Modeling a CyberImmune Architecture: The Case of Open Source Intelligence Systems

Kirill Ivashnev, Dmitry Korzun
Petrozavodsk State University (PetrSU)
Petrozavodsk, Russia
ivashnev@cs.petrsu.ru, dkorzun@cs.karelia.ru

***Abstract*—Software development with cybersecurity requirements attracts much attention in Internet Open-Source Intelligence (OSINT) systems due to high uncertainty, data redundancy, and unreliability in automated information extraction and analysis. In this paper, we introduce a method for making "cyber immune" the architecture for a given OSINT system. Applying the method a developer analyzes all architectural components for the vulnerability. A graph-based model is constructed based on trust level of architectural components and their impact on security metrics. Then, the most important components can be protected with specified security mechanisms, leading to the required CyberImmune architecture. Our method is demonstrated on a real complicated OSINT system (regular monitoring of open web news about geospatial territory). A unique property is that our method can be applied on early phases of system development when no much knowledge (metrics) is available in respect to code size and complexity.**

I. Introduction

Software development with cybersecurity requirements attracts much attention in Internet Open-Source Intelligence (OSINT) systems due to high uncertainty, data redundancy, and unreliability in automated information extraction and analysis. An OSINT system constitute a valuable tool for collecting and analyzing data from online sources. A particular case is news aggregation platforms capable of thematic classification, filtering, and summarization [1], [2]. The vast amount of data processed within an OSINT system leads to high information overload, which complicates the extraction of relevant and reliable content [3].

Beyond conventional risks of unauthorized access, data breaches, and manipulation, particular emphasis must be placed on the accuracy and contextual relevance of collected information. An additional challenge arises from the increasing reliance on the Artificial Intelligence (AI) technology for the classification, filtering, and aggregation of OSINT data. Such an OSINT system becomes vulnerable to adversarial influences, including data poisoning, sample substitution, and model inversion attacks, which threaten both the reliability of processing and the integrity of derived insights [4].

Existing protection methods are typically applied at the operational stage, and architectural vulnerabilities are not explicitly addressed. An alternative is the CyberImmune development approach [5], which implements the well-known Secure by Design concept. The CyberImmune approach applies the MILS approach[1] to stratify security levels [6] The CyberImmune approach employs policies from the FLASK security architecture (Flux Advanced Security Kernel) to enforce controlled inter-component interactions [7]. Nevertheless, methodological foundations for such architectures remain underdeveloped, and a key unresolved issue lies in establishing criteria for assessing vulnerable components within a given system architecture. Notably, assessing the vulnerability is problematic on early phases of system development, when no much knowledge (metrics) is available in respect to code volume and complexity.

The present research aims to address the discussed limitations of existing methods by developing a novel method for identifying and assessing vulnerable components in the architecture of a given OSINT system. As a reference case, we consider a situational monitoring system for some territory with open-source data and AI modules. The application of the proposed method to the architecture of this system demonstrates ability to detect architectural weaknesses. Building on our earlier work on digital modeling platforms for territory monitoring and smart services [8], this study extends the methodology toward security-focused architectural analysis of AI-driven OSINT systems.

The proposed method identifies vulnerable architectural components based on their trust level and impact on security metrics. The identification uses a graph-based model to assess a given system architecture. The model represents directed interactions between system components. Then, the most important components can be protected with developing specified security mechanisms, leading to the required CyberImmune architecture. Basically, the method includes the following steps.

- The architecture of a given OSINT system is represented using a weighted graph.
- A quantitative vulnerability metric is proposed using algorithms of data flow analysis.
- Vulnerable components are identified and assessed using the metric and defined security policies.

The rest of the paper is organized as follows. Section II provides an overview of work related to OSINT system se-

[1]The origin of the term "MILS" was an acronym standing for "Multiple Independent Levels of Security/Safety". Today it is used as a proper name for the approach that starts with partitioning the system under design into isolated compartments, or security domains.

curity, vulnerability assessment methods, and current research challenges. Section III briefly introduces the CyberImmune development approach. Section IV describes the assessment method for identifying vulnerable components in an OSINT system architecture. Section V analyzes properties of the proposed method using a reference case—a real complicated OSINT system for territorial monitoring with AI modules. Section VI considers method's implications, including its limitations, strengths, and scalability. Section VII summarizes the key findings of this study.

II. Related Work

Understanding the current state of research on OSINT system security and architectural vulnerability assessment is essential to position the present study. Existing literature addresses OSINT vulnerabilities, security requirements, and research challenges of OSINT development.

Prior studies have identified major risks, including unauthorized access and dissemination of sensitive information, breaches of user privacy [9], including identity theft, financial fraud [10], malware propagation, data alteration and the distribution of misleading or false information [11]. Assessing the reliability and credibility of OSINT data presents a significant challenge, as publicly available sources can be biased, incomplete, or deliberately misleading.

Studies also address the requirements for OSINT systems, the implementation of which can mitigate the aforementioned threats. Proposed measures include traditional information security mechanisms such as data encryption [12], access control and backups [13], as well as proactive monitoring and threat detection strategies [14]. These approaches are often reactive, focus on individual data elements rather than inter-component interactions and provide limited protection against architectural vulnerabilities. In our solution, we propose identifying and assessing vulnerable OSINT components at the architectural level and quantifying risk.

Recent surveys highlight key research trends in AI-OSINT, including model robustness against adversarial attacks [15], the integration of explainable AI to improve transparency and trust [16], and the ethical and social implications of large-scale OSINT deployment [17]. A particularly pressing issue is the lack of experimental validation: few works provide reproducible studies or controlled testbeds to assess robustness, security, and explainability of AI-driven OSINT. Emerging directions such as digital twins and simulation-based evaluation offer promising opportunities to bridge this gap and enable systematic verification of system reliability under realistic conditions [3].

III. Background

The CyberImmune development approach [5] implements the Secure by Design concept. An example of risk analysis, description of the security concept, and system architecture design can be found in [18]. The CyberImmune approach needs identification and assessing the vulnerability of architectural components. In this section, we apply the following formalization based on set theory, discrete mathematics, and logical implication technique.

The CyberImmune approach is applied stepwise in software development. Step 1. Define the security concept of the system. Let $A = \{a_1, a_2, \ldots, a_{N_\mathrm{A}}\}$ be a set of N_A critical system assets, $F = \{f_1, f_2, \ldots, f_{N_\mathrm{F}}\}$ be a set of N_F functional requirements, and $E = \{e_1, e_2, \ldots, e_{N_\mathrm{E}}\}$ be a set of N_E undesirable events (risks, threats) that may compromise the assets. The outcome of this stage is to identify elements demanding particular focus during the developmental process.

Step 2. Define the security goals and security assumptions to specify desired protection outcomes and the conditions under which they can be achieved. Let $G = \{g_1, g_2, \ldots, g_{N_\mathrm{G}}\}$ denote a set of N_G security goals that the system architecture must ensure. Each security goal is formulated for $N_\mathrm{AE} = N_\mathrm{A}N_\mathrm{E}$ asset–event pairs, i.e.,

$$g_i = g_i(a, e) \quad \forall a \in A, \forall e \in E \text{ for } i = 1, 2, \ldots N_\mathrm{G}.$$

Let $H = \{h_1, h_2, \ldots, h_{N_\mathrm{H}}\}$ denote a set of N_H security assumptions (hypotheses) that considered as security goals for external systems. Each security assumption is formulated for N_AE asset–event pairs, i.e.,

$$h_i = h_i(a, e) \quad \forall a \in A, \forall e \in E \text{ for } i = 1, 2, \ldots N_\mathrm{H}.$$

The following property must be satisfied in the security concept,

$$\forall g \in G \;\; \exists H_g \subseteq H \;\; \text{s.t.} \;\; H_g \Rightarrow g, \tag{1}$$

i.e., for each goal, there are assumptions that support achieving the goal.

Step 3. Design the system architecture based on separating the security domains. Domains correspond to system components and trust zones. Domains are classified into three categories: a) trusted domains D_TR, ensuring high data integrity; b) integrity-enhancing domains D_IN, providing security policies; and c) untrusted domains D_UN, assumed potentially compromised. In the system, the trusted computing base (TCB) and untrusted computing base (UCB) are combined,

$$D = D_\mathrm{TR} \cup D_\mathrm{IN} \cup D_\mathrm{UN}. \tag{2}$$

The architecture represents interactions among domains, specifying permitted and prohibited communications in the form of security policies.

Step 4. Classify the security domains according to their complexity and code size, i.e.,

$$\gamma(d) \to \{\mathrm{SS}, \mathrm{MM}, \mathrm{CL}\}. \tag{3}$$

Each security domain $d \in D$ belongs to one of the three classes: SS (simple, small), MM (medium), CL (complex, large). Domains for TCB are considered as simple and small. This rule supports verification and reduces potential architectural vulnerabilities.

Step 5. Construct the thread model of potential attacks against the security goals. Let $S = \{s_1(g), s_2(g), \ldots, s_{N_\mathrm{S}}(g)\}$ denote a set of N_S negative scenarios for $g \in G$. Any scenario

$s(g)$ provides a rationale for architectural policy decisions, particularly in selecting TCB domains.

Step 6. Refine the architectural policy based on domain classification. CL domains are to be decomposed, and code is to be relocated outside TCB when feasible.

An example is considered further in Section V. The described procedure results in the architecture augmented with security domains (2) and their complexity classes (3). Based on this knowledge, the most important components can be protected with developing additional security mechanisms.

The domain separation and classification would benefit from quantitative assessment. The next section describes the proposed method for identifying vulnerable components in a given system architecture.

IV. Our Method for Identifying Vulnerable System Components

The CyberImmune approach does not provide a mathematical model and formal procedure for identifying vulnerable components based on quantitative assessment models. In fact, experts develop security protection model (based on available code metrics and engineering reasoning)

$$(A, F, E, G(A, E), H(A, E), S(G), D)$$

to describe critical assets A, functional requirements F, undesirable events E, security goals G, security assumptions H, negative scenarios S, and untrusted and trusted domains D. The model satisfies (1), separates the computing base into untrusted and trusted (2), and shows the complexity and size of the domains (3).

A. Graph-based architectural model

Since security domains $d \in D$ one-to-one correspond to system components (modules), the system architecture can be represented as a directed graph $G_{\text{arh}} = (D, L)$, where D corresponds to nodes of the graph and L denotes the set of links (control and data flows between components).

Classification (3) needs knowledge on complexity and size of the computing base. To address this, link weights $w_l = w(d_1, d2)$ for $l = (d_1, d_2) \in L$ are introduced based on such categorical data flow characteristics as operational efficiency, data volume, and priority level, see Table I.

TABLE I. Categorial estimate of data streams

Category	Description
Operational efficiency: set boundaries $0 < \alpha_{\text{vfr}} < \alpha_{\text{frq}} < \alpha_{\text{rar}} \leq 1$	
Very Frequent (VF)	Updated in real time, in the background, with minimal delay.
Frequent (F)	Updated every few hours.
Rare (R)	Updated in specific cases.
Data Volume: set boundaries $0 < \beta_{\text{rqt}} < \beta_{\text{rsp}} \leq 1$	
Request (D1)	Data request, e.g., API or Modbus.
Response Data (D2)	Data of varying size.
Priority Levels: set boundaries $0 < \gamma_{\text{low}} < \gamma_{\text{hgh}} \leq 1$	
Priority Level L1	Minimal transmission delay - data retrieval from storage, authentication.
Priority Level L2	Delays acceptable during data update - calculations, classification, or unavailability of external data sources.

On early phases of system development no much knowledge is available in respect to the code size and complexity. An expert can estimate weight using categorical characteristics with the following rules. In advance, the expert heuristically selects boundary values α (efficiency), β (volume), and γ (priority) to model the categories of data flows, see Table I. Then for each $l \in L$ the expert selects: a) α_l from given boundary parameters $\{\alpha_{\text{vfr}}, \alpha_{\text{frq}}, \alpha_{\text{rar}}\}$, b) β_l from $\{\beta_{\text{rqt}}, \beta_{\text{rsp}}\}$, and c) γ_l from $\{\gamma_{\text{low}}, \gamma_{\text{hgh}}\}$.

The fixed boundary parameters are normalized. The total link weight is estimated as the product

$$w_l = \alpha_l \beta_l \gamma_l \text{ for } l \in L, \quad 0 < w_l \leq 1. \tag{4}$$

According to (4) higher weight is assigned to more critical data stream.

B. Domain Classification Model

To implement (3), we employ graph centrality characteristics. Our model assumption is that the vulnerability of a component $d \in D$ depends on the structural complexity of the system and on the code volume of d. Well-known graph centrality metrics are considered possible assets for the vulnerability [19].

In particular, let the closeness centrality be interpreted as measure of domain accessibility based on the weighted shortest path length.

$$C_{\text{cls}}(d) = \frac{1}{\sum\limits_{d' \neq d} \rho(d, d')}, \quad d \in D, \tag{5}$$

where $\rho(d, d')$ denotes the weighted shortest path length between nodes d and d'. The interpretation supports the observation that domains with higher accessibility in the system topology are likely to play more significant role in data transmission. Therefore, those domains correspond to critical components in terms of meeting functional requirements and maintaining stability.

In addition, we consider the betweenness centrality, which measures the share of domain involvement in data transmission between other components.

$$C_{\text{btw}}(d) = \sum_{d' \neq d \neq d''} \frac{\sigma_{d'd''}(d)}{\sigma_{d'd''}}, \quad d \in D, \tag{6}$$

where $\sigma_{d'd''}$ is the total number of weighted paths between components d' and d'' and $\sigma_{d'd''}(d)$ is the number of the paths $d' \to d''$ passing through d.

Let us introduce the composed asset based on metrics (5) and (6).

$$C(d) = \lambda_1 \sum_{l \in L(d)} w_l + \lambda_2 C_{\text{btw}}(d) + \lambda_3 C_{\text{btw}}(d), \quad d \in D, \tag{7}$$

where the coefficients $\lambda_1 > 0$, $\lambda_2 > 0$, $\lambda_3 > 0$ define the relative contribution of each factor.

Metric (7) combines the local load of d, the routing influence, and the accessibility. The contribution coefficients are set

empirically according to the system use scenario. Examples of empirical values are provided in Section V-B.

Classification (3) is implemented using a threshold function for (7).

$$\kappa(d) = \begin{cases} \text{SS}, & C(d) < \theta_1 \\ \text{MM}, & \theta_1 \le C(d) < \theta_2 \\ \text{CL}, & C(d) \ge \theta_2 \end{cases} \quad (8)$$

where the thresholds θ_1 and θ_2 define the boundaries of complexity levels. The thresholds can be set either empirically or using expert consideration (e.g., as the 33rd and 66th percentiles of the $C(d)$ distribution in graph G). Examples of threshold setting, including values derived from percentile-based distribution analysis, are shown in Section V-B.

The use of functions (7) and (8) provides a quantitative technique for ranking the domains by complexity and code size, i.e., serving as an indicator for excluding components from the TCB.

C. Guidelines for Architectural Policy

The final stage of the CyberImmune development approach establishes the architectural policy, defining domain interactions under operational constraints, usage scenarios, and TCB minimization. We derive the rules from the graph-based and domain classification models. Large and complex domains are to be restricted to non-critical data due to testing limitations and high operational costs. Small domains serving as proxy entities are to be introduced to filter data for such components.

The architectural policy development proceeds in three stages. Stage I labels all components as untrusted ($d \in D_{UN}$). Stage II applies initial TCB labeling based on security goals, with updates guided by functions (7) and (8). The key rules are the following.

- Components performing local computation are labeled as $d \in D_{UN}$;
- Components receiving data from untrusted domains are labeled as $d \in D_{IN}$;
- Data flows from $d \in D_{IN}$ are labeled as high-integrity;
- Components processing only high-integrity data are labeled as $d \in D_{TR}$.

In Stage III, domain labels are applied to the architectural diagram. CL domains assigned to D_{IN} or D_{TR} are either decomposed into smaller ones or assigned proxy entities to maintain TCB minimization.

D. Algorithm for Identifying Vulnerable Components and Defining Architectural Policy

Based on the proposed graph-based architectural and domain classification models, we formulate a stepwise algorithm to identify vulnerable system components and define the architectural policy in accordance with the CyberImmune development approach. The method encompasses graph construction, categorical evaluation of interactions, computation of weighted complexity metrics, and domain classification. Results provide the basis for component labeling and policy formation while minimizing the trusted computing base (TCB). The process for identifying vulnerable components and defining the architectural policy is outlined in Algorithm 1.

Algorithm 1 CyberImmune Architectural Analysis

1: Define sets A and E, G and H;
2: Represent the system architecture as a diagram containing domains and data flows between them;
3: **for all** $d \in D$ **do**
4: Label d as D_{UN}
5: **end for**
6: **for all** $d \in D$ **do**
7: Apply initial class label according to security goals;
8: **end for**
9: Represent the system as a graph $G_{\text{arh}} = (D, L)$;
10: **for all** links $l \in L$ **do**
11: **Determine** categorical characteristics α_l, β_l, γ_l based on boundary values
12: **Compute** edge weight w_l using (4)
13: **end for**
14: Set coefficients $\lambda_1, \lambda_2, \lambda_3$ for $C(d)$ (7)
15: Set thresholds θ_1, θ_2 for $\kappa(d)$ (8)
16: **for all** domains $d \in V$ **do**
17: **Compute** $C(d)$ using (7)
18: **Compute** $\kappa(d)$ using (8)
19: **Label** class $\kappa(d)$ on the architectural diagram
20: **end for**
21: **Define** architectural policy using values from (7), (8) and rules in Section C

V. CyberImmune Architecture for Northern Situational Monitoring System

The methodological principles of CyberImmune architecture design are illustrated using the Northern Situational Monitoring System as a case study. It should be noted that detailed implementation aspects of the monitoring platform are intentionally omitted, as they are the subject of a separate forthcoming study. Here, the case study serves only to illustrate the applicability and effectiveness of the proposed graph-based approach.

A. System Overview

The system under study is an OSINT-based platform for monitoring events within a geographically and thematically constrained domain. Its primary function is the automated collection and analysis of Northern-related publications from news sources in foreign languages.

Each source maintains a collection of materials, including news articles and reports, which are regularly updated. The system automatically scans all source types, filtering data according to geographic regions to track changes in the information environment across different Northern areas. Newly discovered materials are uploaded into the system for further analysis, with duplicate content being automatically excluded.

Users can define and modify a list of topics and associated keywords. Based on the collected data, the system generates

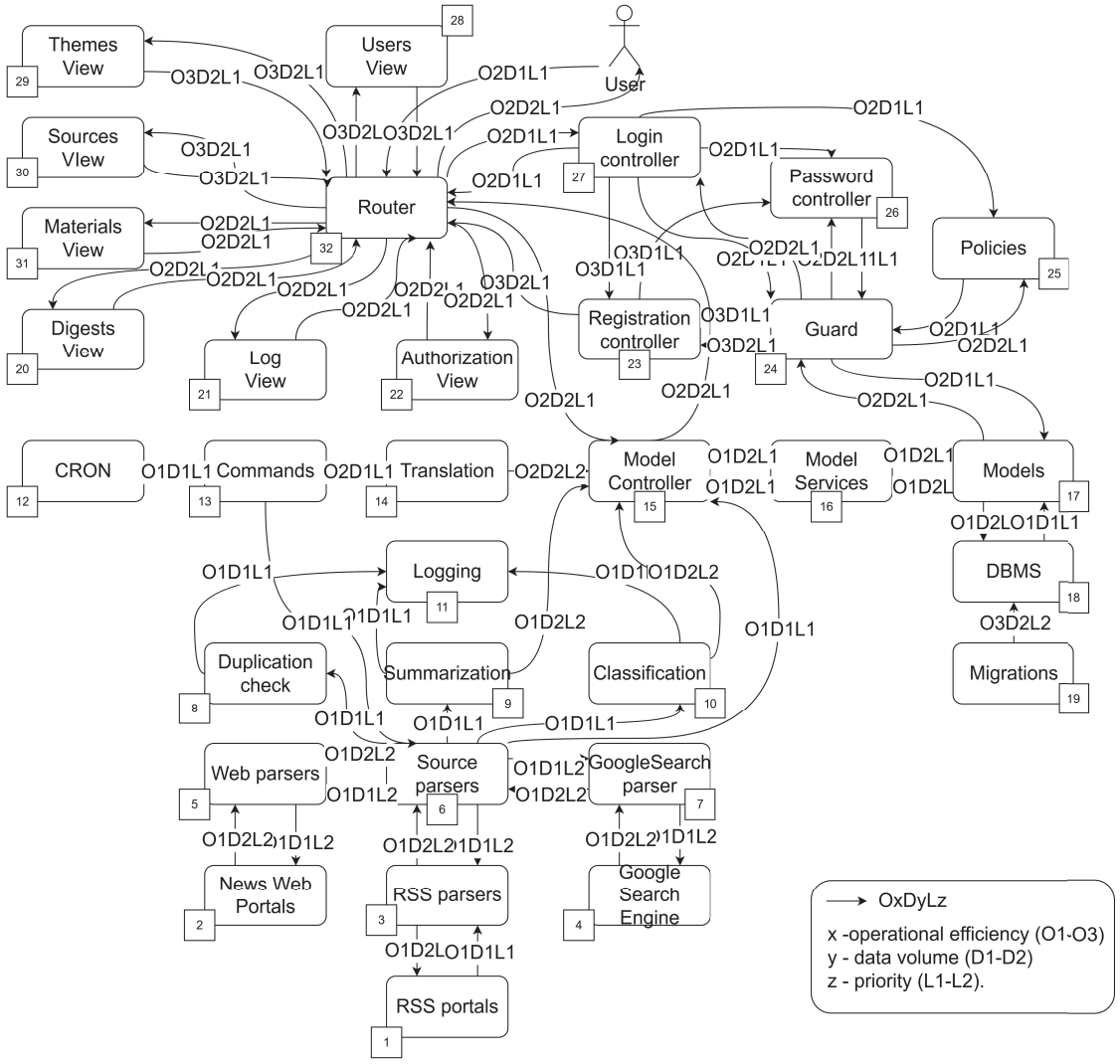

Fig. 1. The initial architecture of the system

digests for selected time periods, categorizing content thematically while preserving geographic context. Each digest entry is associated with one or more topics defined in the system's settings [20]. Automatic summarization generates abstracts highlighting key points, with full-text translations and summaries.

The system supports filtering of materials by topic, digest date, and data source. It is implemented as a web-based platform with authenticated user access via login credentials. The system architecture is illustrated in Figure 1.

The architectural diagram annotates data flow characteristics according to Table I using the notation OxDyLz, where x denotes operational efficiency (O1-O3), y - data volume (D1-D2) and z - priority (L1-L2). These attributes determine the weights of edges in the graph-based model of the system. The diagram was designed using the app.diagrams.net service.

B. Graph-based model

The graph model in Figure 2 was generated automatically from the XML file of the architectural diagram exported from app.diagrams.net. A Python script was implemented to parse system modules and data flows, extract categorical attributes (O, D, L), and compute edge weights according to model (4). The architecture was represented as a directed graph $G = (V, E)$ using the NetworkX library. For each node, the complexity score was computed (7) using coefficients $\lambda_1 = 0.6; \lambda_2 = 0.3; \lambda_3 = 0.1$. The resulting graph was visualized with PyVis, applying a color scheme where green nodes represent low complexity ($C(v) \leq 0.2$), yellow nodes medium complexity ($0.2 < C(v) \leq 0.4$), and red nodes high complexity ($C(v) \geq 0.4$).

External entities, such as the user (usr), the Google Search Engine (gse), and RSS parsers (rss), are included in the graph to illustrate interactions with the system; however, they are not part of the internal architectural model and are therefore not

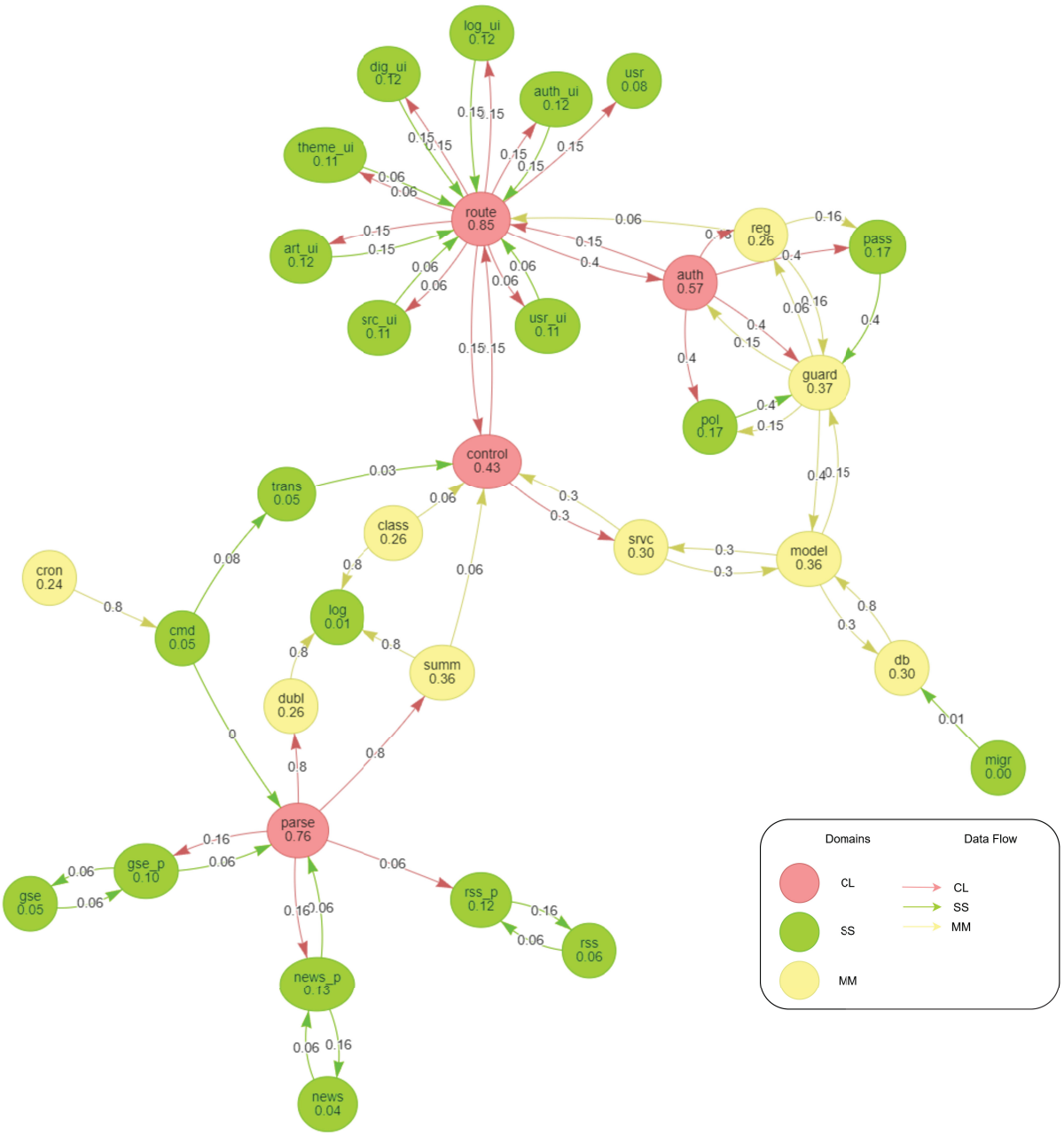

Fig. 2. Graph-based model of the system

subject to CyberImmune design principles.

C. CyberImmune system architecture

Rules for refining the system architecture are defined based on the computed $C(v)$ values. For CL domains, only non-critical data is permitted due to limitations in comprehensive testing and the high cost of such operations. To address this limitation, proxy SS domains responsible for filtering and data validation are introduced.

Considering the above details, a CyberImmune Northern Situational Monitoring System architecture was designed (Fig. 3). Initially, domains $d \in D$ were established and assigned dimensional classes based on (8). Subsequently, the TCB was defined according to CyberImmune principles. All components were considered untrusted D_{UN} . Elements of D_{UN} and D_{IN} were then organized to satisfy the system security objectives G while minimizing the TCB size.

Modules included in D_{IN} consist of web interface components, website parsers, the GoogleSearch module, RSS parsers, command modules, and the password controller. These domains, with SS class label according to $C(v)$, can be incorporated into the TCB without violating the CyberImmune

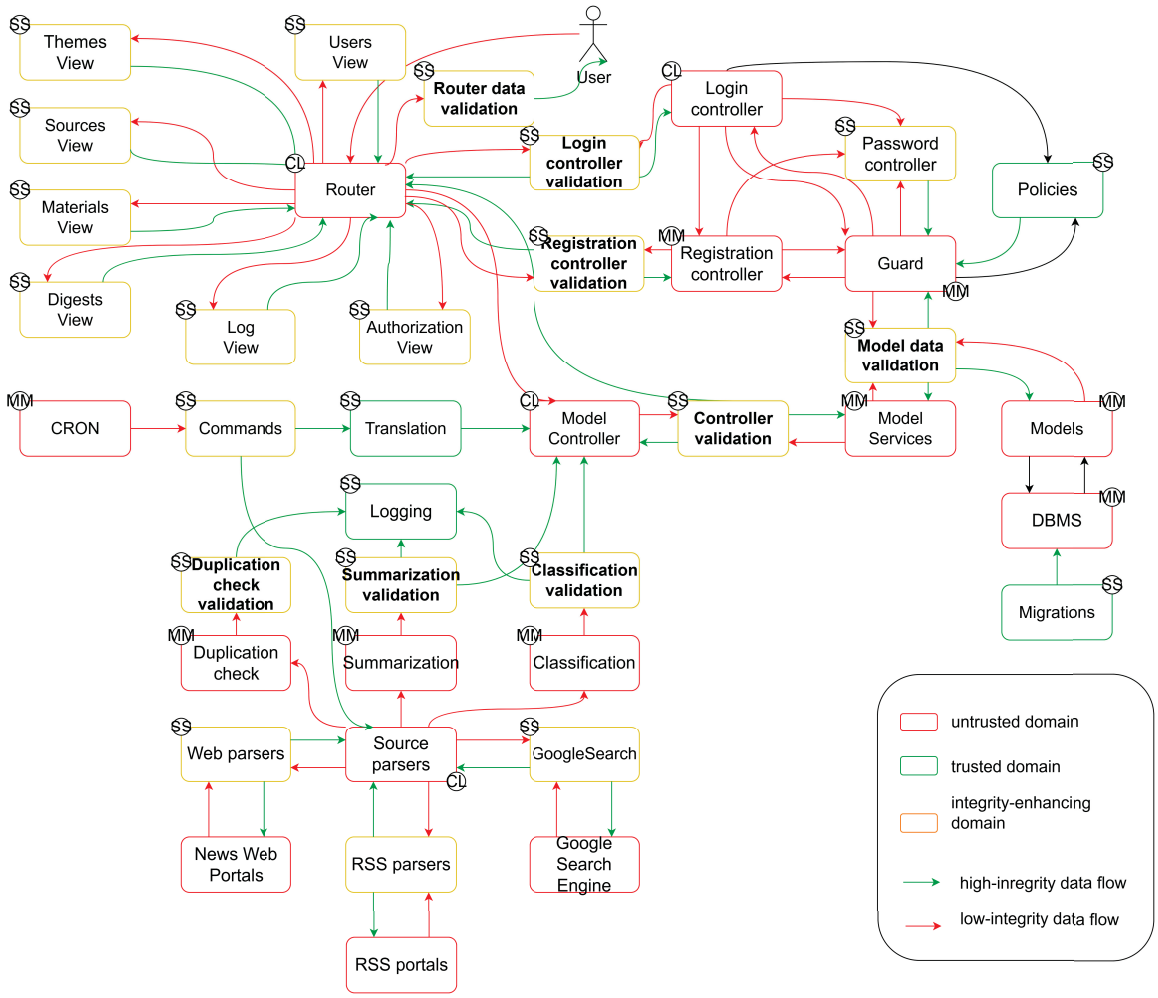

Fig. 3. CyberImmune system architecture

principles. Additionally, several modules such as translation and migration components were classified as D_{TR} due to their low complexity and small size.

Subsequently, data flows from CL-labeled domains were analyzed. In parts of the system where low-integrity data could compromise security goals, complex modules were decomposed and proxy entities were introduced to validate incoming data. For instance, improper handling of low-integrity data from duplication, summarization, or parsing modules when interfaced with the data management system could compromise original news content. To prevent this, SS-labeled proxy domains were introduced to ensure control and validation (e.g., duplicate check validation, classifier validation). The same principle was applied to access control domains, including Registration controller, Login controller, Guard, Password controller . Proxy entities are highlighted in bold in the architectural diagram.

The case study conducted on the Northern Situational Monitoring System serves as a comprehensive and developed example. This analysis, which involves mapping a realistic architecture with multiple complex data flows and module interactions effectively demonstrates the practical feasibility and interpretability of the proposed methodology for architectural decision-making. The derivation of parameters for (7) and (8) followed by subsequent architectural refinement provides prima facie evidence of the method's applicability in early design phases, consistent with the practice of using developed case studies to validate conceptual frameworks in complex engineering domains [21].

VI. Discussion

The presented case study in Section V serves to demonstrate the practical applicability of the proposed methodology. A key advantage of our approach, in comparison with existing works, lies in its ability to generate an initial distribution of vulnerable components into three complexity classes. This is achieved based solely on the high-level system structure (graph topology) and its integration characteristics, without requiring prior detailed knowledge of software implementation. Furthermore, the use of graph centrality metrics provides a comprehensive complexity assessment, synthesizing the influence of local activity, accessibility and the component's routing influence.

The proposed method inherently supports scalability and applicability to large and complex distributed architectures. Since the method operates on the high-level architectural graph, its computational complexity is primarily determined by the graph analysis algorithms, specifically the calculation of weighted centrality metrics. These standard graph algorithms exhibit well-established polynomial time complexity for dense graphs, which can be optimized for sparse graphs. Given that architectural graphs are typically sparse even for large systems, the computational burden remains tractable and does not limit the model's application to large-scale industrial or distributed systems. Furthermore, the use of categorical data for defining edge weights ensures that the model is agnostic to the system's size and remains applicable during the early architectural design stages. This characteristic makes the method suitable for initial TCB minimization and vulnerability prioritization within complex distributed environments.

As future work, we plan to conduct a full-scale experimental study for the quantitative validation of the proposed methodology. The experiment will be based on a real-world implementation of the system architecture used in the Case Study. This research will involve two main stages. We will conduct a series of simulations covering various attack and load scenarios to quantitatively assess system resilience and performance. Experimental tests will be performed on two system variants: the Original variant (control group) and an Optimized variant, where the set of defended modules is prioritized based on the complexity scores derived from the proposed method. Comparison of key metrics, such as recovery time or availability reduction will empirically confirm effectiveness of the method in enhancing system resilience.

VII. Conclusion

This study presented our graph-based method for identifying and assessing vulnerable components for architecture of OSINT systems. The key contribution lies in the integration of CyberImmune development approach, graph-based modeling, domain classification modeling, and architectural policy rules into a unified framework that enables identification of vulnerable components and minimization of the TCB. The proposed approach extends beyond traditional architectural analysis by introducing proxy-entities to ensure resilience of complex domains, thereby addressing the critical challenge of trust in OSINT systems.

The method is illustrated on a case study—OSINT system that implements regular monitoring of open web news about Northern territory. We acknowledge the necessity for quantitative validation. The method's inherent scalability, based on well-established polynomial-time graph algorithms, suggests its applicability to large-scale and distributed architectures.

Our research plan is to focus on an experimental validation utilizing a controlled microservice testbed based on the system architecture introduced in this paper. Based on experimental results, the methodology could be applied to the design and development of many other OSINT systems. Our further research may explore the integration of trustworthy AI components within OSINT systems. Leveraging AI (operate reliably and transparently in critical functions) represents a promising direction for enhancing system adaptability, automated decision-making, and overall trustworthiness, aligning with current trends in both AI and cybersecurity development.

Acknowledgement

The research is implemented with financial support of the Evgeny Kaspersky Scholarship, provided by the Gennady Komissarov Foundation for the Support of Young Scientists. The R&D results are transferred to Center for Top-level Educational Programs in Information Technology at PetrSU.

References

[1] K. Sundaramoorthy, R. Durga, and S. Nagadarshini, "Newsone—an aggregation system for news using web scraping method," in *2017 International Conference on Technical Advancements in Computers and Communications (ICTACC)*. IEEE, 2017, pp. 136–140.

[2] N. L. Venugopal, Visala *et al.*, "Advanced news archiving system with machine learning-driven web scraping and AI-powered summarization using T5, Pegasus, BERT and BART architectures," *Int. J. Exp. Res. Rev.*, vol. 46, pp. 212–221, 2024.

[3] Y. W. Hwang *et al.*, "Current status and security trend of OSINT," *Wireless Communications and Mobile Computing*, vol. 2022, no. 1, p. 1290129, 2022.

[4] H. Xu, Y. Li, W. Jin, and J. Tang, "Adversarial attacks and defenses: Frontiers, advances and practice," in *Proceedings of the 26th International Conference on Knowledge Discovery*, Virtual Event, CA, USA, 2020, pp. 3541–3542.

[5] S. P. Sobolev, "Cyber immune development approach. Microservices based illustration," *Vestnik Sankt-Peterburgskogo Universiteta. Seriya 10. Prikladnaya Matematika. Informatika. Protsessy Upravleniya*, vol. 20, no. 1, pp. 52–61, 2024.

[6] R. J. DeLong and E. Rudina, *MILS Architectural Approach Supporting Trustworthiness of the IIoT Solutions: IIC Whitepaper*. Boston: Industrial Internet Consortium, 2021.

[7] R. Spencer, S. D. Smalley, P. Loscocco, M. Hibler, D. G. Andersen, and J. Lepreau, "The Flask Security Architecture: System Support for Diverse Security Policies," in *Proceedings of the 8th USENIX Security Symposium*. Washington, DC, USA: USENIX, 1999, pp. 23–36.

[8] K.Ivashnev and D.Korzun, "Digital modeling of territory for smart e-tourism services," in *Conference of Open Innovations Association, FRUCT*, no. 35, 2024, pp. 805–811, eDN YUEAPG.

[9] M. Siddula, Y. Li *et al.*, "Privacy-enhancing preferential LBS query for mobile social network users," *Wireless Communications and Mobile Computing*, vol. 2020, pp. 1–13, 2020.

[10] C. Su, "Big data security and privacy protection," in *Proceedings of the 2019 International Conference on Virtual Reality and Intelligent Systems (ICVRIS)*, Jishou, China, 2019, pp. 87–89.

[11] Z. Avrahami, M. Zwilling *et al.*, "Leveraging OSINT for Advanced Proactive Cybersecurity: Strategies and Solutions," *IEEE Access*, vol. 13, pp. 154 229–154 250, 2025.

[12] E. Nonum, O.Avwokuruaye *et al.*, "Role of open source intelligence (osint) in cybersecurity and threat analysis," *International Journal of Latest Technology in Engineering, Management & Applied Science*, vol. 14, no. 3, pp. 189–200, 2025.

[13] J. Zhang and H. Li, "Research and implementation of a data backup and recovery system for important business areas," in *Proceedings of the 2017 9th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, vol. 2, Hangzhou, China, 2017, pp. 432–437.

[14] T. O. Browne, M. Abedin, and M. J. M. Chowdhury, "A systematic review on research utilising artificial intelligence for open-source intelligence (OSINT) applications," *International Journal of Information Security*, vol. 23, pp. 2911–2938, 2024.

[15] O. Obioha-Val, T. I. Lawal *et al.*, "Investigating the feasibility and risks of leveraging artificial intelligence and open source intelligence to manage predictive cyber threat models," *Journal of Engineering Research and Reports*, 2025, to be published.

[16] R. Ghioni, M. Taddeo, and L. Floridi, "Open source intelligence and AI: A systematic review of the GELSI literature," *AI & Society*, pp. 1–16, 2023.
[17] T. M. Kolade, O. Obioha-Val *et al.*, "AI-driven open source intelligence in cyber defense: A double-edged sword for national security," *Asian Journal of Research in Computer Science*, 2025, to be published.
[18] M. Pavlov, E. Rybin, K. Ivashnev *et al.*, "Real-time industrial automated video analytics system for welding defect detection," in *Conference of Open Innovations Association, FRUCT*, no. 36, 2024, pp. 585–592, eDN QZLIUV.
[19] A. H. A. T. Nguyen and A. M. H. Teixeira, "Centrality-based security allocation in networked control systems," in *International Conference on Critical Information Infrastructures Security*. Cham: Springer Nature Switzerland, 2024, pp. 212–230.
[20] K. Ivashnev and D. Korzun, "Text classification of news articles based on keyword sets," in *Digital Technologies in Education, Science, and Society: Proceedings of the XVII All-Russian Scientific and Practical Conference*. Petrozavodsk, Russia: Petrozavodsk State University, 2023, pp. 45–48, conference abstract, November 22–24, 2023.
[21] S. Sengupta, A. Kanjilal, and S. Bhattacharya, "Measuring complexity of component based architecture: a graph based approach," *ACM SIGSOFT Software Engineering Notes*, vol. 36, no. 1, pp. 1–10, 2011.