2021 Reconciling Data Analytics, Automation, Privacy, and Security: A Big Data Challenge (RDAAPS) | 978-1-7281-6937-8/20/$31.00 ©2021 IEEE | DOI: 10.1109/RDAAPS48126.2021.9452029

Leveraging External Data Sources to Enhance Secure System Design

Joe Samuel[1], Jason Jaskolka[1], George O. M. Yee[1,2]
[1]Systems and Computer Engineering, Carleton University
1125 Colonel By Drive, Ottawa ON K1S 5B6, Canada
[2]Aptusinnova Inc., Ottawa, ON K1G 5N7, Canada
Email: {joe.samuel,jason.jaskolka}@carleton.ca, george@aptusinnova.com

***Abstract*—Today's software systems are riddled with security vulnerabilities that invite attack. We envisage a secure software design process at the architectural level, in which the security requirements are adequate, thus enabling appropriate security controls to be implemented to mitigate known threats and vulnerabilities. How can we ensure that the security requirements are adequate? In this paper, we tackle this question by focusing on how external online data sources for vulnerabilities, attack patterns, threat intelligence, and other security information can be leveraged, using Natural Language Processing (NLP), to produce a report to assist designers in validating the adequacy of the security requirements. This validation is done by determining which requirements map to known threats (identified from the external data), which requirements may be extraneous, and which threats may need a closer look to identify new requirements. We first describe the availability and nature of the external data, followed by how we employ NLP to process the data and produce the report. We include an illustrative example of our approach.**

***Index Terms*—security, design, adequate security requirements, vulnerability data, attack patterns, threat intelligence**

I. Introduction

Software systems today are riddled with vulnerabilities that invite exploitation by attackers. To reinforce this statement, consider, for instance, the Verizon 2020 Data Breach Investigations Report [1], which indicates that web applications were involved in 43% of 3,950 data breaches worldwide. Further, the top hacking methods used in retail industry breaches were 45% stolen credentials, 40% vulnerability exploits, and 15% brute force. The prevalence of security vulnerabilities in modern software systems makes the task of developing secure software especially challenging. A common challenge is the rush to market that commercial development teams face, leaving very little time to design software that is secure as well as functional. Another challenge is that the average development team lacks the know-how and the tools to create secure software. A third challenge is that there has not been sufficient motivation for developers to produce secure software. If an application fails due to a security flaw, the application's vendor is not subject to legal penalties, unlike the construction company for a bridge that collapses due to a mechanical design flaw.

In this work, we tackle the above challenge of lack of security know-how and tools by supporting the design process with an approach for ensuring that security requirements are adequate, i.e., the requirements will lead to security controls that mitigate known threats and vulnerabilities. Having an adequate set of requirements can help designers in building systems that have appropriate controls capable of mitigating the threats that actually affect the system components. We leverage available online external data, such as databases of vulnerabilities, threats, and attack patterns, to assist designers in validating the adequacy of their security requirements. This validation is done by determining which threats are mitigated by existing security requirements, which requirements may be extraneous, as well as which threats are not yet mitigated by any existing security requirements and require a more detailed look in terms of the controls and design decisions made about how best to mitigate the threats/vulnerabilities, leading to modified or new requirements. For example, if the system under development is a server, and external vulnerability data shows that certain server links are vulnerable to man-in-the-middle attacks, a new security requirement could be added indicating that those links need to be encrypted, if this requirement does not already exist. The matching done during validation would identify whether or not this requirement is already present. The output of our approach is a report that supports the architecture design phase of the software development lifecycle (SDLC) in terms of security evaluation and assurance activities, by assisting designers in ensuring that there are adequate requirements to mitigate known threats based on their design decisions. An important benefit of our approach is that it is carried out at the architecture design stage early in the SDLC, when it is less costly to make changes, and when the negative impacts of poor design decisions that become multiplied at the code level can be avoided.

The principal contributions of this paper are: (1) identifying the external data sources that provide useful threat and vulnerability information, (2) describing the nature of these data sources, and (3) describing how the data from these sources can be used to ensure adequate security requirements.

The rest of the paper is organized as follows. Section II outlines the problem that we aim to solve. Section III provides an overview of external data sources that contain security-relevant information. Section IV describes our approach. Section V provides a discussion and Section VI presents the related work that situates our contributions in the current state-of-the-art. Lastly, Section VII concludes and highlights our future work.

II. The Problem

Decision support is an essential aspect of designing secure software-intensive systems. Designers are tasked with ensuring that adequate security controls to protect critical system assets are built into the systems that they develop. This demands informed decision-making processes to develop flexible and adaptive design solutions to ensure that these systems will continue to protect these critical assets as the threat landscape continuously evolves. Moreover, design decisions to balance tradeoffs among competing, and sometimes contradictory, system qualities need to be made during system development [2]. These decisions need to be sufficiently documented and justified to adequately support security assurance efforts.

Providing strong security assurance requires a thorough understanding of the security requirements, threats, and controls so that they can be incorporated at all stages of development [3]. Threat modeling is often an essential part of eliciting security requirements and understanding possible vulnerabilities and attack scenarios for systematically analyzing a probable attacker's profile, the most likely attack vectors, and the assets most desired by an attacker, which thereby enables informed decision-making about security risk [4], [5]. However, there are several prominent challenges in relying on threat modeling alone. It is often difficult to determine whether identified threats are rooted in real-world exploits or vulnerabilities. The number and types of threats identified often depend on the security knowledge and experience of the person(s) doing the threat modeling. Subjectivity and bias can lead to unsound decisions for mitigating identified threats [6]. This is compounded by the fact that there are many threat modeling methods and there is a lack of guidance as to which method is best for a particular application.

To overcome some of these challenges, we need to determine how data-driven approaches that leverage external online data sources, such as vulnerability databases, can support system developers in making more objective decisions to improve the security of their designs. In particular, how can this data be used to identify targets, weak patterns, and vulnerabilities during system design? Our answer in this work is to use the data to validate that the security requirements are adequate, and if not, modify or add requirements based on the data to ensure that they are adequate. To the best of our knowledge, leveraging the external data in this manner has not been done.

III. External Data Sources

In this section, we explore different external sources of security-related data that can be leveraged to support secure system design activities. The following categories of data sources were chosen based on their public availability and widespread use, but they are by no means exhaustive of the types of data that can be leveraged in our work.

A. Vulnerability Databases

Several organizations have compiled databases of security threat and vulnerability data. These databases primarily include Common Vulnerabilities and Exposures (CVE) which is a list of publicly known security vulnerabilities [7]. Each CVE vulnerability has a unique identification (ID) number, a description, and at least one public reference.

Another source of vulnerability data is the Common Weakness Enumeration (CWE) [8]. CWE is a community-developed list of common software and hardware weaknesses that can be found in software or hardware implementation, code, design, or architecture that if left unaddressed could result in systems, networks, or hardware being vulnerable to attack. Note that in CWE a *weakness* is different from a *vulnerability* in that a weakness is an error that can lead to vulnerabilities.

A popular source of CVE data is the United States National Vulnerability Database (NVD) [9]. The NVD builds upon the information included in CVE entries and provides enhanced vulnerability information in the form of standardized and analytical attributes [10]. Standardized attributes include a unique id and a detailed CVE description. The CVE description provides details such as the name of the affected product and vendor, a summary of affected versions, the vulnerability type, the impact, the access required by an attacker to exploit the vulnerability, and the code or inputs involved. Analytical attributes include the severity score provided by the Common Vulnerability Scoring System (CVSS) [11] which is considered the de facto standard for quantifying and assessing the severity and risk of security vulnerabilities in computing systems.

The Information Marketplace for Policy and Analysis of Cyber-Risk & Trust (IMPACT) [12] is an information exchange website offering cyber-risk data sets and tools for the benefit of the research community. The data and tools are contributed to IMPACT by various individuals and organizations who continue to retain ownership of their contributions. Data and tools may be requested under different categories such as cyber attack and cyber crime for different years.

The WhiteSource Vulnerability Database [13] focuses on open-source vulnerabilities. It provides comprehensive for-fee services for open-source developers that not only include vulnerabilities but also fixes for the vulnerabilities. It covers over 200 programming languages and over 3 million open-source components, aggregating information from sources such as NVD and security advisories many times a day.

Rounding out the above databases are additional vulnerability databases [14], as follows. The Vulnerability Notes Database run by the Computer Emergency Response Team (CERT) of the Carnegie Mellon Software Engineering Institute provides information on vulnerabilities such as issue summaries, affected vendors, remedial actions, and technical details. The Exploit Database run by Offensive Security includes a searchable list with information on verified and unverified vulnerabilities. Each entry in this database includes any known ID numbers, platforms affected, exploit type, and code that can be used in penetration testing. The Vulnerability Lab open-source database provides information on vulnerability language, type, severity level, exposure volume, and ideas for remediation. Vulnerabilities can be searched using CVE or project name. Finally, the Vulnerability Database (VulDB), another open-source database, has over 140,000 entries and provides information on affected vendors, classification and ID, vulnerability type, timelines, and exploit prices (prices of the exploits on the black market). Entries may also contain threat intelligence and mitigations that can be purchased.

Data delivery from the above sources is typically accomplished through APIs and HTTP. Data formats can be CSV, HTML, JSON, Text, and XML, among others, and depending on the source. For example, CVE and NVD deliver their data using APIs and HTTP. In the fall of 2019, NVD began offering web services (e.g., HTTP GET) that contain queries to filter the data retrieved for use by applications [15]. This allows better management of the size of a data retrieval. CVE offers data in the following formats: CSV, HTML, Text, XML. NVD offers data in JSON and XML formats.

Many of the above sources are intertwined in various ways. For example, CWE uses NVD which is based on CVE. The WhiteSource Vulnerability Database is partially based on NVD. This could be a good reason to focus on NVD which is an enhanced version of CVE.

B. Attack Patterns and Adversarial Behavior Databases

Attack patterns describe the common attributes and approaches employed by adversaries to exploit known weaknesses within systems. They derive from the concept of design patterns, defining the recurring challenges that adversaries may face when conducting an attack and how they go about solving it. Attack patterns are generated from detailed analyses of specific examples of real-world exploits. Each attack pattern typically describes the mechanisms used in an attack and provides potential mitigations to avoid exploits and reduce their effectiveness. By considering such an adversarial perspective to attacks, system designers can better understand patterns across attacks such as common exploits or techniques used. Additionally, this type of information can help system designers better characterize the spread of an attack on their systems following its initial exploitation.

One of the most widely-known attack pattern data sources is MITRE's Common Attack Pattern Enumeration and Classification (CAPEC) [16]. CAPEC provides a publicly available catalog of common attack patterns including those for SQL Injection, Cross-Site Scripting, and Buffer Overflow, just to name a few. Each attack pattern in the CAPEC catalog includes a description of the attack, a list of related attack patterns, associated domains of attack, and mechanisms of attack, an execution flow describing how to conduct the attack, a list of perquisites that serve as preconditions for attack success, and a list of mitigations that can be implemented to reduce the attack's effectiveness. The CAPEC catalog can be downloaded (using HTTP) in various formats including CSV, XML, and a rendered HTML representation.

Another source of attack patterns and adversarial behavior data is the MITRE's Adversarial Tactics, Techniques & Common Knowledge (ATT&CK) framework [17]. ATT&CK is mostly focused on network defense and describes the operational phases undertaken by adversaries both pre- and post-exploit such as persistence, lateral movement, and exfiltration.

ATT&CK consolidates information gathered through community contribution and MITRE's research and security activities such as penetration testing and red-teaming. It is important to note that ATT&CK does not intend to provide a comprehensive list of techniques, but rather an approximation of publicly available information for them. As such, for each identified attack technique, it captures the name of the attack technique, the tactic employed (e.g., *persistence* when the adversary is trying to maintain their foothold), the platform(s) affected (e.g., Windows), the level of permissions required for the technique, any associated techniques identified by their IDs, a brief explanation of this tactic and its potential effects, how to detect the technique, a list of potential mitigations for the technique, a list of known adversaries using the technique, and a list of additional references to information relating to the technique. In particular, there are 14 high-level tactic categories used to classify techniques which are further broken down into sub-techniques. This granularity in technique classification can help in developing a standardized description of attacks for easier review by stakeholders. ATT&CK delivers its data through the use of APIs and typically returns the results in a JSON data format.

Both CAPEC and ATT&CK provide details about adversarial behavior, each focussing on specific sets of use cases. CAPEC is more focused on application-level security, whereas ATT&CK is more focused on network defense, APTs, and threat hunting activities. Both CAPEC and ATT&CK can be cross-referenced in scenarios where CAPEC attack patterns are used by adversaries through specific techniques described in ATT&CK. This information enables contextual understanding and helps to understand how adversaries succeed in executing their objectives through various attack mechanisms. In turn, this information can be leveraged by system designers to better understand the ways in which the systems that they are designing may be attacked and therefore can help in identifying security requirements that can mitigate such attacks.

C. Threat Intelligence Data Sources

Threat intelligence is evidence-based knowledge that is obtained from threat information that has been aggregated, transformed, analyzed, interpreted, or enriched to provide the necessary context for decision-making processes [18]. Threat intelligence data feeds are a critical part of modern cybersecurity efforts, providing several useful threat information attributes and relationships including threat actors, indicators of compromise, adversarial behaviors such as attack patterns, attack campaigns, and attacker tools, observed data related to system assets, possible courses of action in response to the threat, and vulnerabilities that could enable the realization of the threat. These data feeds can be delivered using different mechanisms and data formats including APIs and fixed-interval email subscriptions, in CSV or JSON formats [19].

The Structured Threat Information eXpression (STIX) standard is considered as the most widely adopted standard for describing threat intelligence data [20]. STIX is a language and serialization format designed to support many different cyber threat management capabilities including analyzing cyber threats, specifying indicator patterns, managing response activities, and sharing cyber threat information [21]. Its primary objective is to enable sharing machine-readable cyber threat intelligence to allow for better understanding of potential security threats and attacks. STIX *objects* are represented in JSON and categorize each piece of information with specific attributes to be populated. They are connected via *relationships* that characterize cyber threat intelligence information [22].

Consumers of STIX entries subscribe to a Trusted Automated Exchange of Intelligence Information (TAXII) server [23]. TAXII is an application protocol specifically designed to support the exchange of cyber threat intelligence represented in STIX [24]. Numerous commercial vendors of such threat intelligence data feeds exist and include IBM X-Force Exchange [25], Anomali ThreatStream [26], and FireEye iSIGHT Intelligence [27], just to name a few. Open-source threat intelligence feeds are also available. These feeds are compiled by open-source communities that monitor and aggregate threat-related data from activities such as vulnerability scanning or spam emails. Examples of open-source threat intelligence feeds include ProofPoint's Emerging Threats [28] and AlienVault's Open Threat Exchange (OTX) [29].

Threat intelligence data can be used at the design-level to help understand what needs to be protected in a system. It can assist in identifying potentially critical assets that may not have been initially perceived as vulnerable. This can inform the elicitation of security requirements for which design solutions can be developed to ensure their satisfaction. The primary difficulty in leveraging threat intelligence data at the design-level is in the analysis and translation of threat intelligence information into actionable security and design requirements. Tool support capable of automating these tasks is essential.

D. Alerts and Advisories From Government Agencies

Numerous government agencies around the world including the Canadian Centre for Cybersecurity (CCCS) [30], the United States Cybersecurity & Infrastructure Security Agency (CISA) [31], the European Union Computer Emergency Response Team (CERT-EU) [32], the United Kingdom National Cyber Security Centre [33], the Australian Cyber Security Centre [34], and the New Zealand Computer Emergency Response Team (CERT-NZ) [35] publish alerts, advisories or notices regarding emerging threats and vulnerabilities. The purpose of these communications is to raise awareness of recently identified threats and vulnerabilities that may impact organizational assets and to provide additional detection and mitigation advice to recipients. Generally speaking, these alerts and advisories are delivered by email subscription or RSS feed subscription.

Alerts typically include a unique ID, a brief summary of the incident, threat, or vulnerability, technical details, and lists of products affected, additional risks and vulnerabilities, mitigations, and additional resources related to the incident, threat, or vulnerability. The list of mitigations is often categorized by topic areas such as Plans and Policies, and Best Practices related to the nature of the incident, threat, or vulnerability (e.g., Network, Ransomware, Malware, etc.). In some cases, alerts reference specific CVEs. In the CISA feeds [31], sometimes alerts also contain a MITRE ATT&CK profile for the incident, threat, or vulnerability.

Advisories are much less detailed than alerts and typically provide only a short description of often product-specific vulnerabilities. When an advisory details a recent security incident, several suggested actions are also listed. The suggested actions are often very high-level guidance and best practices such as implementing two-factor authentication (2FA) or conducting a vulnerability analysis. Despite not providing detailed information, such advisories can give hints to system designers about requirements and controls to consider.

Similar to threat intelligence, the information provided in alerts and advisories can be used by system designers to identify potentially critical assets and can be a source of information for data-driven design solutions that enable the dynamic selection of automated security controls. However, it can be difficult to automatically extract information in a machine-readable form from these alerts and advisories due to their varying levels of detail and granularity. This can often pose challenges for system designers to obtain actionable steps from these sources unless the alert or advisory is specifically related to systems or products that are part of the envisaged design.

E. Summary

There are a variety of external online data sources that provide different types of information that can be leveraged to support secure system design activities. Table I categorizes and summarizes these sources, the information they provide, the methods of data delivery, and the data formats.

IV. Approach

In this section, we present an approach that employs NLP on the external data and the existing security requirements for an asset of the system to ensure adequate security requirements.

We first introduce an example system that will serve as a running example in this section. The system is based on an OWASP example of a college library web application [36] that provides online services to students, staff, and librarians for searching and requesting books. Librarians can also add books and add users. We use a UML class diagram to describe the high-level architecture model of the web application as shown in Fig. 1, where software components are represented by classes, relationships between these components are represented by associations, and model annotations describe a set of initial security requirements (blue boxes) and design requirements (red boxes) to mitigate a set of anticipated security threats.

Our approach consists of five steps as depicted in Fig. 2.

1) Capture Security Requirements and Technologies: For a given asset, we first capture its security requirements and design requirements (where available) from the system documentation, which may be in the form of specifications or

TABLE I
SUMMARY OF EXTERNAL ONLINE DATA SOURCES CONTAINING KNOWN THREAT, VULNERABILITY AND ATTACK INFORMATION

Category	Sources	Information Provided	Delivery Method	Data Format
Vulnerability Data	NVD, CVE, CWE, IMPACT, WhiteSource, Notes, Exploit, Vulnerability Lab, VulDB	Vulnerabilities, weaknesses, affected product, affected vendor, attacker access, severity score, applicability definitions, how exploit, risk, tools, fixes, impact, issue summaries, code for penetration testing, vulnerability timeline, exploit prices, threat intelligence	API, HTTP, RSS, email subscription	CSV, HTML, JSON, Text, XML
Attack Pattern Data	CAPEC, ATT&CK	Attack patterns (approaches) and attributes (e.g., description, mechanism, related attack patterns, execution flow, preconditions, mitigations), TTPs (tactics, techniques, and procedures used in advanced persistent threats (APT))	API, HTTP	CSV, HTML, JSON, XML
Threat Intelligence	X-Force Exchange, ThreatStream, iSIGHT Intelligence, Emerging Threats, Open Threat Exchange	Evidence-based threat knowledge and relationships including threat actors, indicators of compromise, adversarial behaviors, possible responses to threats, vulnerabilities to threats	API, email subscription	CSV, JSON, Text
Alerts & Advisories	CCCS, CISA, CERT-EU, United Kingdom National Cyber Security Centre, Australian Cyber Security Centre, CERT-NZ	Alerts: unique identifier, brief summary of incident (or threat or vulnerability), technical details, products affected, other risks and vulnerabilities, mitigations, other related resources; Advisories: short description of product-specific vulnerabilities, suggested actions	RSS, email subscription	HTML, Text

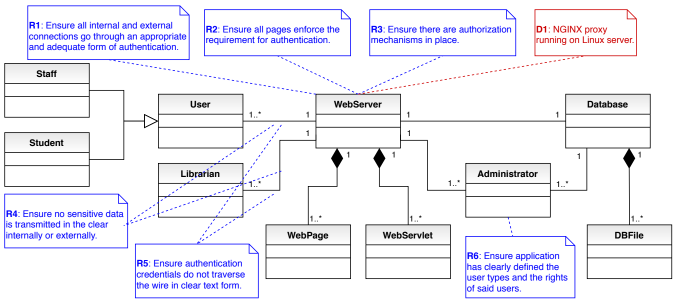

R1: Ensure all internal and external connections go through an appropriate and adequate form of authentication.
R2: Ensure all pages enforce the requirement for authentication.
R3: Ensure there are authorization mechanisms in place.
D1: NGINX proxy running on Linux server.
Staff
Student
User
WebServer
Database
Librarian
Administrator
WebPage
WebServlet
DBFile
1..*
1
R4: Ensure no sensitive data is transmitted in the clear internally or externally.
R5: Ensure authentication credentials do not traverse the wire in clear text form.
R6: Ensure application has clearly defined the user types and the rights of said users.

Fig. 1. High-level architecture of the college library web application annotated with initial security requirements (in blue) and design requirements (in red)

design models. This information helps us determine, among other things, asset characteristics such as required security controls, and associated technologies. In what follows, we assume that an annotated model is available in a documented form providing the requirements so that they can be formulated as a JSON query that can be used in the next steps. Considering the WebServer asset in our running example, we formulate the query as shown in Listing 1 which captures the security requirements (R1, R2, and R3 in Fig. 1) and the technology-related design requirements (D1 in Fig. 1).

```
{
  "requirements": "Ensure all internal and external
       connections in server go through an appropriate and
       adequate form of authentication. Ensure all pages
       enforce the requirement for authentication. Ensure
       there are authorization mechanisms in place.",
  "technologies": "NGINX proxy on Linux server"
}
```

Listing 1. JSON query for the college library application WebServer asset

2) Keyword Extraction: In this step, we process the JSON query formulated in Step 1 using an NLP model. To extract appropriate keywords, we pass the requirements text through a pre-trained NLP model that identifies lemmas (grouping inflectional and derivational forms of a word into its base form) in the text and extracts named entities, part-of-speech tags (such as verbs and nouns), and syntactic dependencies. Lemmatization helps with capturing appropriate keywords even with inflectional forms of the requirements. We then filter the identified keywords using a rule-based matcher to ensure we return only the keywords relevant to the cybersecurity context. The result from this step is a list of keywords representing the security and design requirements provided by the designer.

In our running example, the keywords extracted based on the security requirements and technology-related design requirements from Listing 1 are **authentication**, **authorization**, **NGINX**, and **Linux**. This list of keywords will be used to gather relevant threats from the various external data sources to identify the relationships between the identified threats, security requirements, and design requirements. The more accurate the extraction of the keywords, the more precise our search and analysis will be for the asset analysis. While a generic off-the-shelf NLP model yielded usable results, it also extracted information that was not relevant to our analysis. For this reason, we developed a rule-based method to further filter relevant security control-related and technology-related keywords from the designer-provided data.

3) Obtain Threat Information: Given the keywords extracted in Step 2, we query a selection of external data sources from Section III. This helps us gather the relevant threat information needed for the designer to identify the relationships between known threats and their security and design requirements. As each data source presents its data using a different schema, we format the results into a uniform schema. The result of this step is a JSON object containing all the threats separated based on their source. For our running example, we use NVD, ATT&CK, and CCCS Alerts and Advisories, although any number of external data sources could be used. Listing 2 shows the query response schema for the technology keyword "NGINX" for the WebServer asset in our example. The *meta_data* includes information such as the API version, timestamp of the query, and other query-specific details. The *statistics* contain additional information reflecting the outcomes of the analysis performed in Step 4.

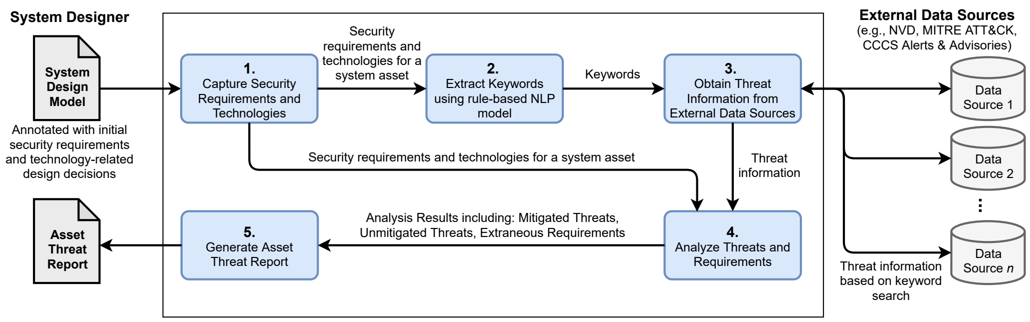

System Designer
System Design Model
Annotated with initial security requirements and technology-related design decisions
Asset Threat Report
1. Capture Security Requirements and Technologies
Security requirements and technologies for a system asset
2. Extract Keywords using rule-based NLP model
Keywords
3. Obtain Threat Information from External Data Sources
External Data Sources (e.g., NVD, MITRE ATT&CK, CCCS Alerts & Advisories)
Data Source 1
Data Source 2
Data Source n
Threat information based on keyword search
Security requirements and technologies for a system asset
Threat information
4. Analyze Threats and Requirements
Analysis Results including: Mitigated Threats, Unmitigated Threats, Extraneous Requirements
5. Generate Asset Threat Report

Fig. 2. Overview of the proposed analysis approach which leverages external data sources to validate the adequacy of a set of security requirements

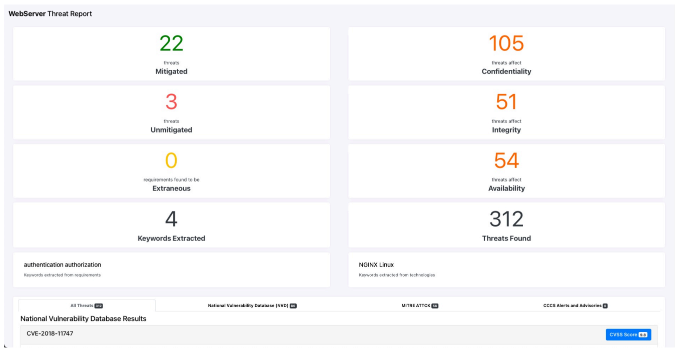
```
{
  "meta_data": {...},
  "statistics": {...},
  "results": [ {
    "NGINX":{
      "nvd_results":{"meta_data":{}, "threats":[...]},
      "mitre_attck_results":{...},
      "cccs_alerts_results":{...}
    } } ]
}
```

Listing 2. JSON search response for the WebServer asset keyword "NGINX"

4) Analyze Threats and Requirements: In this step, we identify the various relationships that exist between the identified threats, security requirements, and design requirements. We first enrich the threat information for each threat received with knowledge inferred from this information describing the threat's impact on security objectives and the various security controls it prescribes. The security objectives information is determined using NLP extractors designed to scan for cues within the threat description to identify its impact on one or more security objectives. Where explicit information of the threat's impact on security objectives is available such as in a threat's CVSS specification, we use that information instead for our classification. For simplicity, we consider the confidentiality, integrity, and availability objectives, but other objectives such as authenticity and accountability may be considered as well. The security controls prescribed by the threats are extracted using another variation of an NLP extractor configured to recognize technical controls defined in ISO/IEC 27001 [37]. The extracted list of controls is then compared with the initial security requirements to yield (1) identified threats addressed by the provided security requirements (i.e., *mitigated threats*), (2) identified threats for which there are currently no security requirements (i.e., *unmitigated threats*), and (3) provided security requirements for which no relevant threats were found (i.e., potentially *extraneous requirements*). For instance, to identify unmitigated threats, we compute the set difference between the set of security requirements extracted from the external threat data related to the design requirements and the set of security requirements provided by the designer. The result is the design-related threats for which a new or modified security requirement may be required.

The results of this step are stored within the *statistics* attribute of our response shown in Listing 2 and will be used to generate the report described in the following step.

5) Generate Report: An interactive report is generated to present the results obtained in Step 4. For our college library WebServer asset, the generated report is shown in Fig. 3. The main goal of this report is to help the designer navigate through an otherwise overwhelming amount of information in a more feasible manner.



WebServer Threat Report

22
threats
Mitigated

105
threats affect
Confidentiality

3
threats
Unmitigated

51
threats affect
Integrity

0
requirements found to be
Extraneous

54
threats affect
Availability

4
Keywords Extracted

312
Threats Found

authentication authorization
Keywords extracted from requirements

NGINX Linux
Keywords extracted from technologies

All Threats 312 | National Vulnerability Database (NVD) 80 | MITRE ATTCK 56 | CCCS Alerts and Advisories 0

National Vulnerability Database Results

CVE-2018-11747 CVSS Score 9.8

Fig. 3. Threat report generated for the college library WebServer asset

The threat analysis report for our college library WebServer asset in Fig. 3 shows that out of 312 threats discovered based on the initial security and design requirements, 105 threats affect confidentiality, 51 threats affect integrity, and 54 threats affect the availability of the WebServer. While these counters are not a solution on their own, they help the designer gauge the extent to which the various security objectives of the asset are impacted by known threats during the system design phase. It is important to note that the 312 total threats refer to threats for which there is sufficient information to be useful for the designer to act on. The report also provides a breakdown of the keywords extracted from the security and design requirements. Additionally, the report provides a summary of the various threat-requirement relationships described in Step 4, identifying 22 threats as mitigated, 3 threats as unmitigated, and no extraneous requirements. Here, the mitigated threats refer to the threats associated with the security controls "authentication" and "authorization". The 3 unmitigated threats are threats affecting the technologies specified in the design requirements. These threats prescribe security controls not specified in the current security requirements, namely two-factor authentication and logging. Since both security controls identified from our requirements have a number of associated threats, no extraneous threats were found. The report also provides the identified list of threats organized by their source.

The steps of the proposed approach have been implemented as a set of APIs available at https://compass.carleton.ca/api/. Anyone wishing to adopt this approach and use the developed NLP models can generate a similar report by connecting to the asset threat report API.

V. Discussion

Our approach assists system designers in identifying known threats from external data sources which may warrant modified or new requirements to enhance the security of the system design. Our approach also helps in identifying potentially extraneous requirements that do not appear to address or mitigate any known threats and thereby may not be necessary for the system. This type of information enables the designer to prioritize the unmitigated threats, especially under time and budget constraints, improving development efficiency. While some identified threats may not apply to the system, our approach helps distill large amounts of data into an actionable form for the designer. For example, results from such external data sources given a keyword such as NGINX can commonly exceed tens of thousands of threats and is unreasonable for designers to sift through for each asset of their system. To reduce this volume of data, our approach presents only the threats that affect the asset for which no security controls have been mentioned within the security requirements. Therefore, our approach provides much-needed designer support in these areas.

In developing the proposed approach we faced several challenges. One challenge was the aforementioned data size. We developed a script to automate the process of classifying and categorizing threats from external sources into a database to minimize the response time of the results. Another challenge was the inconsistencies with the data from the various sources. We developed a unified, simple schema as shown in Listing 2 to capture sufficient information in a flexible format for further analysis and applications. The approach is dependent on the quality and detail of the security and design requirements provided as input. If these inputs are poor, the approach will not yield quality results. Lack of input detail will raise difficulties in the keyword extraction and subsequent threat and requirements analysis as the search queries to the external data sources may be insufficient to obtain any meaningful results.

VI. Related Work

Bakirtzis et al. [38] proposed the cyber security body of knowledge (CYBOK) as an algorithmic way to explore vulnerabilities. Within CYBOK they used a system model represented as a graph with vulnerabilities input from CVE, CWE, and CAPEC to identify potential attack vectors. While CYBOK focuses on identifying the overall security posture of systems using a system model, our approach focuses on ensuring the adequacy of security requirements during the system design phase by leveraging external data sources without requiring a complete system model. Elahi et al. [39] presented a goal model with which to evaluate the effects of vulnerabilities and countermeasures with a view to requirements elicitation. However, this work does not directly leverage external data sources for dynamically identifying unmitigated threats. Rather, it proposes a formal framework to support the requirements elicitation process by using vulnerability information as a means to develop countermeasures that then form new security requirements. Feng et al. [40] focused on the impact of architectural design decisions on security. They proposed a Design Rule Spaces-based analysis approach to identify architectural design flaws and used CVEs to find correlations with *hotspot patterns* which were architectural flaws related to violations of proper design principles. Our work does not focus on architectural design decisions but rather on missing security controls found through leveraging external databases. Nerwich et al. [41] addressed the problem of lack of knowledge management systems for Internet of Things (IoT)-specific vulnerabilities and attacks. They presented the construction of a community-driven, IoT-specific database which documented the vulnerabilities and attacks on IoT infrastructures and supported integration with other vulnerability databases such as NVD. Our work does not involve developing a database to maintain threat information.

VII. Conclusion and Future Work

In this paper, we explored how external data sources, such as vulnerability databases, attack patterns, threat intelligence data feeds, and security alerts and advisories can be leveraged to assist system designers in validating the adequacy of their security requirements and to support them in making more informed design decisions. This enables them to build systems that have appropriate controls capable of mitigating the known threats and vulnerabilities that actually affect the system components. We identified and described the currently available external data sources that provide useful information for this purpose. We also presented an NLP-based approach that uses available design documentation including initial security and design requirements. The approach produces a report that summarizes which threats to the system design identified from external data are mitigated by existing security requirements, which threats are not yet mitigated by any existing security requirements, and which requirements may be extraneous. This information can enable the system designer to make more informed design decisions in terms of how best to mitigate the threats/vulnerabilities, leading to modified or new requirements. This approach demonstrates that external threat, vulnerability, and attack data can support secure software design activities and address security concerns early in the SDLC at the architecture design stage when it is less costly to make changes. As a system designer makes changes and adds new security requirements, our approach can be applied iteratively to support incremental development and improvement of the system design and therefore help to increase confidence that the last iteration of security requirements is adequate in support of security evaluation and assurance activities.

In future work, we seek to develop a machine learning-based domain-specific NRE to detect a broader range of entities and security controls. This will provide a more context-tuned cybersecurity NLP model to refine the keyword extraction and threat analysis steps of the approach. To keep the size of the processed data manageable, we aim to improve our identification of unmitigated threats and investigate better mechanisms to generate a more actionable set of results and present only the most relevant threats for the system designer's review. We also aim to improve the tool support for the approach to automatically extract the annotations from system design models, as well as additional information such as the inter-relationships and dependencies between the system assets. This will enable a more detailed analysis of the system design which could further assist system designers to build systems that are secure and resistant to a variety of threats and attacks.

Acknowledgment

This research was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) grant RGPIN-2019-06306.

References

[1] Verizon Enterprise Solutions, "2020 data breach investigations report." Available: https://enterprise.verizon.com/resources/reports/dbir/, May 2020. [Accessed: Dec-2020].

[2] C. B. Weinstock and H. F. Lipson, "Evidence of assurance: Laying the foundation for a credible security case," tech. rep., Software Engineering Institute, Pittsburgh, PA, USA, Aug. 2013.

[3] J. Jaskolka, "Recommendations for effective security assurance of software-dependent systems," in *Intelligent Computing, SAI 2020* (K. Arai, S. Kapoor, and R. Bhatia, eds.), vol. 1230 of *Advances in Intelligent Systems and Computing*, pp. 511–531, Springer, Cham, 2020.

[4] A. Shostack, *Threat Modeling: Designing for Security*. John Wiley & Sons, 2014.

[5] T. UcedaVélez and M. M. Morana, *Risk Centric Threat Modeling: Process for Attack Simulation and Threat Analysis*. John Wiley & Sons, first ed., 2015.

[6] J. Samuel, K. Aalab, and J. Jaskolka, "Evaluating the soundness of security metrics from vulnerability scoring frameworks," in *19th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, (Guangzhou, China), pp. 442–449, 2020.

[7] The MITRE Corporation, "Common vulnerabilities and exposures." https://cve.mitre.org/. [Accessed: Dec-2020].

[8] The MITRE Corporation, "Common weakness enumeration." https://cwe.mitre.org/. [Accessed: Dec-2020].

[9] National Institute of Standards and Technology, "National vulnerability database." https://nvd.nist.gov/. [Accessed: Dec-2020].

[10] Kaseya, "The national vulnerability database (nvd) explained." https://https://www.kaseya.com/blog/2020/10/22/national-vulnerability-database-nvd/. [Accessed: Dec-2020].

[11] P. Mell, K. Scarfone, and S. Romanosky, "The common vulnerability scoring system (CVSS) and its applicability to federal agency systems," NIST Interagency Report 7435, National Institute of Standards and Technology, Aug. 2007.

[12] IMPACT, "Information marketplace for policy and analysis of cyber-risk & trust." https://www.impactcybertrust.org/. [Accessed: Dec-2020].

[13] WhiteSource Software, "Whitesource vulnerability database." https://www.whitesourcesoftware.com/vulnerability-database/. [Accessed: Dec-2020].

[14] Cornell University, "Top vulnerability databases for open-source components." https://blogs.cornell.edu/react/top-vulnerability-databases-for-open-source-components/. [Accessed: Dec-2020].

[15] B. Byers and H. Owen, "Automation support for CVE retrieval." https://csrc.nist.gov/CSRC/media/Projects/National-Vulnerability-Database/documents/web%20service%20documentation/Automation%20Support%20for%20CVE%20Retrieval.pdf. [Accessed: Jan-2021].

[16] The MITRE Corporation, "Common attack pattern enumeration and classification." https://capec.mitre.org/. [Accessed: Dec-2020].

[17] B. E. Strom, A. Applebaum, D. P. Miller, K. C. Nickels, A. G. Pennington, and C. B. Thomas, "MITRE ATT&CK: Design and philosophy," White Paper MP180360R1, The MITRE Corporation, Mar. 2020.

[18] C. Johnson, L. Badger, D. Waltermire, J. Snyder, and C. Skorupka, "Guide to cyber threat information sharing," Special Publication (NIST SP) 800-150, National Institute of Standards and Technology, Oct. 2016.

[19] M. Bromiley, "Threat intelligence: What it is, and how to use it efectively," white paper, SANS Institute, Sep. 2016.

[20] C. Sillaber, C. Sauerwein, A. Mussmann, and R. Breu, "Data quality challenges and future research directions in threat intelligence sharing practice," in *2016 ACM on Workshop on Information Sharing and Collaborative Security*, pp. 65–70, 2016.

[21] S. Barnum, "Standardizing cyber threat intelligence information with the Structured Threat Information eXpression (STIX^{TM})," White Paper Version 1.1, Revision 1, The MITRE Corporation, Feb. 2014.

[22] OASIS Open, "Introduction to STIX." https://oasis-open.github.io/cti-documentation/stix/intro. [Accessed: Dec-2020].

[23] OASIS Open, "Introduction to TAXII." https://oasis-open.github.io/cti-documentation/taxii/intro.html. [Accessed: Dec-2020].

[24] J. Connolly, M. Davidson, M. Richard, and C. Skorupka, "The Trusted Automated eXchange of Indicator Information (TAXII^{TM})," white paper, The MITRE Corporation, Nov. 2012.

[25] IBM Corporation, "IBM X-Force Exchange." https://exchange.xforce.ibmcloud.com/. [Accessed: Dec-2020].

[26] Anomali, "Threatstream threat intelligence platform." https://www.anomali.com/products/threatstream. [Accessed: Dec-2020].

[27] FireEye, "FireEye iSIGHT Intelligence." https://www.fireeye.com/blog/products-and-services/2016/05/introducing_fireeye.html. [Accessed: Dec-2020].

[28] ProofPoint, "Emerging threats." https://rules.emergingthreats.net/. [Accessed: Dec-2020].

[29] AlienVault, "Open Threat Exchange." https://otx.alienvault.com/. [Accessed: Dec-2020].

[30] Canadian Centre for Cyber Security, "Alerts & advisories." https://cyber.gc.ca/en/alerts-advisories. [Accessed: Dec-2020].

[31] United States Cybersecurity & Infrastructure Security Agency, "National cyber awareness system – alerts." https://us-cert.cisa.gov/ncas/alerts. [Accessed: Dec-2020].

[32] European Union Computer Emergency Response Team, "Security advisories." https://cert.europa.eu/cert/newsletter/en/latest_SecurityBulletins_.html. [Accessed: Dec-2020].

[33] U.K. National Cyber Security Centre, "Reports & advisories." https://cert.europa.eu/cert/newsletter/en/latest_SecurityBulletins_.html. [Accessed: Dec-2020].

[34] Australian Cyber Security Centre, "Alerts." https://www.cyber.gov.au/acsc/view-all-content/alerts. [Accessed: Dec-2020].

[35] New Zealand Computer Emergency Response Team, "Alerts." https://www.cert.govt.nz/business/recent-threats/. [Accessed: Dec-2020].

[36] OWASP, "CRV2 app threat modeling." https://owasp.org/www-community/CRV2_AppThreatModeling, 2020. [Accessed: Dec-2020].

[37] International Organization for Standardization, "ISO/IEC 27000:2018 Information technology – Security techniques – Information security management systems – Overview and vocabulary," Feb. 2018.

[38] G. Bakirtzis, B. J. Simon, A. G. Collins, C. H. Fleming, and C. R. Elks, "Data-driven vulnerability exploration for design phase system analysis," *IEEE Systems Journal*, vol. 14, pp. 4864–4873, Dec. 2020.

[39] G. Elahi, E. Yu, and N. Zannone, "A vulnerability-centric requirements engineering framework: Analyzing security attacks, countermeasures, and requirements based on vulnerabilities," *Requirements Engineering*, vol. 15, pp. 41–62, Mar. 2010.

[40] Q. Feng, R. Kazman, Y. Cai, R. Mo, and L. Xiao, "Towards an architecture-centric approach to security analysis," in *13th Working IEEE/IFIP Conference on Software Architecture*, pp. 221–230, Apr. 2016.

[41] M. Nerwich, P. Gauravaram, H.-y. Paik, and S. Nepal, "Vulnerability database as a service for IoT," in *Applications and Techniques in Information Security* (L. Batina and G. Li, eds.), (Singapore), pp. 95–107, Springer Singapore, 2020.