UDC 004.6

VITALII MISTRIAKOV, PAN TIANDE

PROCESSING CONTENT QUERY REQUESTS FOR CSAF DOCUMENTS USING A GRAPHQL-BASED API

West Ukrainian National University, Ternopil, 46009, Ukraine

Анотація. Рекомендації щодо безпеки є важливим ресурсом для осіб, які відповідають за безпеку, оскільки вони пропонують інформацію про вразливі місця та ІТ-системи та компоненти програмного забезпечення, які зазнали впливу. Ці рекомендації випускають виробники, експерти з ІТ-безпеки або координаційні організації, щоб допомогти користувачам зрозуміти вразливі місця та вжити заходів для їх усунення або зменшення. Відсутність стандартизованого формату в різних виробників ускладнює роботу з порадами щодо безпеки для спеціалістів із ІТ-безпеки. Для вирішення цієї проблеми було запроваджено Загальну консультативну структуру безпеки (CSAF), яка надає стандартизований формат порад щодо безпеки. Отже, деякі поради тепер пропонуються у форматі CSAF. Збільшення кількості документів CSAF впливає на час обробки цих документів фахівцями з ІТ-безпеки. Щоб підвищити ефективність обробки цих запитів, у цьому документі пропонується зосередитися на документах, що містять певний вміст, а не на обробці всіх документів. Для цього пропонується використовувати GraphQL, мову запитів даних з відкритим кодом, яка дозволяє ефективно формалізувати запити документів CSAF. Ця стаття спрямована на впровадження API на основі GraphQL для підвищення ефективності обробки запитів документів CSAF.

Ключові слова: Common Security Advisory Framework, формат CSAF, API, GraphQL, відповідна документація

Abstract. Security advisories serve as an essential resource for individuals tasked with ensuring safety, as they offer information about vulnerabilities and the impacted IT systems and software components. These advisories are released by manufacturers, IT security experts, or coordinating organizations to assist users in comprehending vulnerabilities and taking steps to either eliminate or reduce them. The lack of a standardized format across different manufacturers has made working with security advisories more complex for IT security professionals. The Common Security Advisory Framework (CSAF) was introduced to address this issue, providing a standardized format for security advisories. Consequently, certain advisories are now offered in the CSAF format. The increase in the number of CSAF documents affects the processing time of these documents by IT security specialists.

To improve the efficiency of processing these queries, this paper proposes to focus on documents containing specific content, rather than processing all documents. To do this, it proposes to use GraphQL, an open source data query language that allows for efficient formalization of CSAF document queries. This paper aims to implement a GraphQL-based API to improve the efficiency of processing CSAF document queries.

Keywords: Common Security Advisory Framework, CSAF format, API, GraphQL, relevant documentation

DOI: 10.31649/1681-7893-2024-48-2-152-161

INTRODUCTION

IT systems and software components often contain vulnerabilities, and identifying these vulnerabilities and mitigating them is a primary challenge for IT security professionals [1,2].

Security advisories serve as an essential resource for individuals tasked with ensuring safety, as they offer information about vulnerabilities and the impacted IT systems and software components. These advisories are released by manufacturers, IT security experts, or coordinating organizations to assist users in comprehending vulnerabilities and taking steps to either eliminate or reduce them [3].

The lack of a standardized format across different manufacturers has made working with security advisories more complex for IT security professionals. The Common Security Advisory Framework (CSAF) was introduced to address this issue, providing a standardized format for security advisories. Consequently, certain advisories are now offered in the CSAF format. However, with the increase in the number of advisories formatted in CSAF, IT security professionals are now required to sift through a much larger quantity of advisories to assess their applicability. Relevant advisories are those that relate to the IT systems or software components for which they are responsible [4,5].

To assess relevance, it is essential to query and examine each security advisory. The effectiveness of these queries could be enhanced by focusing on documents containing particular content instead of querying all documents indiscriminately. GraphQL, a data query language that is open-source, might provide a more efficient method for querying CSAF documents. This research is aimed at implementing a GraphQL-based API to improve the efficiency of processing requests for CSAF documents [5,6].

PROBLEM STATEMENT

Security advisories offer an overview of vulnerabilities and the affected products. Security advisories are released by different manufacturers for their products or by IT security experts. The Common Security Advisory Framework (CSAF) offers an organized method for these advisories, outlining specific roles, document formats, and ways to distribute the information.

The CSAF standard offers two distribution methods: one based on directories and the other on ROLIE.However, neither method allows for direct searching of CSAF documents by relevant content, such as specific products. Instead, all documents must be downloaded to determine if a product is included and if it is impacted by a vulnerability with a significant rating. This process needs to be conducted repeatedly for all new and updated CSAF documents, which is both time-intensive and not readily automatable.

This method is inefficient and consumes valuable time and resource, Consequently, this paper's aims to reduce the time interval between the release of a security advisory identifying affected products, and getting this information to the appropriate people. By reducing this time, valuable human resources can be redirected from searching for relevant information to taking action. To achieve this, an API is required that enables querying for relevant security advisories. Ideally, it should allow for the bulk download of all relevant advisories for all products in use. Previous work on a CSAF API serves as a foundation for this development.

The enhanced CSAF API query functionality will empower users to search for specific components and content within CSAF documents, as well as for properties linked to dependent content. For instance, it will facilitate queries based on product names to identify products whose product IDs are referenced in the product status of a vulnerability.. This streamlines the CSAF consumption process by eliminating an extra step, thus speeding up the workflow.

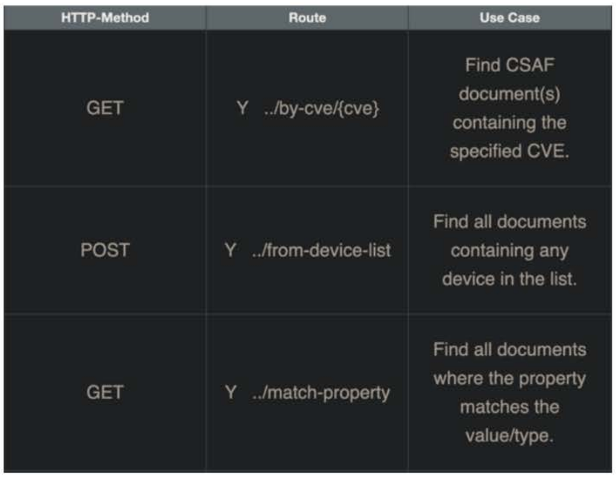
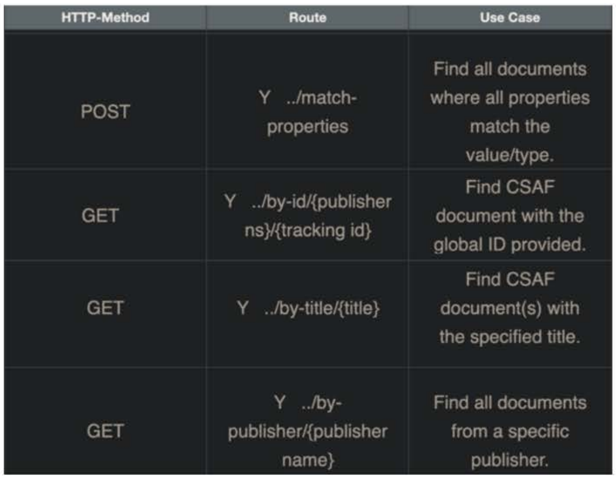
To effectively manage the large volume of information, filter parameters will be implemented to refine the results. Logical operators such as AND, OR, XOR, and NOT will facilitate more precise queries, enabling users to combine or exclude criteria to align with specific requirements. Furthermore, the API will provide functionality to verify the presence or absence of CSAF document components. This feature opens up new possibilities for quality management, such as querying for missing components (e.g., */product_ull_product_names[]/product_ identification_helper*). Achieving these capabilities will require a well-structured approach.

RELATED WORKS

Existing solutions have explored API design for querying CSAF documents, including various REST-based approaches and their limitations. REST APIs often face constraints due to their fixed routes and use cases (see Table 1). For example, retrieving CSAF documents by CVE or device list requires separate queries. Combining routes, such as querying for documents that match either one of two criteria, necessitates separate requests and leads to the challenge of handling duplicates. This issue, known as the union problem, occurs because documents may be retrieved more than once if multiple routes are queried. Complex queries involving logical operators like XOR are also challenging with REST APIs [5,6].

Table 1

REST API limitations

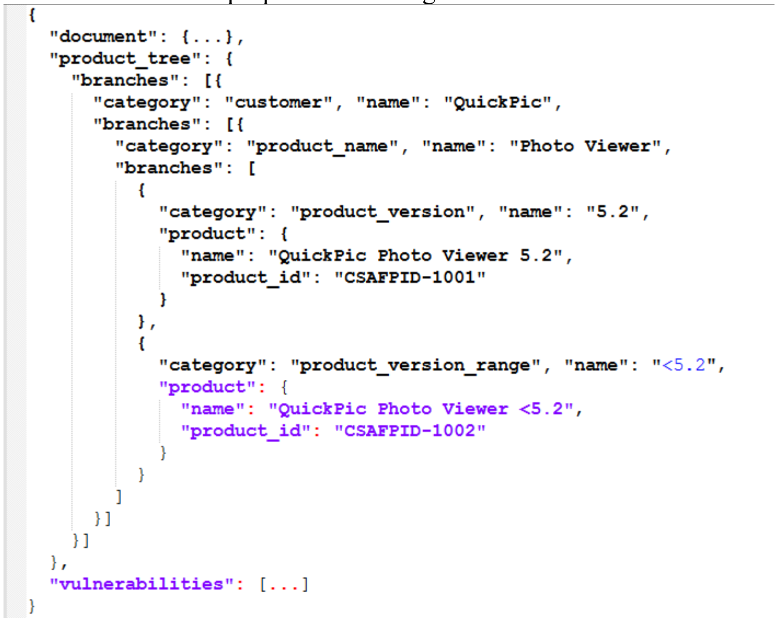



HTTP-Method	Route	Use Case
GET	Y ../by-cve/{cve}	Find CSAF document(s) containing the specified CVE.
POST	Y ../from-device-list	Find all documents containing any device in the list.
GET	Y ../match-property	Find all documents where the property matches the value/type.
POST	Y ../match-properties	Find all documents where all properties match the value/type.
GET	Y ../by-id/{publisher ns}/{tracking id}	Find CSAF document with the global ID provided.
GET	Y ../by-title/{title}	Find CSAF document(s) with the specified title.
GET	Y ../by-publisher/{publisher name}	Find all documents from a specific publisher.

Current REST-based solutions lack a feature to track updates to CSAF documents. Consumers must manually compare different versions of documents to identify changes. As highlighted by existing work, the absence of versioning support limits the ability to track document updates effectively [3-5].

Another challenge is the lack of a unique product identifier across CSAF documents. The attribute `/ products / names/full/product` offers a unique ID solely within the scope of an individual CSAF document. This limitation prevents effective cross-publisher searches. While the product name could be used for searching across publishers, variations in naming conventions and hierarchical structures further complicate this process. Full product names might differ based on the CSAF publisher's conventions, and the product ID can change over the document’s lifecycle.

Hierarchical structures in product trees, such as those shown in Fig 1, provide some relief. For example, product names and versions are included in a structured format. However, variations in hierarchical depth and naming conventions must be considered in the proposed API design.



```
{
  "document": {...},
  "product_tree": {
    "branches": [{
      "category": "customer", "name": "QuickPic",
      "branches": [{
        "category": "product_name", "name": "Photo Viewer",
        "branches": [
          {
            "category": "product_version", "name": "5.2",
            "product": {
              "name": "QuickPic Photo Viewer 5.2",
              "product_id": "CSAFPID-1001"
            }
          },
          {
            "category": "product_version_range", "name": "<5.2",
            "product": {
              "name": "QuickPic Photo Viewer <5.2",
              "product_id": "CSAFPID-1002"
            }
          }
        ]
      }]
    }]
  },
  "vulnerabilities": [...]
}
```

Figure 1 – Hierarchical structures in product trees

FEATURES OF CSAF API IMPLEMENTATION

To grasp the necessity of CSAF and, specifically, a CSAF API, it's important to understand the background. The subsequent sections will elucidate the definition of a security advisory, its components, and the process by which it evolves into a Common Security Advisory Framework (CSAF) document. It is essential to discuss the various roles involved in the CSAF documentation process, emphasizing that this process begins with the issuance of a security advisory and culminates in the remediation or mitigation of the identified vulnerability.

Considering that an earlier concept for the CSAF API exists, it is imperative to review it in order to incorporate the lessons learned. To ensure the secure operation of the API, it is crucial to avoid common vulnerabilities, necessitating a discussion of the Open Web Application Security Project (OWASP). Vulnerabilities constitute one of the three fundamental elements of a CSAF document; thus, this discussion will commence with an examination of vulnerabilities.

A weakness is classified as a vulnerability only when it can be demonstrably exploited. Upon the discovery of a vulnerability, it is recorded in the Common Enumeration of Vulnerabilities Program database and assigned a Common Vulnerabilities and Exposures (CVE) identifier. These CVE identifiers can be categorized under a Common Weakness Enumeration (CWE) classification. Vulnerabilities are pertinent when they affect a product within the organization's scope of responsibility or when they impact components of the organization's own products.

When a vulnerability in a third-party software component is identified, manufacturers incorporating that component into their products must assess whether their products are similarly affected. If it is determined that their products are impacted and a solution or mitigating measures are available, these manufacturers will issue a security advisory, detailing the affected products. The product tree serves as the second core element of a CSAF document.

It is important to note that not all products from a manufacturer are typically affected by a vulnerability to the same degree. Therefore, CSAF allows manufacturers to specify the product status for each vulnerability within the CSAF document. The CSAF standard includes various product status categories (see Table 2). For example, the status "not_affected" enables a manufacturer to indicate which products are not impacted by the vulnerability, thereby reducing unnecessary inquiries from customers. While this "not_affected" status may not seem dramatic, it provides reassurance to IT security professionals. Vulnerabilities and the affected products are crucial components of any security advisory.

Here's the information presented as a table, rewritten in a similar style, Table 2.

Table 2

CSAF standard with various product status category

CSAF Product Status	Description
First Affected	These represent the earliest versions of releases identified as being impacted by the vulnerability.
First Fixed	These versions contain the initial resolution for the vulnerability, although they may not be the versions recommended as fixes.
Fixed	These versions include a remedy for the vulnerability, yet they might not be the recommended versions for resolution.
Known Affected	These versions have been confirmed to be impacted by the vulnerability.
Known Not Affected	These versions have been verified as unaffected by the vulnerability.
Last Affected	These are the final versions within a release train identified as impacted by the vulnerability, with subsequent versions containing a fix.
Recommended	These versions include a resolution for the vulnerability and are endorsed by the vendor as the appropriate means to address the issue.
Under Investigation	It has not yet been established whether these versions are affected by the vulnerability; however, an investigation is currently in progress, with findings to be shared in a forthcoming document release.

Security advisories are disseminated not only by manufacturers but also by IT security researchers and coordinating entities such as the Federal Office for Information Security (BSI). These advisories encompass details regarding vulnerabilities or groups of vulnerabilities, a catalog of affected products, and pertinent information about the advisory document itself and its issuer. This constitutes the third fundamental element of a CSAF document. Ideally, security advisories offer a definitive solution or, at the very least, mitigation strategies, thereby facilitating IT security professionals in comprehending the issue, recommending appropriate actions, or substantiating the necessity for intervention. The CSAF security advisory consolidates all this information into a CSAF document. If new research provides additional insights, the advisory is updated. Similarly, if the severity of a vulnerability changes, the advisory is revised accordingly.

In addition to the CVE identifier, a security advisory may encompass various details regarding the vulnerability, including a rating and a concise description. The concise description is designed for the end-user, who assesses its significance. When a rating is included, its relevance has been evaluated in an external context. Nonetheless, the end-user must ascertain whether this context and the corresponding rating are applicable to their specific circumstances. Despite this, the external assessment enables the end-user to pre-filter for potentially more critical vulnerabilities.

Security advisories are generated not only by manufacturers but also by IT security researchers and coordinating bodies, such as the Federal Office for Information Security (BSI). These advisories contain information regarding vulnerabilities or groups of vulnerabilities, a list of affected products, and details about the advisory document itself and its issuer. This constitutes the third essential component of a CSAF document. Ideally, security advisories offer a definitive solution or, at the very least, mitigation strategies, thereby facilitating IT security professionals in understanding the issue, recommending appropriate actions, or justifying the necessity for intervention. The CSAF security advisory compiles all this information into a comprehensive CSAF document. Should new research yield further insights, the advisory will be updated accordingly. Likewise, if the severity of a vulnerability shifts, the advisory will be modified to reflect this change.

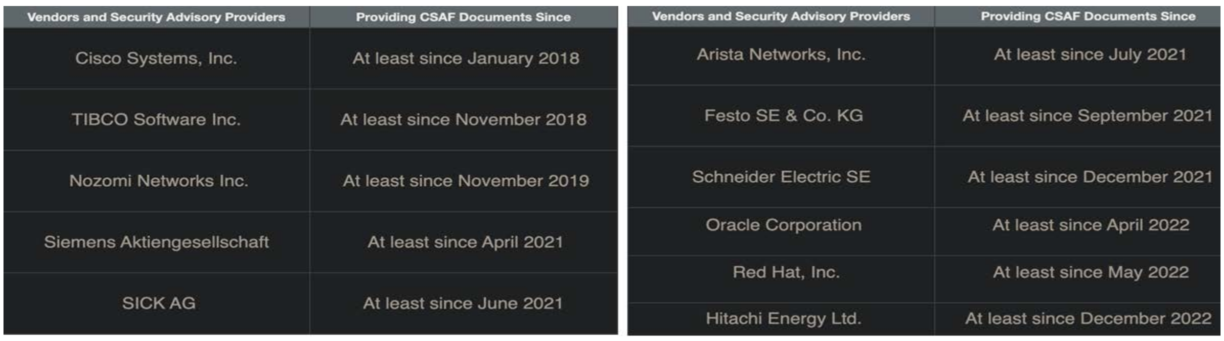
COMMON SECURITY ADVISORY FRAMEWORK

A CSAF document comprises a product tree, details concerning vulnerabilities, information about the publisher and the document, and ideally, a vendor fix. An increasing number of vendors and other security advisory providers are presenting their security advisories in the form of CSAF documents (Table 3).

CSAF enhances the security advisory landscape by offering a comprehensive framework. It delineates specific roles, including CSAF publisher, CSAF provider, CSAF trusted provider, CSAF lister, and CSAF aggregator, along with defined distribution methods such as ROLIE and directory-based approaches. Additionally, it standardizes the structure of the security advisory using JavaScript Object Notation (JSON), encompassing components such as /document, /product_tree, and /vulnerabilities.

Table 3

Providers that offer CSAF documents



Vendors and Security Advisory Providers	Providing CSAF Documents Since
Cisco Systems, Inc.	At least since January 2018
TIBCO Software Inc.	At least since November 2018
Nozomi Networks Inc.	At least since November 2019
Siemens Aktiengesellschaft	At least since April 2021
SICK AG	At least since June 2021
Arista Networks, Inc.	At least since July 2021
Festo SE & Co. KG	At least since September 2021
Schneider Electric SE	At least since December 2021
Oracle Corporation	At least since April 2022
Red Hat, Inc.	At least since May 2022
Hitachi Energy Ltd.	At least since December 2022

The CSAF consumer represents the end user or client who engages with CSAF documents. These documents may originate from various sources, including IT security researchers acting as CSAF publishers or product manufacturers serving as CSAF providers. The CSAF lister functions as a directory, cataloging CSAF providers and publishers, but does not supply the documents directly.

Conversely, the CSAF aggregator assumes a distinct role by collecting documents from multiple CSAF publishers and providers, subsequently making them available for download, contingent upon the consent of the CSAF publishers and providers. This consent is explicitly conveyed through the /mirror_on_CSAF_aggregators attribute in the provider-metadata.json, which authorizes the aggregation of CSAF documents.

CSAF document comprises three primary components: the document itself, the product tree, and the vulnerabilities.

The planned API will enable users to conduct searches within the content of these documents., making it essential to understand the relevant sections of the CSAF document, which will be detailed in the following subsections.

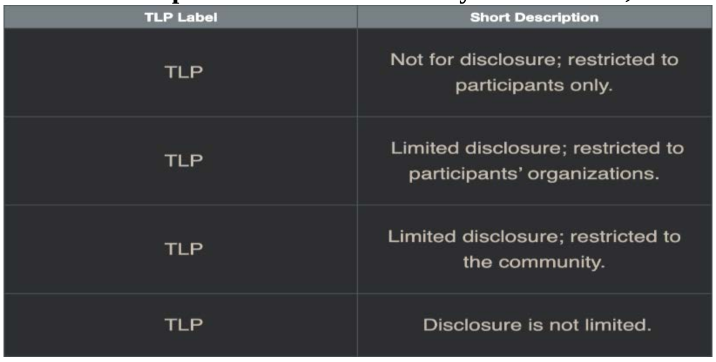
The document section of a CSAF document contains metadata pertaining to the publisher. This metadata is easily identifiable when the document is sourced directly from the CSAF publisher or provider. However, when the CSAF document is retrieved via a CSAF aggregator, this metadata becomes increasingly significant as a search parameter within the CSAF API.

Agside publisher information, the canonical URL located at /document/references/url allows the CSAF consumer to access the original source of the information. Direct access to first-hand information can provide more comprehensive and timely updates.

The title of the document acts as a useful search parameter since it is typically distinctive. The Traffic Light Protocol (TLP) label is likewise useful for both searching and sharing. There are various interpretations of the TLP, and the CSAF documents issued by the Federal Office for Information Security in Germany (BSI) follow the TLP definitions established by FIRST.ORG. As per this definition, only TLP labeled documents can be shared without restriction (see Table 4 for short descriptions).

Table 4

Short Descriptions of TLPv1 Labels by FIRST.ORG, Inc.



TLP Label	Short Description
TLP	Not for disclosure; restricted to participants only.
TLP	Limited disclosure; restricted to participants’ organizations.
TLP	Limited disclosure; restricted to the community.
TLP	Disclosure is not limited.

The document/timeline/release_date/current is particularly useful for automation, especially when dealing with the most recent documents. Interfaces designed to fetch the interfaces developed for the automatic retrieval of the latest documents will utilize this attribute for querying (Figure 2).

Each CSAF document is assigned a unique ID by the publisher, which can be located in the */document/tracking/id attribute*. This ID is unique within the context of the publisher. When combined with the publisher namespace found in /document/publisher/namespace, it serves as a basis for querying the current status of a CSAF document. Figure 1 illustrates an overview of the complete CSAF document object located at /document.

In addition to *document/timeline/release_date/current*, the products specified in CSAF documents also play a significant role in determining the relevance of the documents retrieved via the API.

In a CSAF document, products can be defined in various locations (see Table 5). For instance, entire product families can be specified within the */product_tree/branches* section. This section allows for an infinite nesting of branches, which can make querying complex due to the unpredictable depth of these nestings.

Table 5

Options for Product Definition in the CSAF Document

Product ID	Definitions
1	**Product Tree Branches** `GET /products/branches/` *Lists branches within the product tree.*
2	**Full Product Names** `GET /products/names/full` *Returns a list of full product names.*
3	**Product Relationships by Full Product Name** `GET/products/{productId}/relationships` *Retrieves relationships for a product based on its full name.*
4	**Product Groups and IDs** `GET /products/groups` *Lists product groups and their corresponding product IDs.*

Once a product is defined, it can be referenced throughout the CSAF document by its product ID (refer to Table 6). For instance, products can be categorized into groups along the specified path: /products/groups[]/product[].

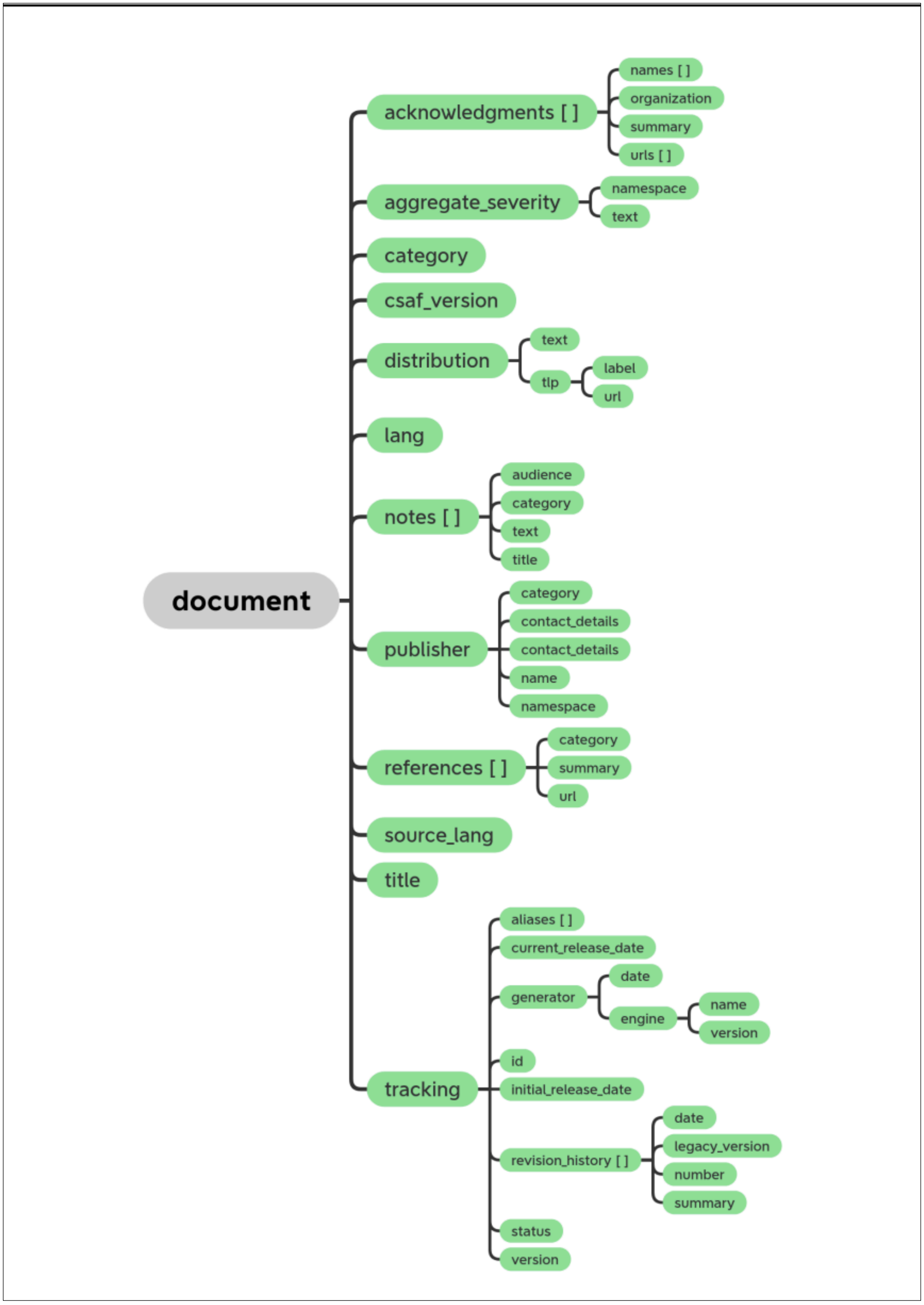


Figure 2 – Document tree structure

Table 6

CSAF Document Elements That Reference a Product Using the Product ID

Product ID	References
1	**Product Group References by Product ID** `GET /products/groups/{groupId}/references` *Returns product references within a group based on a product ID.*
2	**Product Relationships by Reference** `GET /products/{productId}/relationships/{referenceId}` *Fetches relationships for a product based on a reference.*
3	**Product Relationships by Relates to Reference** `GET /products/{productId}/relationships/related/{referenceId}` *Fetches product relationships based on a related product reference.*
4	**Vulnerabilities Flags by Product ID** `GET /vulnerabilities/products/{productId}/flags` *Lists vulnerabilities for a product reference.*
5	**Product Vulnerability Status - First Affected** `GET /vulnerabilities/products/{productId}/status/first-affected` *Lists the first affected product in a vulnerability.*
6	**Product Vulnerability Status - First Fixed** `GET /vulnerabilities/products/{productId}/status/first-fixed` *Lists the first fixed product in a vulnerability.*
7	**Product Vulnerability Status – Fixed** `GET/vulnerabilities/products/{productId}/status/fixed` *Lists fixed products in a vulnerability.*
8	**Product Vulnerability Status - Known Affected** `GET /vulnerabilities/products/{productId}/status/known-affected` *Lists products that are known to be affected.*
9	**Product Vulnerability Status - Known Not Affected** `GET /vulnerabilities/products/{productId}/status/known-not-affected` *Lists products that are known not to be affected.*
10	**Product Vulnerability Status - Last Affected** `GET /vulnerabilities/products/{productId}/status/last-affected` *Lists the last affected product in a vulnerability.*
11	**Product Vulnerability Status - Recommended** `GET /vulnerabilities/products/{productId}/status/recommended` *Lists recommended products in a vulnerability.*
12	**Product Vulnerability Status - Under Investigation** `GET /vulnerabilities/products/{productId}/status/under-investigation` *Lists products under investigation for vulnerabilities.*
13	**Remediations by Product ID** `GET/vulnerabilities/remediations/{productId}` *Lists available remediations for a product based on its ID.*
14	**Scores by Product** `GET /vulnerabilities/scores/{productId}` *Fetches vulnerability scores for a product*

Figure 3 depicts the product tree, showcasing its potentially infinite nesting within the ../branches[] structure, highlighted in light blue.

The array */product_tree/full_product_names[]* enumerates all products that are not part of */product_tree/branches[]*. Each product entry comprises a name, product_id, and product_identification_helper. The product name serves as a valuable reference for querying dependencies associated with products relevant to one's area of responsibility. Concentrating on specific products improves the relevance of search results, particularly when evaluating vulnerabilities that may impact those products.

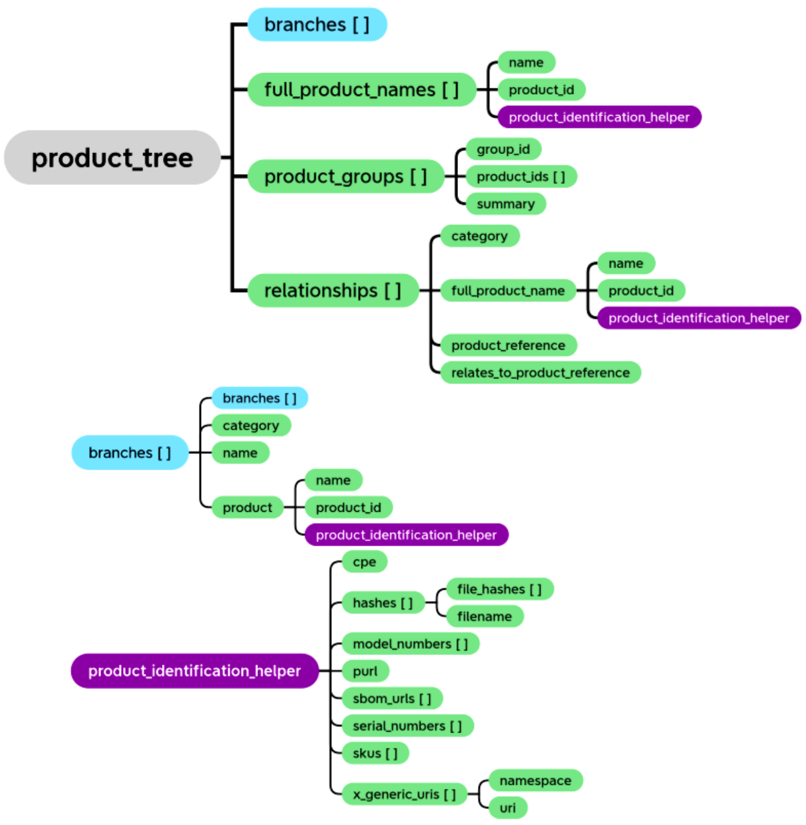


Figure 3 – CSAF document - product tree

DISCUSSION OF RESEARCH RESULTS

GraphQL can be likened to a specialized filter that allows users to retrieve precisely the data they request, presenting only those elements of an object that have been explicitly queried (see Description of GraphQL).

In the context of the intended API, however, a challenge arises. When a CSAF consumer queries for elements, there may be a need to retrieve all elements of a CSAF document without prior knowledge of which elements exist or having to query each possible element individually [6-10]. This situation can undermine the following principle:

Submit a GraphQL query to your API to receive precisely the information you require—no excess and no shortage. GraphQL queries consistently yield reliable outcomes. Applications that utilize GraphQL are quick and dependable since they manage the data they receive rather than relying on the server.

Despite this challenge, GraphQL remains promising. It allows for targeted querying of individual elements, validates inputs and outputs, and has a track record of successful implementation ion the Internet. These benefits are particularly relevant for ensuring the secure operation of the intended API. Previous attempts at API implementation, including those based on REST, provide valuable insights and experience that will inform the development of the GraphQL-based API.

Creating a GraphQL-based API for CSAF documents brings a range of advantages, especially in handling complex data structures like security advisories. Overall, a GraphQL API for CSAF documents offers a high degree of flexibility, efficiency, and adaptability, making it easier to manage and deliver complex security advisory data in a way that is both scalable and secure.

CONCLUSIONS

The paper addresses the issue of processing security recommendations in a single data format. These recommendations are issued by vendors, IT security experts, or coordinating organizations to help users understand vulnerabilities and take steps to eliminate or mitigate them. To improve the efficiency of query processing, this paper proposes to focus on documents containing specific content rather than processing all documents. It proposes to use GraphQL, an open-source data query language that allows for efficient formalization of CSAF document queries. It describes the features of implementing a GraphQL-based API to improve the efficiency of CSAF document query processing.

Creating a GraphQL-based API for CSAF documents brings a range of advantages, especially in handling complex data structures like security advisories. IT security professionals managing multiple products would benefit from an API that can efficiently retrieve all relevant security advisories affecting their products.

REFERENSES

1. Trim, Peter R. J., and Yang-Im Lee. (2024). "Advances in Cybersecurity: Challenges and Solutions" Applied Sciences 14, no. 10: 4300. https://doi.org/10.3390/app14104300
2. Park, M.; Lee, H.; Kim, Y.; Kim, K.; Shin, D. (2022). Design and implementation of multi-cyber range for cyber training and testing. Appl. Sci., 12, 12546.
3. Xu, S.; Qian, Y.; Hu, R.Q. (2019), Data-driven edge intelligence for robust network anomaly detection. IEEE Trans. Netw. Sci. Eng. 7, 1481–1492.
4. Langley Rock, Stefan Hagen, and Thomas Schmidt, (June 2022). eds. Common Security Advisory Framework. Version 2.0 Committee Specification (CS) 02. OASIS, url:https://docs.oasis-open.org/csaf/csaf/v2.0/cs02/csaf-v2.0-cs02.html.
5. CSAF-documentation, url: https://oasis-open.github.io/csaf-documentation/
6. Lawi, Armin, Benny L. E. Panggabean, and Takaichi Yoshida. (2021). "Evaluating GraphQL and REST API Services Performance in a Massive and Intensive Accessible Information System" Computers 10, no. 11: 138. https://doi.org/10.3390/computers10110138
7. Ala-Laurinaho, Riku, Joel Mattila, Juuso Autiosalo, Jani Hietala, Heikki Laaki, and Kari Tammi. (2022). "Comparison of REST and GraphQL Interfaces for OPC UA" Computers 11, no. 5: 65. https://doi.org/10.3390/computers11050065
8. Malo-Perisé, Pedro, and José Merseguer. (2022). "The “Socialized Architecture”: A Software Engineering Approach for a New Cloud" Sustainability 14, no. 4: 2020. https://doi.org/10.3390/su14042020
9. Swagger. OpenAPI Specification. Version 3.0.3. Swagger, Feb. 2020. url: https : //swagger.io/specification/.
10. Smartbear. SoapUI Docs: Working With REST Services and WADL. https : //www.soapui.org/docs/rest-testing/working-with-rest-services/. 2022.

Надійшла до редакції: 15.09.2024

MISTRIAKOV VITALII – master student, Computer Science department, West Ukrainian National University, ***e-mail: pan54453@163.com***

PAN TIANDE – PhD student, Computer Science department, West Ukrainian National University, ***e-mail: pan54453@163.com***

ВІТАЛІЙ МІСТРЯКОВ, ПЕН ТІАНДЕ

ОПРАЦЮВАННЯ ЗАПИТІВ КОНТЕНТУ CSAF ДОКУМЕНТІВ З ВИКОРИСТАННЯМ API НА ОСНОВІ GRAPHQL

Західноукраїнський національний університет, Тернопіль Україна