdf?seque

[36] N. Marangunić, and A. Granić, "Technology acceptance model: a literature review from 1986 to 2013," Univers. Access Inf. Soc., vol. 14, pp. 81–95, Mar. 2015. https://doi.org/10.1007/s10209-014-0348-1

[37] Python. (1995). *Netherlands*. Accessed: Sep. 20, 2023. [Online]. Available: https://www.python.org/

[38] Spacy. *Industrial-Strength Natural Language Processing*. (2016). Accessed: Sep. 20, 2023. [Online]. Available: https://spacy.io/

[39] S. Loria. Textblob (Python). (2023). Accessed: Sep. 23, 2023. [Online]. Available: https://pypi.org/project/textblob/

[40] S. Aurangzeb, M. Aleem, M. Azhar Iqbal, and M. Arshad Islam, "Ransomware: A Survey and Trends," *Journal of Information Assurance and Security*, vol. 12, Jun. 2017. https://www.researchgate.net/publication/317380115_Ransomware_A_Survey_and_Trends

[41] S. G. Abbas et al., "Identifying and mitigating phishing attack threats in IoT use cases using a threat modelling approach," *Sensors*, vol. 21, no. 14, p. 4816, Jul. 2021. https://doi.org/10.3390/s21144816

[42] L. Chang, "*A Proactive Approach to Detect IoT Based Flooding Attacks by Using Software Defined Networks and Manufacturer Usage Descriptions,*" M.S thesis, Arizona State University Tempe Campus, EE. UU. 2018. [Online]. Available: https://core.ac.uk/download/pdf/161995314.pdf

[43] J. Liu, Y. Xiao, and C. L. P. Chen, "Authentication and Access Control in the Internet of Things," in *2012 32nd International Conference on Distributed Computing Systems Workshops, Macau, China*, 2012, pp. 588-592. https://doi.org/10.1109/ICDCSW.2012.23

[44] Q. M. Ashraf, and M. H. Habaebi, "Autonomic schemes for threat mitigation in Internet of Things," *J. Netw. Comput. Appl.*, vol. 49, pp. 112–127, 2015. https://doi.org/10.1016/j.jnca.2014.11.011

[45] J. Deogirikar, and A. Vidhate, "Security attacks in IoT: A survey," in *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), Palladam, India, 2017*, pp. 32-37. https://doi.org/10.1109/I-SMAC.2017.8058363

[46] Decisioning, "The second workshop on Collaboration in knowledge discovery and decision making." *unicauca.edu.co*. Accessed: Sep. 23, 2023. [Online]. Available: https://www.unicauca.edu.co/versionP/eventos/conversatorio/decisioning-2023-second-workshop-collaboration-knowledge-discovery-and-decision-making

Notes

9. ACKNOWLEDGEMENT AND FUNDING :
This study was partially funded by the STIC AmSud program (project code 22STIC-01). All authors declare that they have no conflicts of interest.

CONFLICTS OF INTEREST :
. The authors declare that there is no conflict of interest.

AUTHOR CONTRIBUTIONS :
- **Julio Ariel Hurtado:** Conceptualization, Supervision.
- **Leandro Antonelli:** Conceptualization, Supervision.
- **Santiago López:** Methodology, Investigation, Resources, Writing - Review and Editing, Validation.
- **Adriana Gómez:** Methodology, Investigation, Resources; Writing - Review and Editing, Validation.
- **Juliana Delle Ville:** Methodology, Investigation, Resources, Writing - Review and Editing, Validation.
- **Giuliana Maltempo:** Methodology, Investigation, Resources, Writing - Review and Editing, Validation.
- **Frey Giovanny Zambrano:** Validation, Writing - Review and Editing.
- **Andrés Solis:** Investigation, Writing - Review and Editing.
- **Marta Cecilia Camacho:** Validation, Writing - Review and Editing.
- **Miguel Solinas:** Writing - Review and Editing.
- **Gladys Kaplan:** Writing - Review and Editing.
- **Freddy Muñoz:** Investigation, Writing - Review and Editing

Información adicional

How to cite / Cómo citar: J.A Hurtado et al., "Semiotics: An Approach to Model Security Scenarios for IoT-Based Agriculture Software," *Tecnológicas*, vol. 27, no. 59, e2923, Apr. 2024. https://doi.org/10.22430/22565337.2923

Enlace alternativo

https://revistas.itm.edu.co/index.php/tecnologicas/issue/view/135 (html)

Julio Ariel Hurtado, Leandro Antonelli, Santiago López, Adriana Gómez, Juliana Delle Ville, Giuliana Maltempo, Frey Giovanny Zambrano, Andrés Solis, Marta Cecilia Camacho, Miguel Solinas, Gladys Kaplan, Freddy Muñoz

Semiotics: An Approach to Model Security Scenarios for IoT-Based Agriculture Software

Semiótica: un enfoque para modelar escenarios de seguridad para software de agricultura basado en IoT

TecnoLógicas
vol. 27, núm. 59, p. 1 - 21, 2024
Instituto Tecnológico Metropolitano, Colombia
tecnologicas@itm.edu.co

ISSN: 0123-7799 **/ ISSN-E:** 2256-5337

DOI: https://doi.org/10.22430/22565337.2923