

# [Appendix] AdvisoryHub: Design and Evaluation of a Cross-Platform Security Advisory System for Cyber Situational Awareness

Marc-André Kaufhold<sup>[0000–0002–0387–9597]</sup>, Julian Bäumler<sup>[0000–0002–4535–8036]</sup>,  
Nicolai Koukal, and Christian Reuter<sup>[0000–0003–1920–038X]</sup>

Technical University of Darmstadt, Science and Technology for Peace and Security  
(PEASEC), Pankratiusstraße 2, 64289 Darmstadt, Germany  
{kaufhold,baeumler,koukal,reuter}@peasec.tu-darmstadt.de

## Introduction

This document constitutes the appendix to the above-mentioned paper which was published in the Proceedings of the 20th International Conference on Availability, Reliability and Security (ARES 2025) [6]. It comprises supplementary information on how requirements were derived from related work and interviews (Section 1), the technical system was implemented (Section 2), and the coding for the extractor performance evaluation was conducted (Section 3).

## 1 Empirical Pre-Study

### 1.1 Derivation of Requirements from Related Work and Interviews

Based on the framework of [4], the overall requirement of the system is to support CERT members with the perception of the elements relevant to CSA alongside their attributes, understanding their significance to the situation and their relation to each other and enable the user to make projections on how the situation will evolve. The perception of relevant elements can be supported by retrieving security advisory documents and extracting vulnerabilities with their attributes from the retrieved documents. The comprehension of the significance of the advisories or vulnerabilities and their relation to each other is best achieved by performing processing steps such as extracting information on the impact, correlating and fusing multiple alerts related to the same vulnerability and filtering out unimportant information. In a vulnerability-centered design, it is harder to create projections compared to a threat-centered design, since vulnerability exploitation depends on the (external) threat environment. Nevertheless, temporal aspects and, therefore, projections should be taken into account, for example, to model how the risk of exploitation evolves over time.

To achieve these overall goals, a chain of processing steps needs to be executed. The specific requirements for these steps were extracted from [2], [1], [5] and combined with the requirements identified in the interview analysis. Since

the literature-based approaches were rather threat-centered than vulnerability-centered, not all of their respective requirements were applicable to a vulnerability-centered system. The combination of the requirements in table 1 results in a series of objectives that describe different modules of the overall system.

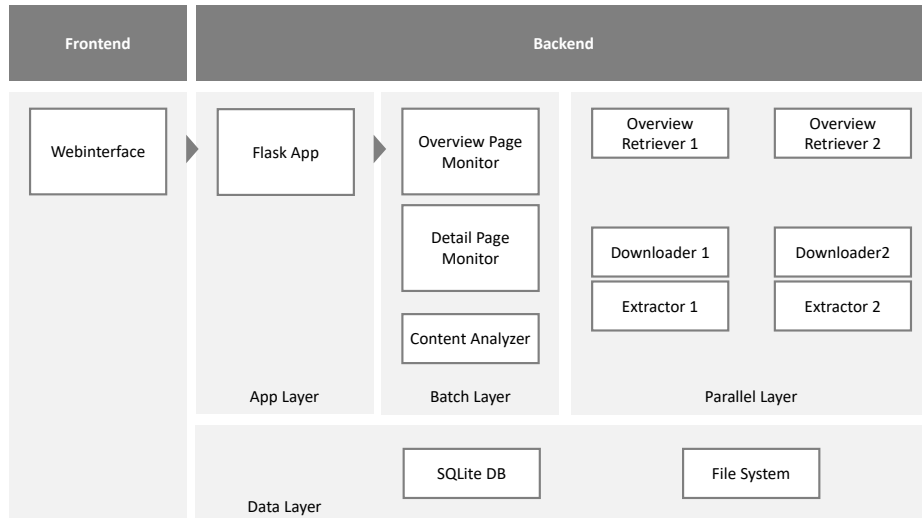
Jajodia, 2010	Alavizadeh, 2022	Erbacher, 2010	Interviews	Objective
	Data collection from different sources	Using as many sources as possible	There should be a comprehensive prefilled catalog of relevant sources.	Source Selection
Data collection	Data collection from different sources		Different sources are monitored and new advisories are retrieved automatically.	Retrieval
Impact assessment	Data parsing, cleansing, and normalization, determining impact, using scoring systems and unique identifiers	Identification of vulnerabilities and their respective impact	Advisories are transformed into a standardized format to enable automation, source format changes are handled gracefully, vulnerabilities are tagged with a score.	Extraction, Identification, and Impact Assessment
			Advisories from different national CERTs can be used.	Translation
	Topology, configuration, components of systems	System importance	It is checked if a vulnerability is relevant for the constituents.	ITAM Matching
Data quality (soundness, completeness, and freshness)	Alert correlation, duplicate elimination, data fusion	Weight based on accuracy, data origin, trust	Data from different sources is merged, information coming from upstream CERTs is supplemented and enriched, a confidence level is displayed.	Correlation, Fusion, Credibility Assessment
Awareness on how situations evolve	Awareness on how situations evolve	Timeline, criticality, prevalence of particular attack	It is taken into account if information is already exploited in the wild. The CVSS score provided by the source is adjusted. Social media is used to determine how relevant a vulnerability is.	Risk Calculation
Data can overwhelm cognitive capacity		Ability to filter data	Vulnerabilities are sorted, prioritized and filtered.	Prioritization Filtering
	Collation of data	Identification of remediation steps	Recommendations are added and information is summarized.	Summarizing Recommendations
			Reports or exports are created using the collected, processed and analyzed data.	Dissemination

**Table 1.** Functional Requirements for a Vulnerability-Based CSA System

## 2 Implementation

### 2.1 Architecture

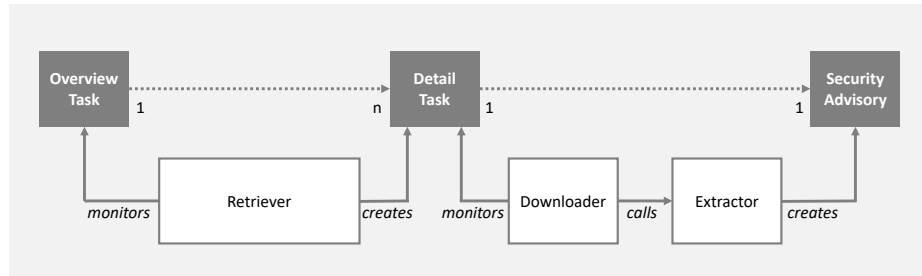
Overall, we implemented a Python application. The frontend web interface for configuring monitoring and retrieval of the advisory sources was implemented using the Bootstrap framework in combination with the JavaScript library D3.js, which was utilized to render charts. The backend is divided into four layers. The app layer forms the counterpart to the frontend by receiving the HTTP requests and responding to them accordingly. The batch layer is responsible for continuously running processes in the background to monitor **overview pages** (e.g., lists or tables) that contain references to advisory document detail pages, as well as monitoring these **detail pages** (including detailed attributes), and analyzing the retrieved content. To retrieve overview pages and process the content on the detail pages, the monitoring functions of the batch layer spawn retriever, extractor, and constructor function in the parallel layer. The data layer is responsible for persisting both the configuration data and the retrieved security advisory documents. While the configuration data and selected metadata are stored in an SQLite database, the retrieved security advisory documents are stored as files on disk. While the overall system is designed to address the functional requirements of data source selection (R1), document retrieval (R2) and data extraction (R3), we will discuss the consideration of non-functional requirements within the following subsections.



**Fig. 1.** Overall Architecture of the Implemented System

## 2.2 Data Model

The system's high-level data model was based on the way security advisory documents are commonly published. Product CERTs typically offer a feed of new advisory documents, for example in the form of a web page or an RSS feed. These feeds contain several elements representing advisory documents, which each consist of a reference to a detail page and metadata. The advisory attributes are published on the detail page. To monitor both the overview pages for new advisories and the detail pages for changes in the advisory attributes, the data structures *overview tasks*, *detail tasks*, and *security advisories* were created. As depicted in figure 2, the retriever class monitors the overview tasks for changes and creates a detail task for every new detail page. The downloader monitors the detail pages for changes, downloads them, extracts the information by using the extractor class and creates security advisories, one for every retrieved detail page.



**Fig. 2.** Process of the Retrieval and Extraction System

**Overview Tasks** Overview tasks are configuration items that describe how a particular advisory overview page should be monitored by the system. Table 2 describes the attributes of an overview task object and their respective functions. Overview tasks are executed by a monitoring method and result in the creation of detail task objects.

Attribute	Description
id	A unique ID for identifying the task
type	Provides information on how the page can be monitored
request_method	HTTP request method that is used for retrieval
url	URL of the site on which the desired information is displayed

Continued on next page

Table 2: Attributes of Overview Tasks

retrieval_endpoint	URL from which the system can retrieve the desired content, such as an API endpoint or the URL of an RSS feed
selector	A string that helps the system to locate the desired content in the response coming from the server
body	Body of the (non-GET) HTTP request
header	Additional HTTP request header fields
mapping	Used to identify column types in response
content mapping	Used to identify which content types are present in the retrieved advisory document
ignore references	References that are part of the template and not part of the individual advisory document
interval	Interval of monitoring/retrieval attempts
statusmessage	Storing error messages
lastretrieval attempt	Last time the system tried to retrieve the overview page
lastretrieved	Last time the system successfully retrieved the overview page

Table 2: Attributes of Overview Tasks

**Detail Tasks** Detail task objects are created when retrievers are fed overview tasks. The detail tasks consist of a reference to an advisory page and metadata on the referenced advisory page. Retriever objects transform overview tasks into detail tasks and extractor objects transform detail tasks into advisory documents. Advisory tasks are identified either by their URL or by a composite key consisting of the vendor-provided identifier and the URL of the overview page from which the detail task was created). This system allows identifying detail tasks and their respective detail pages in two distinct cases: In the default case, the vendor provides a unique detail page URL on the overview page, one for each detail page. Alternatively, there are no detail pages, but the available data consists only of the data provided on the overview page. In that case, an identifier provided by the vendor is necessary.

The missing detail page URL can occur when retrieving data from XHR/JSON APIs where all advisory information is contained in the API response and no static page containing the advisory detail information is available. In contrast, it is often the case that there is no ID field when extracting data from HTML tables or using the HTML URL retriever, which only extracts the links to the detail pages. Since the extracted metadata is also used for extraction of advisory information, a URL is not strictly necessary. Detail page attributes that are unusual or specific to only one advisory provider are not inserted into dedicated database columns, but stored in the JSON-format in a catch-all column. Since multiple vendors may use overlapping name-spaces, the vendor-provided identifier can only be used for global identification if it is combined with the URL of

the overview page from which the detail task was created.

Attribute	Description
domain	Reference to the overview page on which the detail page was found
url	An URL under which the detail page can be found
firstseen	Timestamp of when the detail task was created
lastretrieved	Timestamp of last retrieval
title	Title of the advisory
identifier	Vendor-provided identifier of the advisory
products	Products affected by the advisory
vendor	Vendors affected by the advisory
published	Publishing timestamp
updated	Timestamp of last update
cve	CVEs referenced by the advisory
score	CVSS score
severity	Severity of the advisory
content	Textual content of the advisory (long)
summary	Textual content of the advisory (short)
type	A vendor-provided category
retrievalserverts	Timestamp provided by server in request response
retrievaletag	A entity tag used for cache validation
properties	List of additional metadata attributes (as JSON)

Table 3: Attributes of Detail Tasks

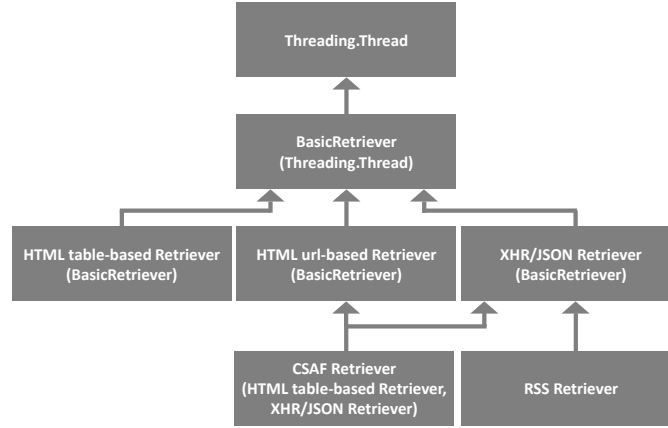
As can be seen in table 3, the detail task consists of the attributes domain, URL, and identifier as well as additional metadata attributes. The lastretrieved timestamp is used for retrieval scheduling. The retrievalserverts and the retrievaltag are used to send conditional requests to the web server that only result in a response in cases the content has changes. This leads to a reduction in the bandwidth used. To enable the system to store advisory metadata that is not part of the data model, the attribute properties allows for storing additional attributes as a JSON string.

**Advisory Documents** Advisory documents are created when detail tasks are executed. They are stored as files in subfolders which carry the name of their

provider and adhere to the CSAF format. For displaying a preview list of stored advisory documents, a subset of their attributes are stored in a database. This allows for quicker response times when the user queries the list of advisories.

### 2.3 Monitoring of Overview Pages

The system is able to monitor a variety of different data sources such as HTML sites, CSAF feeds, and RSS feeds. The HTML-based sources have been further divided based on how the information is presented on the site. To account for the differences between the data source categories, a hierarchical system of retriever classes was developed, that structures the retrieval logic. Each of these retriever classes ultimately inherits from the Python class *threading.thread* to enable asynchronous and parallel retrieval.



**Fig. 3.** Inheritance Model of the Retriever Modules

The retrievers differ in the steps performed to process the received HTTP responses. In particular, the identification and standardization of the URL that points to the detail page varies from retriever to retriever. Another example becomes apparent when considering overview pages that contain multiple HTML tables: The table-based retriever must determine which of them is the one most likely to contain the desired advisory information. In contrast, the XHR/JSON retriever works with nested JSON responses, from which a list of references to detail pages and their respective metadata needs to be extracted. All retriever transform the retrieved data into a Pandas dataframe table structure. The rows of these tables represent the different advisory pages and the columns represent the attributes such as the URL, the title, and the severity (if available).

When the user sets up a new source for monitoring, a test request is performed to make sure that the appropriate retriever type was selected. The retrieved overview table is used to generate a mapping from source-specified column names to standardized column types. The system assists the user by suggesting column types based on a pre-filled dictionary of column names and their respective column types. This mapping is stored in the database alongside the other configuration properties such as the URL to be retrieved, the retriever type and special request header fields that should be sent when making the HTTP request. When the individual retriever has finished downloading the overview page, tabularizing the data, and transforming the table into a list of detail tasks, it checks whether the detail tasks already existed in the database. This is either done based on the URL of the detail page, or alternatively based on the composite key consisting of the vendor-provided identifier in combination with the URL of the overview page. If the detail task did not previously exist in the database, it can be concluded that the provider published a new advisory document and therefore the detail task is added to the database.

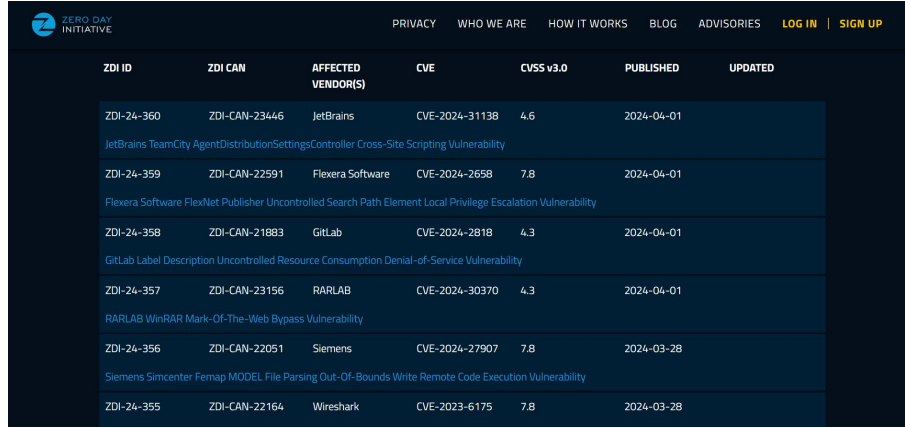
**Base Retriever** The Base retriever class is an abstract base class inheriting from the `threading.Thread` class. This enables multiple retrievers to run in parallel. To ensure efficient use of computing resources, the number of objects inheriting from the base retriever can be counted before creating a new retriever. In case the number of active retrievers threads exceeds a set level, the creation of new retrievers is paused until a retriever finishes execution and a slot becomes available again.

**HTML Table Retriever** The HTML Table Retriever allows the system to extract data from static websites containing the desired data within HTML table tags. The rows of these tables represent the advisory detail pages and their columns represent advisory metadata, including a link to the respective detail page for the advisory document.

## 2.4 HTML XPATH Retriever

The HTML XPATH Retriever allows the system to create detail tasks not only from table tags, but arbitrary HTML elements that are part of the Document Object Model (DOM) of the monitored site. A representative application for this retriever are sites that contain multiple `div` elements that represent the advisories and their respective metadata attributes. The elements of interest are selected via the XPATH expression language. For example, the XPATH string `//div[contains(@class, 'advisory')]` can be used to select all `div` elements in the page DOM that have the string `advisory` in their class name. The retriever converts the list of selected elements and their respective child nodes into a list of dictionaries. The keys of these dictionaries consist of the tag name and the attribute name, the values are the values that are extracted from the element





The screenshot shows the AdvisoryHub website interface. At the top, there is a navigation bar with the 'ZERO DAY INITIATIVE' logo on the left and links for 'PRIVACY', 'WHO WE ARE', 'HOW IT WORKS', 'BLOG', 'ADVISORIES', 'LOG IN', and 'SIGN UP' on the right. Below the navigation bar is a table with the following columns: ZDI ID, ZDI CAN, AFFECTED VENDOR(S), CVE, CVSS v3.0, PUBLISHED, and UPDATED. The table contains seven rows of advisory data, each with a link to the full advisory details.

ZDI ID	ZDI CAN	AFFECTED VENDOR(S)	CVE	CVSS v3.0	PUBLISHED	UPDATED
<a href="#">ZDI-24-360</a>	<a href="#">ZDI-CAN-23446</a>	JetBrains	<a href="#">CVE-2024-31138</a>	4.6	2024-04-01	
<a href="#">JetBrains TeamCity AgentDistributionSettingsController Cross-Site Scripting Vulnerability</a>						
<a href="#">ZDI-24-359</a>	<a href="#">ZDI-CAN-22591</a>	Flexera Software	<a href="#">CVE-2024-2658</a>	7.8	2024-04-01	
<a href="#">Flexera Software FlexNet Publisher Uncontrolled Search Path Element Local Privilege Escalation Vulnerability</a>						
<a href="#">ZDI-24-358</a>	<a href="#">ZDI-CAN-21883</a>	GitLab	<a href="#">CVE-2024-2818</a>	4.3	2024-04-01	
<a href="#">GitLab Label Description Uncontrolled Resource Consumption Denial-of-Service Vulnerability</a>						
<a href="#">ZDI-24-357</a>	<a href="#">ZDI-CAN-23156</a>	RARLAB	<a href="#">CVE-2024-30370</a>	4.3	2024-04-01	
<a href="#">RARLAB WinRAR Mark-Of-The-Web Bypass Vulnerability</a>						
<a href="#">ZDI-24-356</a>	<a href="#">ZDI-CAN-22051</a>	Siemens	<a href="#">CVE-2024-27907</a>	7.8	2024-03-28	
<a href="#">Siemens Simcenter Femap MODEL File Parsing Out-Of-Bounds Write Remote Code Execution Vulnerability</a>						
<a href="#">ZDI-24-355</a>	<a href="#">ZDI-CAN-22164</a>	Wireshark	<a href="#">CVE-2023-6175</a>	7.8	2024-03-28	

**Fig. 4.** Example for an HTML Table-Based Overview Page

attributes. For example, the key *a.href* could point to the detail page of a given advisory document. After the transformation into a list of dictionaries, a table is created, from which the detail tasks are created, similar to the processing step of the table retriever.

## 2.5 HTML URL Retriever

The HTML URL Retriever acts similar to the HTML XPATH retriever, but only extracts a-tags from the DOM. To filter out references to pages that are not detail pages, the results are filtered with a prefix provided by the user. This prefix must be common to all the detail pages; however, it must be specific enough to distinct detail pages from not detail pages. An example could be a string such as *http://www.example.com/advisories/*. The HTML URL retriever is more robust than the HTML Table retriever, but it is not able to extract the same amount of metadata provided by the overview page, such as the publishing data or the affected products. This information can only be extracted from the content of the detail page later on.

## 2.6 HTML Ajax Retriever

In cases where the information to be extracted is not part of the static HTML DOM, but is later added by the execution of dynamic JavaScript, another approach is needed. The HTML Ajax Retriever allows the system to extract the advisory information directly from JSON APIs that are called by the JavaScript code to populate the overview pages. For setting up the retriever in the current state of development, the user must manually inspect the overview site and determine the URL and the request parameters of the called API. This information is provided to the retriever to request the JSON data containing the desired information. The retriever is able to automatically find the required data within

the JSON response and transform it into a table. The retriever does this by searching for lists in the response and constructing tables from them. Then the correct table is selected heuristically by taking both the shape of the table and the column names into account. After transforming relative URLs into absolute URLs, the selected table can be processed in the same way the other HTML retriever classes process the constructed table.

**RSS Retriever** The RSS Retriever transforms an XML-based RSS feed into a list of item elements representing the different advisories. The list is then transformed into a table and used to construct detail tasks, as in the other tables. The advisory content which is often already part of the RSS feed is stored in the detail task content field.

**CSAF Retriever** The CSAF Retriever transforms the JSON-based CSAF feed into a table, which is used to construct detail tasks.

## 2.7 Interface

The interface of the web application consists of a dashboard, a page to add and inspect overview monitoring tasks, a page to get recommendations for new advisory sources, and a preview that shows the retrieved documents.

**Dashboard** The dashboard consists of status elements indicating whether the retriever works properly and metrics such as the number of new advisory documents discovered in the last 24 hours. Additionally, there is a time-series column chart showing how many advisory documents were published on a given date. This allows the user to quickly identify abnormalities, such as higher-than-usual number of new vulnerabilities. By clicking on the date in question, the user can investigate the advisories published on that date in the preview menu.

**Inspecting Monitoring Tasks and Adding New Advisory Sources** On the overview task page, the user can inspect the running tasks. By clicking on the individual tasks, the user can inspect the task properties such as the configured request body and header or error messages in case the retriever failed to poll a data source. New sources can be either added individually via a web form or in bulk via a CSV import function. When adding new sources via the web form, the user must first select the retriever type the system should use and then set the other task properties. After submitting the task, a test request is conducted, to verify that the task was configured properly. If the task was configured incorrectly, or the polled server returned for other reasons, the user is displayed an error message. In case the request worked, the user needs to set column types in the tabularized form of the response. The system suggests automatically column types based on the column names coming from the server. After the column types such as *URL*, *title*, or *identifier* were set, the user can add

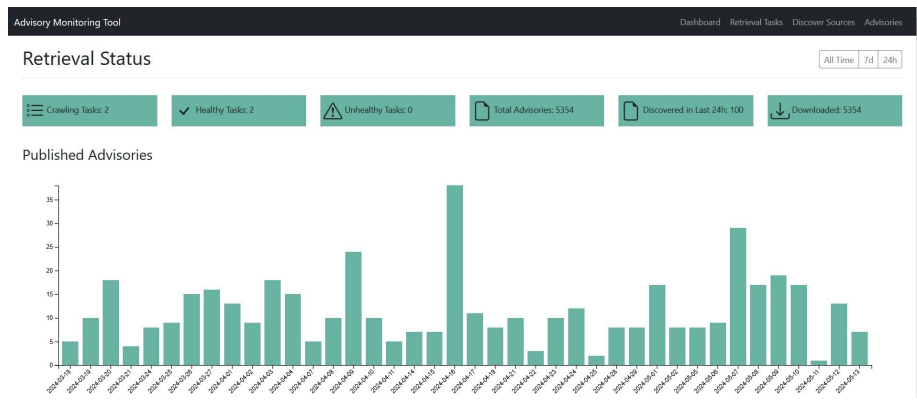


Fig. 5. Screenshot of the Dashboard View

the task to the database. The system then automatically executes the retrieval in the background and creates detail tasks if new advisory documents are detected.

XPATCH	First
Match classes of elements	//element-tag[contains(@class, '%')]
Get the n-th child div	//div/div[n]

Fig. 6. Web Form for Adding New Overview Tasks

**Recommendation of New Data Sources** The system automatically suggests new data sources based on the references of the documents already ingested and analyzed. In the recommendation view, that can be seen in figure 7, the user sees how often a particular provider was referenced by the ingested advisories. If there is already a monitoring task for that vendor, the respective bar is marked as green; otherwise it is marked red. The user can therefore screen the suggested sources by clicking on the red bars. This forwards the user to one example referenced detail page. After deciding, whether the advisory source may be of interest to the user, the user needs to navigate to the overview site of the provider and add it as a new task as described in section 2.7

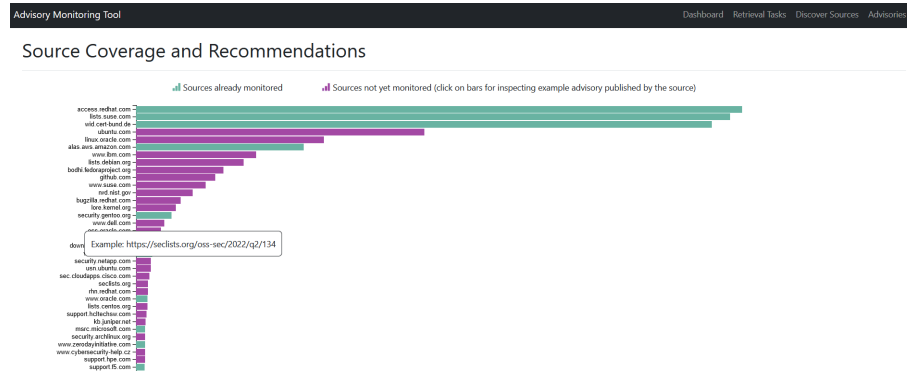


Fig. 7. Suggestions for Additional Sources to Monitor

**Advisory Preview** As can be seen in figure 8, the implementation also includes a preview that displays the retrieved and extracted advisory documents. The user can access the original resource by clicking on the link symbol left of the title. By clicking on the provider of an advisory document, the user can filter for only documents that were published by that particular provider. This functionality is also offered for the name of the affected vendor and CVE numbers. The preview does not contain all advisory attributes, but they are stored in the CSAF files that were saved to disk.

Advisory Monitoring Tool						
Dashboard Retrieval Tasks Discover Sources Advisories						
Retrieved Advisories						
Title	Published	Updated	Severity	Provider	CVEs	Affected
<input type="checkbox"/> strongswan: Schwachstelle ermöglicht Umgehen von Sicherheitsvorkehrungen	2024-05-14 22:00:00	2024-05-14 22:00:00	hoch	<a href="#">wid.cert-bund.de</a>	<a href="#">CVE-2022-4967</a>	<a href="#">open source</a>
<input type="checkbox"/> IBM QRadar SIEM: Mehrere Schwachstellen	2024-05-14 22:00:00	2024-05-14 22:00:00	mittel	<a href="#">wid.cert-bund.de</a>	<a href="#">CVE-2023-31582</a> <a href="#">CVE-2023-51775</a>	<a href="#">ibm</a>
<input type="checkbox"/> Fortinet FortiOS: Mehrere Schwachstellen	2024-05-14 22:00:00	2024-05-14 22:00:00	mittel	<a href="#">wid.cert-bund.de</a>	<a href="#">CVE-2023-46714</a> <a href="#">CVE-2023-44247</a> <a href="#">CVE-2024-26007</a>	<a href="#">fortinet</a>
<input type="checkbox"/> Dell BIOS: Schwachstelle ermöglicht Codeausführung	2024-05-14 22:00:00	2024-05-14 22:00:00	mittel	<a href="#">wid.cert-bund.de</a>	<a href="#">CVE-2024-22429</a>	<a href="#">dell</a>
<input type="checkbox"/> VMware Workstation und VMware Fusion: Mehrere Schwachstellen	2024-05-14 22:00:00	2024-05-14 22:00:00	hoch	<a href="#">wid.cert-bund.de</a>	<a href="#">CVE-2024-22268</a> <a href="#">CVE-2024-22269</a> <a href="#">CVE-2024-22270</a> ... show more ...	<a href="#">vmware</a>
<input type="checkbox"/> Intel Chipset Software: Schwachstelle ermöglicht Privilegieneskalation	2024-05-14 22:00:00	2024-05-14 22:00:00	mittel	<a href="#">wid.cert-bund.de</a>	<a href="#">CVE-2024-21814</a>	<a href="#">intel</a> , <a href="#">lenovo</a>
<input type="checkbox"/> Microsoft Power BI: Schwachstelle ermöglicht Offenlegung von Informationen	2024-05-14 22:00:00	2024-05-14 22:00:00	mittel	<a href="#">wid.cert-bund.de</a>	<a href="#">CVE-2024-30054</a>	<a href="#">microsoft</a>
<input type="checkbox"/> Intel FPGA Firmware: Mehrere Schwachstellen	2024-05-14 22:00:00	2024-05-14 22:00:00	mittel	<a href="#">wid.cert-bund.de</a>	<a href="#">CVE-2023-41092</a> <a href="#">CVE-2024-22380</a>	<a href="#">intel</a>

Fig. 8. Preview of Processed Advisory Documents

## 2.8 Monitoring of Detail Pages and Scheduling of Detail Tasks

The detail pages are monitored by the Downloader class, which executes the detail tasks created by the retrievers. The class inherits, like the retriever classes,

from the Python Thread class to enable parallel processing. The Downloader checks the timestamp of the last retrieval attempt of all detail tasks stored in the database and determines whether they are eligible for execution. Whether a detail task is eligible for retrieval or not is based on a simple heuristic aimed to save both bandwidth and minimize the duration until new information is available to the operator.

New detail tasks which have never been executed and detail tasks which have a vendor-provided update date set after their last retrieval are always downloaded. All other tasks are downloaded at regular intervals, while newer detail tasks are downloaded more often than older detail tasks. Detail tasks for advisories younger than seven days are downloaded every six hours. Detail tasks younger than 30 days are retrieved every 48 hours. All other detail tasks are retrieved every 15 days. This heuristic ensures that even older advisories are monitored for changes frequently, which is important since vulnerabilities may be classified with a low severity early on and scored higher after some time.

To further minimize data transfers, the ETag header field, as well as server time header field, are stored for every executed detail tasks. On subsequent retrieval attempts, this data is sent to the server in the *ETag* and the *If-Modified-Since* header field of the request. The sent data allows the server to check whether the content changed since the last request and if that is the case, send a 304 *Not Modified* response. Given a 304 response code, it is not necessary for the system to parse the response, since the server already indicated that nothing changed since the last request.

After the successful retrieval of a detail page, the downloaded decides on how to process the response based on the *content-type* header field. If the content-type header field contains values such as *application/json*, *application/octet-stream*, *text/plain*, the response is assumed to be a CSAF file and directly written to disk. For the advisory preview page, some attributes from within the CSAF file, such as the tile and affected vendor, are written to the preview table. In cases where the server responds with a *text/html* response, the downloaded calls the extractor, which parses the HTML file. The extractor is then handed over to the constructor, which builds a CSAF from the extractor. After that, the CSAF file is written to disk, while some attributes are written to the preview table.

## 2.9 Extraction

The extractor is structured modular and utilizes multiple subextractors for extracting the various advisory attributes. For extraction, it is provided with the detail task that contains metadata extracted from the overview page, as well as an HTTP response in case the retrieval of the detail page succeeded. The extractor extracts the text from the response and then calls the subextractors to populate its attributes. The subextractors are called with a reference to the ex-

tractor object, which allows them to access the metadata of the task, the HTTP response and other previously extracted attributes.

**Extraction of Identifier** The identifier of an advisory document is directly taken from the detail task, in case it was provided on the overview page by the advisory provider. In cases where no identifier was provided, it is generated from the URL path. Given that a detail task must either contain a provider-unique identifier or a globally unique URL, it is guaranteed that a unique identifier for every retrieved advisory document exists. In some cases, the unique identifier is not part of the path, but of the query part of the URL. In these cases, the identifier is extracted from the query.

**Extraction of Title** The title is usually extracted from the detail task, as even the HTML URL Retriever is able to set it by reading it from the text property of the identified links. If there is no title attribute set in the detail task, the title is extracted from the title tag of the HTML response.

**Extraction of Vulnerabilities** Vulnerabilities are extracted both from the metadata extracted from the overview pages, and also the text content of the detail pages by using a regex pattern matching CVE numbers. Since all task metadata fields are searched for CVE numbers and not only the CVE field, it is ensured, that even CVE numbers in titles and summaries are identified.

**Extraction of CVSS Vectors and Scores** CVSS vectors are extracted both from metadata and text content of the detail page using a regex pattern. CVSS scores extracted by first checking the dedicated score attribute of the task object. If the score attribute is not set in the score attribute, strings such as CVSS or base score are searched both in the other metadata fields and the text content of the response. If one of the strings is found, potential score values are extracted from the proximity of the location. The score values of 0.0, 2.0, 3.0, 3.1, 4.0 are excluded from the results to prevent false positive hits caused by the indication of which CVSS version is used. While this filtering may cause false negative results, the risk is considered low since values are on the lower part of the CVSS score spectrum. If a CVSS vector can be extracted, but no CVSS score, the CVSS score is generated from the vector using the *cvss* Python library.

**Extraction of Languages** The language of the advisory is detected using the *langdetect* Python library. The library is provided with the complete text content of the advisory document and returns the ISO 639-1 code of the detected language.

**Extraction of Dates** The subextractor responsible for building the timeline of the security advisory first tries to get the publication date and the date of

the last update from the detail task metadata. After that, the extractor ties to find strings that are formatted like dates in the response content text. The search is performed by using multiple regex patterns. For each search result, the surrounding area is searched for keywords such as *published*, *update*, or *updated*. After establishing the context of the found dates, the timeline is constructed, beginning with the publishing dates, followed by the update events. If no publication date is found, the time the advisory document was first discovered is assumed as the publication date. If multiple potential publication dates are found, the earliest is assumed to be the correct one.

**Extraction of Vendors and Products** For identifying vendor and product names, a CPE database which was generated from the official dictionary file of the NIST NVD <sup>1</sup> is used. The SQLite database was populated by extracting the CPE string from the provided XML file and populating the tables, one for the vendors and one for the products. The products table also contains information on which vendor offers the product.

The vendors names are searched for in the vendor attribute, the product attribute, the url attribute, and the title attribute. To also find vendor names consisting of multiple words, n-grams are generated and joined by the underscore character before lookup, adhering to the CPE standard [3]. Since the lookup is very compute-intensive, the content text is not searched for vendor names. After vendor candidates were found, their product offerings are queried from the database. These product names are then searched in the task title and product attribute. If no product can be found using this strategy, the content of the product field is directly checked against the database without taking the vendor-product relation into account. If a product was found using this strategy, it is added to the product list. Its respective vendor is also extracted, even when the vendor name itself has not been found in the advisory document or its metadata. If multiple vendors or vendor product combinations have been found, they are all assumed to be affected by the advisory document, since false positives can be filtered out by the human operator, but false negatives may be suppressed by filters and lead to detrimental effects on the situational awareness.

**Content Segmentation and Extraction of Notes** In earlier versions of the system, a function for segmenting the site content of the detail page was implemented. It worked by identifying the HTML h1-, h2-, and h3-tags as anchors for content segmentation. Based on multiple samples of detail pages on the same provider website, the system aligned the h-tags and the content between them and used the information for building wrappers. Since the text of some h-tags was different between instances, these *dynamic* headings were replaced by an asterisk. For determining the content type of the different segments, the user was asked during configuration of the overview page task. During the implementation, it turned out, that the approach did not work well enough with real-world

<sup>1</sup> <https://nvd.nist.gov/products/cpe>

data due to the complex structure of the detail pages.

Instead, an approach was chosen, that extracts the entire site text after removing undesired element such as the site header and navigation elements.

**Extraction of References** References are extracted by finding a-tags in the site’s content. The href attribute is extracted alongside the link’s text. To make sure that only references specific to the individual advisory document are extracted and not such links that are part of the site’s template, a list of template URLs is generated during monitoring configuration. The system automatically retrieves four example instances of advisory sites for analyzing. Links that occur on each of the example pages are considered as part of the template and not extracted.

## 2.10 Construction of CSAF Files

On instantiation, the Constructor class is provided with an extractor object, from which it gets all necessary data to construct a CSAF file. The Constructor class determines in which subdirectory the document should be saved based on the URL of the overview page. The filename is created with the extracted advisory identifier as input. Since the identifier in combination with the provider unambiguously identifies an advisory document, file collisions are prevented. This is also used to check whether the advisory document was retrieved at an earlier date and, if so, whether there were any changes in the source document since then. The comparison between the old file and the new file is done with the *DeepDiff* library.

# 3 Evaluation

## 3.1 Coding for the Extractor Performance Evaluation

The evaluation of the extraction and standardization functionality was conducted similarly to the evaluation of the retrieval function. For each of the 48 sites that were successfully monitored by the retrieval modules, one random extracted document was manually inspected and rated regarding the quality of the extraction of its most important attributes. The attributes selected were title, vendor, product, the text content, a timeline consisting of publication and update dates, a severity string, a CVSS score, and the references. The extraction quality was rated on a scale from 0 to 3, with 0 meaning an attribute was not extracted or a wrong value was extracted that would be detrimental to the situational awareness of the operator. One example of such a case would be the extraction of a CVSS score of 2.1 when the real score was 9.3, since that could lead to the advisory document being omitted by an automatic filtering system or being disregarded by a human operator. While value 1 was used for extractions that contained some correct pieces of information, but were mainly incorrect,



the value 2 was used for extracted attributes that were inaccurate, but were not detrimental for establishing situational awareness. An example of this would be the extraction of an unrelated link as a reference. The value 3 was used for cases where the extracted value was identical to the value a human would extract. The detailed scoring criteria can be found in Table 4.

Attribute	Rating 3	Rating 2	Rating 1	Rating 0
Vendor	All correct vendors, but no FP vendors were extracted.	All correct vendors, but some unrelated (FP) vendors were extracted.	Some affected vendors were missed by the extraction.	No or only incorrect vendors were extracted.
Product	All correct products, but no FP products were extracted.	All correct products, but some unrelated (FP) products were extracted.	Some affected products were missed by the extraction.	No or only incorrect products were extracted.
Vulnerability	All vulnerabilities mentioned in the advisory were extracted.	All vulnerabilities mentioned in the advisory were extracted, but also additional unrelated vulnerabilities.	Some mentioned vulnerabilities are missing.	No vulnerabilities were extracted.
Timeline	Both creation and update dates were extracted correctly.	Either creation or update dates are correct, while the other one is incorrect.	Either creation or update dates are correct, while the other is incorrect.	Both creation and update date are incorrect.
Title	The correct title was extracted. If there were multiple titles, the best one was chosen.	The correct title was extracted. If there were multiple titles, the best one was not chosen.	Another string was extracted from the page. The string can be used to differ between multiple advisories from the same provider.	No string was extracted. Alternatively: The extracted string cannot be used to differ between different advisories of the same source.
Content	All relevant content was extracted, only minimal (10%) unrelated page content (such as advertisements and navigation elements) was extracted.	All relevant content extracted, significant (10%) unrelated page content (such as advertisements and navigation elements) was extracted.	Not all relevant content was extracted (e.g. summary was extracted, but details text was not extracted.)	No page content was extracted.
Severity	The correct severity was extracted.	An incorrect, higher severity was extracted.	An incorrect, lower severity was extracted.	No severity was extracted, the extracted string does not represent any severity.
Score	The correct score was extracted.	An incorrect, higher score was extracted.	An incorrect, lower score was extracted.	No score was extracted.

Continued on next page

Table 4: Coding Used for the Extractor Performance Evaluation

Refer- ences	All references to other (external) advisory detail pages were extracted, only a minimum (<10%) of unrelated references were extracted	All references to other (external) advisory detail pages were extracted, but some (<50%) other links were extracted too	Not all references to other (external) advisory detail pages were extracted, and/or the majority (>50%) of extracted links are unrelated	No references were extracted.
-----------------	---	---	--	-------------------------------

Table 4: Coding Used for the Extractor Performance Evaluation

## Bibliography

- [1] Alavizadeh, H., Jang-Jaccard, J., Enoch, S.Y., Al-Sahaf, H., Welch, I., Camtepe, S.A., Kim, D.D.: A Survey on Cyber Situation-awareness Systems: Framework, Techniques, and Insights. *ACM Computing Surveys* **55**(5), 107:1–107:37 (Dec 2022). <https://doi.org/10.1145/3530809>, <https://dl.acm.org/doi/10.1145/3530809>
- [2] Barford, P., Dacier, M., Dietterich, T.G., Fredrikson, M., Giffin, J., Jajodia, S., Jha, S., Li, J., Liu, P., Ning, P., Ou, X., Song, D., Strater, L., Swarup, V., Tadda, G., Wang, C., Yen, J.: Cyber SA: Situational Awareness for Cyber Defense. In: Jajodia, S., Liu, P., Swarup, V., Wang, C. (eds.) *Cyber Situational Awareness*, vol. 46, pp. 3–13. Springer US, Boston, MA (2010). [https://doi.org/10.1007/978-1-4419-0140-8\\_1](https://doi.org/10.1007/978-1-4419-0140-8_1), <http://link.springer.com/10.1007/978-1-4419-0140-8%5F1>, series Title: *Advances in Information Security*
- [3] Cheikes, B., Waltermire, D., Scarfone, K.: Common Platform Enumeration: Naming Specification Version 2.3. Tech. Rep. NIST Internal or Interagency Report (NISTIR) 7695, National Institute of Standards and Technology (Aug 2011). <https://doi.org/10.6028/NIST.IR.7695>, <https://csrc.nist.gov/pubs/ir/7695/final>
- [4] Endsley, M.R.: Situation awareness global assessment technique (SAGAT). In: *Proceedings of the IEEE 1988 national aerospace and electronics conference*. pp. 789–795. IEEE (1988), <https://ieeexplore.ieee.org/abstract/document/195097/>
- [5] Erbacher, R.F., Frincke, D.A., Wong, P.C., Moody, S., Fink, G.: A Multi-Phase Network Situational Awareness Cognitive Task Analysis. *Information Visualization* **9**(3), 204–219 (Sep 2010). <https://doi.org/10.1057/ivs.2010.5>, <http://journals.sagepub.com/doi/10.1057/ivs.2010.5>
- [6] Kaufhold, M.A., Bäuml, J., Koukal, N., Reuter, C.: AdvisoryHub: Design and Evaluation of a Cross-Platform Security Advisory System for Cyber Situational Awareness. In: *Proceedings of the 20th International Conference on Availability, Reliability and Security (ARES 2025)*. Ghent, Belgium (2025)