National University of Science and Technology
POLITEHNICA Bucharest

PhD Thesis

in Computer Science, Information Technology and System Engineering

Architecture of Distributed Reputation System for Internet Domains

Arhitectură unui sistem distribuit de reputație pentru domeniile Internet

presented by

Drd.Inf. Cristian Alexandru GHEORGHIȚĂ

supervised by

Prof.dr.ing. Florin POP

2024
Bucharest, Romania

Contents

1 | Introduction

In the contemporary digital era, the internet has become an integral part of daily life, facilitating communication, commerce, and access to information on a global scale. However, this widespread connectivity also presents significant challenges, particularly in the realm of cybersecurity. One of the pressing issues is the hijacking and misuse of domain names, which can lead to severe social and financial repercussions for both organizations and individuals. Malicious actors exploit compromised domains to distribute malware, conduct phishing attacks, and steal sensitive information such as social security numbers, bank account details, credit card information, and personal data. Such activities not only jeopardize the security and privacy of users but also tarnish the reputation of legitimate domains involved in these incidents [1].

Existing domain reputation systems, including Notos [1], Exposure [2], Kopis, [3] and publicly accessible blacklists and whitelists like Black Mirror [2] and Pi-hole BL [3], have been developed to identify and manage the credibility of domains. These systems utilize various techniques such as real-time monitoring, analysis of domain features, and evaluation of IP reputations to detect and block malicious domains. While these solutions have contributed significantly to enhancing internet security, they often face limitations related to resource constraints, scalability, and the ability to process large volumes of data efficiently. Centralized architectures struggle with the computational demands required for comprehensive real-time analysis, leading to delays in threat detection and increased vulnerability to evolving cyber attacks.

To address these challenges, this thesis proposes an innovative distributed domain reputation system that leverages the idle computational resources of voluntary nodes, specifically mobile phones that are idle while charging overnight. Inspired by projects like Folding@Home [4], which utilize distributed computing for complex scientific research, this approach aims to harness the collective processing power of a global network of devices. By distributing the workload across numerous volunteer nodes, the system seeks to overcome the limitations of centralized processing, enhancing scalability, efficiency, and the timeliness of domain reputation assessments.

[1]`https://owasp.org/`, [Accessed on 11 October 2023]
[2]`https://github.com/T145/black-mirror`, [Accessed on 11 March 2023]
[3]`https://github.com/Pyenb/Pi-hole-blocklist`, [Accessed on 25 November 2023]

1.1 Main Features of the Proposed System

The key features of the proposed distributed domain reputation system include:

- **Distributed Computing Architecture:** Utilizing a network of voluntary mobile devices to perform large-scale data processing tasks, thereby increasing computational capacity and reducing the burden on centralized servers.
- **Leveraging Mobile Cloud Computing (MCC):** Incorporating advanced web technologies and MCC [5] principles to facilitate efficient communication and processing across the distributed network.
- **Real-Time Monitoring and Analysis:** Implementing algorithms capable of analyzing domain reputations in real-time, allowing for prompt detection of malicious activities.
- **Data Anonymization and Privacy Protection:** Ensuring user privacy through data anonymization techniques and secure protocols, addressing concerns related to data security and participant trust.
- **Scalability and Resource Optimization:** Designing the system to scale horizontally with the addition of more volunteer nodes, optimizing resource utilization without significant additional costs.

1.2 Structure of Research

The research is structured to systematically explore and address the various aspects of developing the proposed system. It begins with a comprehensive review of existing domain reputation systems and cybersecurity threats, followed by the development of advanced algorithms for domain reputation assessment. The architecture of the distributed system is then designed, incorporating key principles and leveraging modern technologies. A proof-of-concept implementation is developed and evaluated using real-world data, specifically focusing on the .ro domains database. The thesis concludes with an analysis of the results, discussion of challenges and limitations, and recommendations for future research.

1.3 Methodology and Objectives

The methodology employed in this research includes a combination of literature review, algorithm development, system design, implementation, and empirical evaluation. The primary objectives oh this thesis are:

- **Analyze Existing Domain Reputation Systems and Cybersecurity Threats:**

To understand the current landscape, identify limitations, and establish the need for an improved solution.

- **Develop Advanced Algorithms for Establishing Domain Reputation:** To create effective methods for evaluating domain credibility based on various features and behaviors.
- **Design an Innovative Architecture for a Distributed Domain Reputation System:** To conceptualize a scalable and efficient system leveraging voluntary mobile devices.
- **Develop and Demonstrate a Proof-of-Concept Implementation:** To validate the proposed system through practical implementation and testing with real-world data.
- **Evaluate the System's Performance and Impact:** To assess the effectiveness of the solution compared to existing systems and its potential contribution to cybersecurity.

1.4 Thesis Outline

The thesis is organized into the following chapters:

- **Introduction:** Presents the background, significance of the research, and outlines the objectives and structure of the thesis.
- **Critical Analysis of Previous Work:** This section critically examines existing domain reputation systems, including Notos, Exposure, and Kopis, alongside commonly used open blacklists and whitelists. Each of these solutions employs distinct methodologies to assess the trustworthiness of domains. This analysis highlights the need for a more adaptive and comprehensive solution to domain reputation. The proposed architecture integrates machine learning, real-time analytics, and adaptive algorithms to overcome these limitations. By combining the strengths of existing systems and addressing their constraints, the proposed framework aims to deliver a scalable, robust, and proactive approach to domain reputation assessment, equipping stakeholders with better tools to mitigate cyberthreats effectively.
- **Domain Related Cybersecurity Threats:** This section explores key threats impacting domain reputation, including ransomware, which leverages compromised domains for malicious payload delivery [6]; Fast Flux DNS, used to obscure malicious infrastructure through rapid IP address changes [7]; and SSL certificate abuse, where attackers exploit legitimate certificates to establish trust while engaging in fraudulent activities [8]. Based on these threats, a comprehensive threat model is developed, outlining key attack vectors such as phishing, botnet command-and-control, and credential theft. This model identifies critical vulnerabilities and informs strategies for strengthening domain reputation systems to mitigate evolving cyber-risks effectively.

- **Proposed Algorithms for Establishing Domain Reputation:** This chapter provides a comprehensive exploration of domain evaluable features in Section **??**, the development of reputation algorithms in Section 4.2, and the implementation of domain reevaluation mechanisms in Section 4.3. These components collectively establish the foundational framework for the proposed architecture, enabling a systematic approach to assessing domain reputation and ensuring the integrity of reevaluated scores over time.
- **Architecture for a Distributed Reputation System for Internet Domains:** This section provides a comprehensive overview of the proposed architecture, detailing the foundational design principles in Section 5.1 and the key components that constitute the system in Section 5.2. It thoroughly examines the methodologies employed for data collection, emphasizing the selection and processing of domain features to ensure robust analysis. Additionally, in Section 5.5 it explores the integration of Mobile Cloud Computing (MCC) to enhance scalability and efficiency, leveraging distributed nodes for computational tasks. The discussion highlights how these elements synergize to create a resilient and adaptive system for accurately assessing the reputation of Internet domains.
- **Evaluation of the Proposed Architecture:** This section presents a detailed analysis of the architecture's performance, beginning with the design objectives and system requirements in Section 6.1 that guided its development. It outlines the technological stack in Section 6.2 and implementation workflow in Section 6.3, providing insights into the steps taken to achieve the desired functionality. The discussion also addresses the challenges encountered during the proof-of-concept implementation and the strategies employed to overcome them. Additionally, it highlights the key features demonstrated, showcasing the system's ability to meet its objectives and validate its effectiveness in the context of the proposed domain reputation framework.
- **Case Study on .ro Internet Domains:** Applies the system to the .ro domains database, presents the encountered particularities of the implementation regarding the access to more granular domain data, and discusses challenges and limitations.
- **Conclusions:** Summarizes the research findings, evaluates the impact of the distributed architecture, and provides recommendations for future work.

This research aims to contribute to the field of cybersecurity by introducing a novel solution to enhance domain reputation systems. By leveraging idle mobile devices in a distributed computing framework, the proposed system addresses critical limitations of existing solutions related to scalability and resource constraints. The approach promotes efficient utilization of existing resources, reduces reliance on centralized infrastructure, and fosters community engagement in cybersecurity efforts.

The potential benefits include:

- **Improved Detection of Malicious Domains:** Enhanced computational capacity allows for more comprehensive and timely analysis, leading to earlier detection of threats.

- **Scalability and Flexibility:** The system can adapt to increasing data volumes and evolving cyber threats without significant additional costs.
- **Community Empowerment and Awareness:** Involving volunteers raises awareness about cybersecurity issues and empowers individuals to contribute to collective defense mechanisms.
- **Sustainable and Cost-Effective Solution:** Maximizing the use of existing devices during idle periods promotes sustainable computing practices.

2 | Critical Analysis of Previous Work

For organizations and individuals alike, the hijacking of domain names poses significant social [1] and financial [9] threats. This malicious activity can lead to unauthorized access to sensitive information by redirecting users to fraudulent websites or distributing malware. Attackers exploit hijacked domains to disseminate malicious software that infects devices, granting them access to valuable and private data such as social security numbers, bank account details, credit card information, personal photos, and other confidential assets. The compromised domain's reputation suffers as a result, leading to a loss of trust among users and potential legal and financial repercussions for the affected parties [2].

Given these substantial risks, it is crucial to have effective mechanisms in place to identify domains with poor reputations to prevent system infections and safeguard sensitive information. Over the years, specialized systems have been developed to provide comprehensive information about a domain's reputation based on various factors. These systems utilize different methodologies, including: Real-time monitoring, Whitelists and Blacklists and IP Reputation. These include, but are not limited to:

- Publicly Accessible Whitelists and Blacklists;
- Notos;
- Exposure;
- Kopis.

In this chapter, we have conducted a comprehensive review of existing domain reputation systems and resources, namely Open Blacklists and Whitelists, Notos, Exposure, and Kopis. These systems represent significant efforts in the ongoing battle against malicious domains and cyber threats. By analyzing their methodologies, strengths, and limitations, we gain valuable insights into the current state of domain reputation analysis and identify areas where improvements are necessary.

[1] `https://itp.cdn.icann.org/en/files/security-and-stability-advisory-committee-ssac-reports/hijacking-report-12-07-2005-en.pdf`, [Accessed on 25 March 2023]

[2] `https://owasp.org/`, [Accessed on 11 October 2023]

3 | Domain Related Cybersecurity Threats

In the modern digital landscape, domain names are not merely internet addresses; they are fundamental components of the global online infrastructure that enable communication, commerce, and information exchange. The Domain Name System (DNS) serves as the internet's phonebook, translating human-readable domain names into IP addresses that computers use to identify each other on the network [10]. However, this critical role also makes domain names and the DNS infrastructure attractive targets for cybercriminals seeking to exploit vulnerabilities for malicious purposes. Domain-related cybersecurity threats have become increasingly sophisticated and pervasive, posing significant risks to individuals, organizations, and even national security.

Cyberattackers leverage domain names and DNS mechanisms to conduct a wide array of harmful activities, including phishing, malware distribution, command and control (C2) of botnets, data exfiltration, and denial-of-service attacks [11]. Among these threats, the use of Domain Generation Algorithms (DGAs) has emerged as a particularly challenging tactic. DGAs enable malware to algorithmically generate large numbers of domain names that can be used for establishing communication with C2 servers [12]. This technique allows attackers to maintain control over infected systems while evading detection and takedown efforts, as security measures struggle to block or monitor the ever-changing list of domains.

This chapter provides a comprehensive exploration of domain-related cybersecurity threats, with a focus on understanding how attackers exploit domain name vulnerabilities and DNS infrastructure. We begin by establishing a threat model [13] that identifies the assets at risk, potential adversaries, and the methods they employ to carry out attacks. By examining the attack vectors used by malicious actors, we gain insights into how domain hijacking [14], DNS spoofing [15], phishing [16], Fast Flux DNS techniques [17], SSL certificate abuse [18], and DGAs are utilized to compromise security.

The discussion on Domain Generation Algorithms (DGAs) [19] delves into how malware uses these algorithms to generate and switch between domains dynamically, complicating efforts to block malicious domains and disrupting traditional security defenses. We explore how DGAs contribute to the resilience of botnets and malware campaigns by enabling robust and flexible C2 infrastructures [20]. Understanding DGAs is crucial for developing effective detection and mitigation strategies, as they represent a significant evolution in the tactics used by cybercriminals to avoid detection and sustain their operations.

By analyzing these threats, we aim to highlight the weaknesses that attackers exploit and underscore the importance of advanced domain reputation systems capable of real-time detection of malicious domains. The knowledge gained from this exploration serves as a foundation for subsequent chapters, where we propose innovative algorithms and a distributed architecture designed to enhance the detection and prevention of these threats.

This chapter emphasizes the critical need for a comprehensive understanding of domain-related cybersecurity threats to develop effective countermeasures. Through this examination, we aim to contribute to the broader effort of securing the internet's domain infrastructure and protecting users from the ever-evolving tactics of cyberadversaries.

3.1 Threat Model

A threat model is a structured representation of the potential security threats to a system, which helps in understanding the attacker's objectives, the vulnerabilities they might exploit, and the impact of their actions. In the context of domain name related cybersecurity, the threat model focuses on how malicious actors can exploit the Domain Name System (DNS) and domain names to conduct harmful activities. This includes manipulating domain registrations, DNS records, and leveraging domain-related vulnerabilities to compromise systems, steal data, or disrupt services. This section outlines the threat model by identifying the assets at risk, potential adversaries, attack vectors, and the possible impacts of these threats.

3.1.1 Assets

In the context of domain name-related cybersecurity, several critical assets are vulnerable to exploitation by malicious actors. These are the following:

- Domain Names and DNS Infrastructure;
- User Trust and Reputation;
- Sensitive Information;
- Network Resources;
- Business Operations.

3.1.2 Potential Adversaries

The threat landscape for domain name-related cybersecurity involves a diverse array of adversaries, each with distinct motivations and methods. Cybercriminals are among the most prevalent adversaries. These individuals or organized groups are primarily driven by financial gain. They engage in activities such as stealing sensitive data, conducting fraud, distributing malware, and orchestrating ransomware attacks [21]. Cybercriminals often operate trans-nationally, leveraging sophisticated techniques to exploit vulnerabilities in domain names and DNS infrastructure, and they may sell stolen data or access on the darkweb [22].

The principal potential adversaries are the following:

- Hacktivists;
- State-Sponsored Entities;
- Insiders;
- Script Kiddies;
- Competitors.

3.2 Attack Vectors

This section reviews significant attack vectors threatening domain security, focusing on strategies that exploit domain features rather than DNS-specific vulnerabilities. These methods underscore the need for sophisticated reputation algorithms to mitigate domain-related risks. The discuessed methods are the following:

- Domain Hijacking;
- Phishing and Typosquating;
- Fast Flux DNS;
- Malware Distribution via Compromised Domains;
- Domain Generation Algorithms;
- Command and Control (C2) via DNS.

3.3 Conclusions

In this section, we have outlined the primary methods by which a domain may become compromised. By synthesizing and analyzing these techniques, we can construct a more comprehensive understanding of the characteristics and behaviors exhibited by compromised domains. This enhanced understanding serves as a foundation for refining and optimizing the algorithm to improve the accuracy and reliability of domain compromise detection.

4 | Proposed Algorithms for Establishing Domain Reputation

In this chapter we will create a model for a domain reputation algorithm by identifying domain names evaluable features by using classifications as: syntactical features, network features and zone configurations and we will establish a scoring methodology in order to achieve a high true positive rate and also to obtain an as small as possible false positives rate. Finally we will tackle the novelty part of our algorithm represented by the ability to reevaluate an already processed domain to make sure that the reputation is upheld or in some cases, the reputation has been cleaned by repurposing the domain name by a legitimate actor.

4.1 Data Collection and Verification

In building a robust domain reputation system, the foundation lies in the comprehensive collection and meticulous verification of data pertaining to domain names. This data serves as the critical input for our reputation algorithm, influencing its ability to accurately assess the trustworthiness of domains. Given the vast number of domains and the dynamic nature of the internet, collecting relevant, high-quality data presents both a significant challenge and a vital necessity.

For a more efficient gathering of the data, in a similar fashion to Antonakis et al [1], we have split the domain features in three categories: Domain features, DNS features and Network features.

- **Domain Features**:
  This category includes:
  - Domain age,
  - Registration length,
  - Registrant details,
  - Domain expiration,
  - Lexical analysis,
  - Typosquatting,
  - Domain name length,

  - Blacklist inclusion,
  - Previous ownership,
  - Assessments from third-party reputation services,
  - Content relevance and quality,
  - Email spam reports,
  - Algorithmically generated domains,
  - Domain names containing high entropy.

- **DNS Features**:
  This category includes DNS configuration-related issues, such as:
  - Fast Flux techniques,
  - Unusual subdomain usage,
  - DNSSEC adoption,
  - Email authentication records,
  - Email open relays.

- **Network Features**:
  This category includes network configuration-related issues and associations with malicious IPs or networks, such as:
  - IP address,
  - SSL configuration,
  - Domain infrastructure security (obsolete technologies, vulnerable security protocols),
  - Redirections to known malicious domains.

The domain features from the first category are obtained using a combination of data sources and analytical techniques. This section outlines the methods employed to collect and analyze these features, ensuring that the data gathered is accurate, up-to-date, and relevant for the reputation algorithm.

The main data sources are the following:

- **WHOIS:**
  A query response protocol widely used for obtaining information about Internet resources, including domain names, IP address blocks, and autonomous system numbers (ASNs);

- **DGA Detection Tool:**
  A tool based on the DGA identifying algorithm provided by the Safing Portmaster project[1];

[1]`https://github.com/safing/portmaster/tree/v0.6.4/detection/dga`, [Accessed on 13 June 2023]

- **Typosquatting tool:**
  A developed tool that combines the ability of URLInsane[2] to identify typosquatting domain names, based on a legitimate domain, but implemented to work recursively to identify if a given domain name is a typosquatted domain;
- **Inspecting DNS Features:**
  By inspecting the DNS features, we can extract information about the configuration of the domain name and it's potential vulnerabilities.

As a summarization of the domain features that will be included in our algorithm, we have created Table 1.

4.2 Reputation Algorithm

We chose to implement the reputation score algorithm using machine learning, and more specifically, with the help of the XGBoost [23] algorithm, more specifically the regression variant. This allows the system to adapt over time with more domains being validated. The results of the newly validated domains will be feedback to a trainer algorithm that will generate a new and more accurate model.

4.2.1 Machine Learning Algorithms

XGBoost, or eXtreme Gradient Boosting, is an advanced variant of Gradient Boosting Machines originally introduced by J.H. Friedman [24]. Gradient Boosting builds models sequentially, with each model correcting errors from its predecessors by minimizing a loss function through models aligned with the negative gradient of the loss. XGBoost enhances this framework with optimizations tailored for high efficiency, scalability, and performance, particularly on large-scale, high-dimensional data.

A distinguishing feature of XGBoost is its regularized objective function, which combines a loss function with a regularization term to penalize overly complex models, such as large decision trees. This explicit regularization mitigates overfitting, balancing accuracy and model simplicity. Additionally, XGBoost introduces a histogram-based algorithm for feature splitting, which discretizes continuous features into bins. This approach reduces computational costs while maintaining high accuracy, enabling the rapid construction of decision trees.

Another critical advantage is XGBoost's native handling of missing values. During training, it automatically learns optimal splits for missing data without requiring preprocessing or imputation. This capability is particularly valuable in our domain feature collection process, where data gaps could compromise model integrity. By addressing missing values

[2]`https://github.com/ziazon/urlinsane`, [Accessed on 13 June 2023]

Feature	**Data Type**	**Description**
Domain Age	Categorical	Age of the domain in years (<1 year, 1 - 5, 5 - 10, 10 - 20, >20)
Domain Expiration (years)	Numeric	Days until domain expiration
Domain Length	Numeric	Number of characters in the domain name
DGA Suspect	Boolean	1 if suspected to use Domain Generation Algorithm, else 0
Typosquatting Suspect	Binary	1 if suspected typosquatting, else 0
DNS TTL (Time To Live)	Categorical	One of (<300, 300 - 1800, 1800 - 86400, 86400 - 604800, >604800)
DNSSEC	Boolean	1 if DNSSEC is enabled, else 0
Fast Flux Suspect	Boolean	1 if suspected of fast flux DNS technique, else 0
SSL	Boolean	1 if SSL is configured, else 0
SSL Certificate Issuer	Categorical	Categorical encoding of the SSL issuer
SSL Certificate Valid From	Numeric	Days since SSL certificate was issued
SSL Certificate Validity	Numeric	Duration in days that the SSL certificate is valid
Registrar	Categorical	Categorical encoding of the domain registrar
IP Address	Numeric	Numerical representation of the IP address
AS Number	Numeric	Autonomous System Number
Is Blacklisted	Boolean	1 if the domain is blacklisted, else 0
Is IP Blacklisted	Boolean	1 if associated IP is blacklisted, else 0
TLS Version	Categorical	Categorical encoding of the TLS version
Strict-Transport-Security	Boolean	1 if HSTS is enabled, else 0
Content-Security-Policy	Boolean	1 if CSP is implemented, else 0
Has Email Server	Boolean	1 if the domain has an email server configured, else 0
Is Email Server Blacklisted	Boolean	1 if the email server is blacklisted, else 0
Email Has DMARC	Boolean	1 if DMARC is configured, else 0
Email Has SPF	Boolean	1 if SPF is configured, else 0
Email Has DKIM	Boolean	1 if DKIM is configured, else 0
Email Server Is Open Relay	Boolean	1 if the email server is an open relay, else 0

Table 1: Features Used for Domain Reputation Assessment.

directly during tree construction, XGBoost ensures robustness and reliability, minimizing the adverse effects of incomplete datasets and enhancing the predictive model's overall performance.

4.2.2 Training Data Structure

As shown in Table 1, the domain features used in this process consist of a combination of numeric, boolean, and categorical values. Categorical values are inherently more complex to process, as they represent discrete, non-numeric information, such as Registrar Name or domain classifications. These values require encoding techniques to convert them into formats that can be interpreted by the algorithm. A common approach is One-Hot encoding [25], where each category is represented as a separate binary feature. Alternatively, label encoding can be used, assigning a unique numerical value to each category, though this may introduce unintended ordinal relationships.

Registrar Name	**Encoded Representation (One-Hot)**
GoDaddy	[1, 0, 0, 0]
Namecheap	[0, 1, 0, 0]
Google Domains	[0, 0, 1, 0]
Romarg	[0, 0, 0, 1]

Table 2: One-Hot Encoded Representation of Registrars.

According to XGBoost documentation, One-Hot is the recommended encoding because it ensures the absence of implicit ordinal relationships, which is critical when no inherent order exists between categories. Proper handling of such features ensures that the algorithm can process diverse data types effectively, thereby enhancing its ability to produce accurate and unbiased reputation scores.

4.2.3 Model training

To train the domain reputation model, a dataset in CSV format was prepared, comprising both benign and malicious domains to ensure diversity and robust generalization capabilities. For benign domains, a subset of Alexa's Top 1 Million Domains dataset [3] was used, selecting the top 100,000 domains for efficient processing. For malicious domains, the Black Mirror Blacklist [4], a dynamic and frequently updated repository, was utilized. The custom-built domain features extractor application, implemented in Go, facilitated the extraction of features for 100,000 benign and 100,000 potentially malicious domains. This balanced dataset provided a strong foundation for training, minimizing overfitting risks.

The dataset was structured to include an additional column indicating domain origin—either benign or from the blacklist. It was then divided into training and testing subsets, enabling performance evaluation and generalization testing. Cross-validation,

[3]`https://www.kaggle.com/datasets/cheedcheed/top1m`, [Accessed on 20 June 2023]

[4]`https://github.com/T145/black-mirror`, [Accessed on 11 March 2023]

specifically the k-fold methodology [26], was employed to enhance evaluation rigor. By partitioning the data into k subsets and iteratively designating one subset for testing while the rest were used for training, a comprehensive performance assessment was achieved, mitigating overfitting risks. Results from these iterations were aggregated for robust model evaluation.

The finalized model was serialized into a portable file format for distribution across mobile worker nodes in the distributed system. This ensured uniformity and compatibility across various environments, enabling worker nodes to perform effective predictions or classifications. By decentralizing the trained model, the system leveraged the distributed resources of the network for scalable and efficient domain analysis, ensuring high redundancy and resilience. This approach underscored the system's adaptability and scalability, supporting diverse hardware configurations and optimizing computational capacity within the distributed computing network.

4.2.4 Model evaluation

To evaluate the performance of the initial model, a subset of domains, comprising both legitimate and malicious examples, was excluded from the training process and reserved for testing purposes. These domains, referred to as L1 and L2 for legitimate domains and M1 and M2 for malicious domains, were carefully selected to ensure a balanced representation of the data's diversity.

The evaluation was conducted using the worker application specifically developed for deployment on the distributed computing nodes. This application processed the reserved sample domains, allowing us to assess the model's ability to classify previously unseen data.

The analysis of extracted domain features highlights key indicators of suspicious domains, with domain age being a significant factor, as younger domains are frequently linked to malicious activities due to their transient nature. Time-to-Live (TTL) settings also serve as a reliable metric, with unusually low values often signaling malicious behavior. Additionally, blacklisting status—whether for IP addresses, mail servers, or the domain itself—provides substantial evidence of potential threats. While SSL certificates were once a trust indicator, their reliability has waned due to the ease of obtaining short-lived certificates, often used by malicious actors. These findings underscore the importance of a multifaceted approach, integrating multiple domain features to enhance detection accuracy and ensure a comprehensive evaluation of domain trustworthiness.

The evaluation results, summarized in Table 4, demonstrate the prediction algorithm's ability to classify domains based on reputation scores and associated accuracy. High reputation scores (0.93 and 0.89) for legitimate domains, with accuracy values of 0.87 and 0.91, indicate strong model performance, albeit with some variability suggesting potential

Feature	L1	L2	M1	M2
Domain Age	>20	>20	<1 year	<1 year
Domain Expiration	917	1282	194	256
Domain Length	5	4	7	29
DGA Suspect	0	0	0	1
Typosquatting Suspect	0	0	0	0
DNS TTL	1800 - 86400	1800 - 86400	<= 300	<= 300
DNSSEC	1	0	0	0
Fast Flux Suspect	0	0	0	0
SSL	1	1	1	1
SSL Issuer	ISSUER1	ISSUER2	ISSUER3	ISSUER4
SSL Certificate Valid From	304	116	29	53
SSL Certificate Valid To	92	255	61	392
Registrar	REG1	REG2	REG3	REG3
IP Address	3231846509	783192862	3259220266	2890123802
AS Number	AS3233	AS47388	AS214231	AS13335
Is Blacklisted	0	0	1	1
Is IP Blacklisted	0	0	1	0
TLS Version	TLSv1.2	TLSv1.3	TLSv1.3	TLSv1.3
Strict-Transport-Security	0	0	0	0
Content-Security-Policy	0	0	0	0
Has Email Server	1	1	0	0
Is Email Server Blacklisted	0	0	0	0
Email has DMARC	0	1	0	0
Email has SPF	1	1	0	0
Email has DKIM	0	1	0	0
Email Server is Open Relay	0	0	0	0

Table 3: Feature Comparison Across Domains.

Domain	Reputation Score	Accuracy
L1	0.93	0.87
L2	0.89	0.91
M1	0.21	0.82
M2	0.15	0.84

Table 4: Domain Reputation Score and Accuracy.

improvements in feature weighting or dataset diversity. Conversely, low reputation scores (0.21 and 0.15) for malicious domains, accompanied by accuracy values of 0.82 and 0.84, highlight the model's effectiveness in identifying threats while exposing challenges in distinguishing edge cases where malicious and benign domain features overlap.

Key features such as domain age, DNS TTL, blacklisting status, and email configurations strongly influence the model's predictions, underscoring their relevance in domain reputation assessment. However, features like SSL attributes and security policies exhibit limited variability, suggesting opportunities for refinement to enhance discriminative power. The balanced dataset of 100,000 benign and 100,000 malicious domains contributes to generalizability, though the inclusion of additional domain types, real-time DNS data, and behavioral analytics could further optimize performance. Overall, the model demonstrates robust classification capabilities while revealing avenues for future enhancements to address remaining limitations and improve reliability.

4.3 Domain Reevaluation and its Effect on Current Score

The system has been designed to incorporate a domain reputation score decay mechanism for domains previously evaluated. This feature addresses two critical challenges: maintaining the validity of the reputation score over time and prioritizing domains for reevaluation.

Once a domain has been evaluated, it is reasonable to assume that its status will remain relatively stable in the immediate aftermath, thereby maintaining its reputation score. This assumption is particularly valid for domains with a long history and minimal updates, as their stability weighs more heavily in comparison to younger domains with frequent changes. These factors are considered to optimize the scheduling process and improve the accuracy of reputation metrics.

In light of these considerations, a formula has been developed to introduce a controlled "decay" in the reputation score of a domain after its evaluation. This decay mechanism ensures that as time passes, domains with older evaluations are naturally prioritized for reevaluation, maintaining the reliability and timeliness of the system's metrics.

$$\text{new\_score} = \text{old\_score} \cdot \left(1 - \frac{\lambda}{\text{domain\_age\_days} + \text{revalidation\_days}}\right)^{\text{days}}$$

This version of the formula balances the decay dynamics, ensuring fairness across domains of varying ages and validation histories while maintaining the integrity and reliability of the reputation scoring system.

Having implemented this reputation score decaying formula, ensures that younger, more vulnerable domains are reevaluated more often than older and more established domains. The rate at which new domains are being registered [Monitoring the Initial DNS Behavior of Malicious Domains. Shuang Hao] it makes it hard to evaluate and monitor all of them. We think that our approach improves on this issue by offering more validation bandwidth to these domains in favor of already reputable domains.

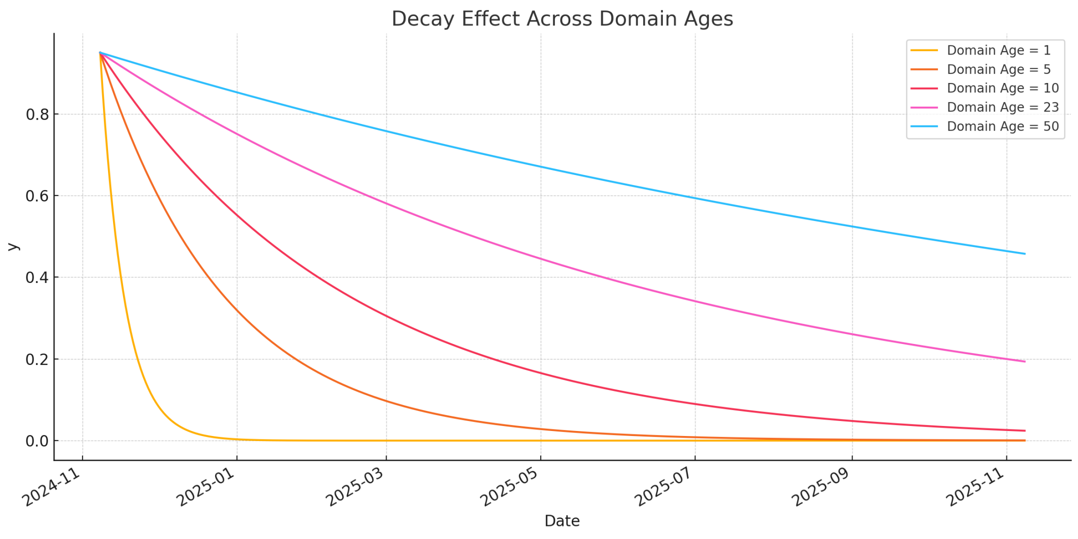

Figure 1: Decay Effects with Final Formula.

4.4 Conclusions

This chapter has introduced a comprehensive model for assessing domain reputation by identifying evaluable domain features, establishing scoring methodologies, and integrating a novel reevaluation mechanism. By categorizing features into syntactical, network, and zone configuration groups, the model captures the complex nature of domain behavior and security. Validated through rigorous analysis of feature extraction, data encoding, and machine learning evaluation, the approach demonstrated high accuracy and reliability in distinguishing legitimate from malicious domains. Addressing challenges like rapid domain registration, dynamic DNS configurations, and inconsistent security practices, the model provides a scalable, adaptive framework with real-time revalidation to ensure up-to-date and fair assessments. This foundational chapter paves the way for practical implementation and refinement, setting the stage for the system's architecture, evaluation, and applications in enhancing cybersecurity and trust.

5 | Architecture for a Distributed Reputation System for Internet Domains

In the ever-changing landscape of cybersecurity, accurately and efficiently assessing domain reputation is critical. Centralized systems often face limitations in scalability, latency, and resource availability when analyzing the vast and dynamic nature of the internet. This chapter introduces an innovative distributed domain reputation architecture leveraging distributed computing and mobile cloud computing (MCC) [5] to enhance scalability, reduce latency, and improve accuracy. By decentralizing processing tasks across a network of volunteer nodes, including mobile devices, the system addresses core challenges in domain reputation assessment.

The proposed architecture functions by partitioning large datasets and distributing computational tasks, such as feature extraction or preliminary analysis, to volunteer nodes. Results are aggregated by a central server, which refines reputation scores and redistributes tasks. This design emphasizes scalability, fault tolerance, and efficient resource utilization, drawing inspiration from successful distributed computing frameworks like Folding@home. Key benefits include:

- **Scalability:** The system can handle increasing data volumes by enlisting more volunteer nodes, avoiding bottlenecks of centralized processing;
- **Latency Reduction:** Distributing tasks across multiple nodes or processing data closer to its source decreases the time required for reputation scoring;
- **Resource Optimization:** Utilizing underutilized devices maximizes resource use without significant additional investment;
- **Community Engagement:** Involving volunteers fosters a sense of collective effort, promoting cybersecurity as a shared responsibility akin to Folding@home.

By addressing the inherent scalability challenges of centralized systems, this distributed approach enables the efficient processing of vast datasets, fostering a collaborative and scalable solution for real-world cybersecurity needs.

5.1 Key Principles and Assumptions

In designing the architecture for a distributed domain reputation system, several key principles and foundational assumptions guide the development and implementation. These principles ensure that the system is robust, scalable, efficient, and secure, while the assumptions establish the operational context within which the system functions. This section delves into these guiding principles and the assumptions that underpin the system's architecture.

The following are the key principles that guided us in desiging the proposed architecture:

- Scalability;
- Reliability;
- Efficiency;
- Security.

The design of a distributed domain reputation system is guided by several key assumptions, which establish the foundational context for its implementation and operation. These assumptions address the technical, operational, and regulatory considerations that are critical to the system's functionality and success. The following are the main assumptions taken while designing the proposed architecture:

- Node Participation;
- Network Connectivity;
- Resource Availability;
- Data Privacy Compliance.

By explicitly defining and addressing these assumptions, the system design ensures that potential risks are mitigated, operational parameters are realistic, and the architecture aligns with technical and regulatory standards. These foundational assumptions not only guide the development process but also serve as a benchmark for evaluating the system's feasibility and scalability in real-world deployments.

5.2 Proposed Architecture Components and their Interactions

The proposed architecture is composed from two main components:

- **Nexus:** The main control and command server responsible for:
  - Scheduling domains for validation.
  - Validating the responses from the mobile workers.

  - Versioning and storing the results in the database.
  - Providing statistics and reports for analyzed domains.
  - Handling queries for specific domains.
- **BOINC:** [refference here] A volunteer work processing scheduler tasked with:
  - Managing the connected volunteer devices.
  - Allocating the work they are supposed to carry out.
  - Collecting and managing the results obtained from the workers.

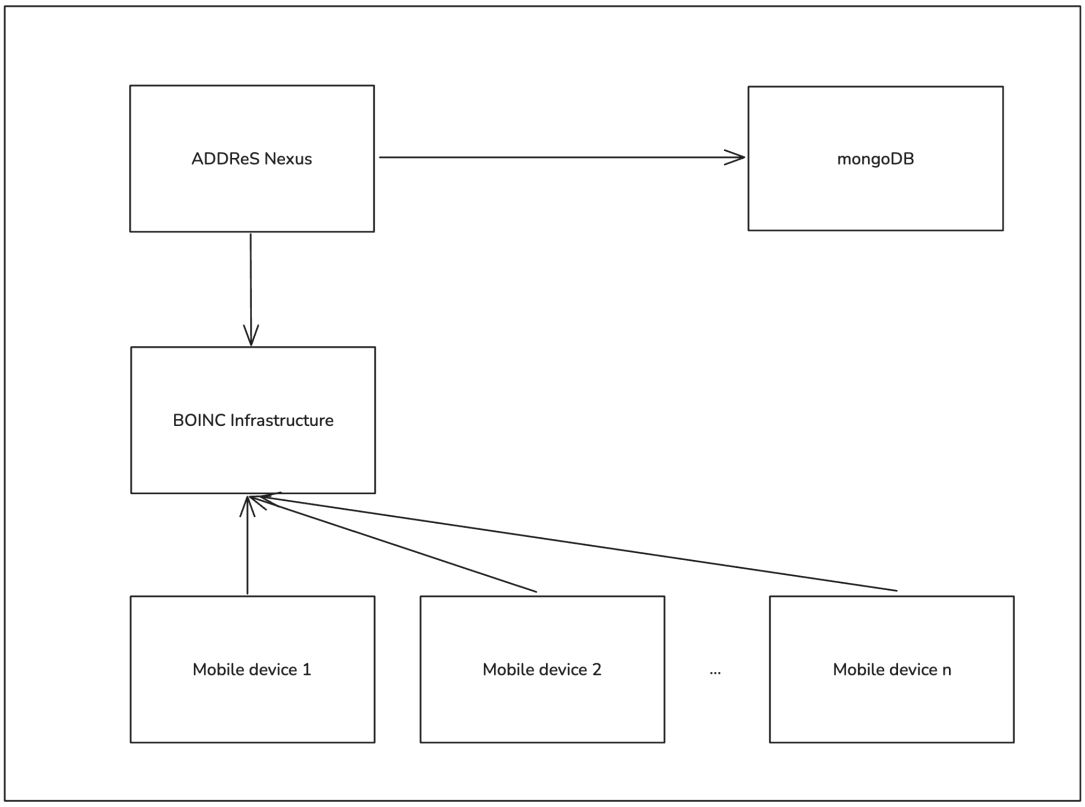

Figure 2: ADDReS architecture components.

5.3 The Command & Control System - Nexus

Although the majority of the moving parts of the system are decentralized, in order to preserve the privacy and security of the system, we preferred to keep the decisional factor as a closed service. This way we mitigate possible attempts of tampering the results of the domain evaluations by employing random distribution of work and consensus for the discovered results.

5.3.1 Web Service

The web service acts as a gateway to the system by providing tools that enable the management of the system. It employs an User Management system that supports different roles based on the level of access for the operators. It allows operators to schedule specific domains for validation and also to interact with the results obtained from domain evaluations.

5.3.2 Domain Evaluation Scheduler

The domain scheduler is in charge of scheduling new and old domains for evaluation. We say new and old because the system is designed to reevaluate domains that were prior evaluated based on the level of score decay. The more decay a score presents, the more likely it is that the domain will be reevaluated. This method allows the system to revisit domains that are more prone to become suspicious and to achieve near real-time monitoring of the domains.

The scheduler starts by checking the available resources of the system and allocates them equally between the new domains and old domains. If there are now old domains to revalidate, the resources are reallocated towards the new domains queue. At first the balance will tip towards new domains because of lesser to none revalidations, but as the time passes and more domains get validated, the balance will get restored and new and old domains will get bandwidth allocated.

```
for domain in existing_old_domains:
    domain_obj = new Domain(
        domain_name=domain.name,
        decay_value=domain.decay_value,  # decay_value > 0
        last_evaluated=domain.last_evaluated
    )
    old_pq.insert(domain_obj, priority=domain_obj.decay_value)

while True:
    R = get_total_resources()
    R_new = R * 0.5
    R_old = R - R_new  # Handle odd R

    for i from 1 to R_new:
        if new_queue is not empty:
            domain = new_queue.dequeue()
            schedule_evaluation(domain)
        else:
            R_old += (R_new - i + 1)
            break

    for i from 1 to R_old:
        if old_pq is not empty:
            domain = old_pq.extract_max()
            schedule_evaluation(domain)
        else:
            R_new += (R_old - i + 1)
            break

    new_domains = get_new_domains()
    for domain in new_domains:
        domain_obj = new Domain(
            domain_name=domain.name,
            decay_value=0,
            last_evaluated=null
        )
        new_queue.enqueue(domain_obj)
    wait_until_next_cycle()
```

Listing 5.1: Scheduling Algorithm for Domain Evaluation.

5.3.3 Domain Evaluation Distributor

The domain evaluation distributor is in charge of distributing the work to the volunteer worker nodes in such a way that it ensures the security of the results. Being an open service and allowing anyone to participate in the process of evaluating a domain's reputation, this allows for bad actors to try to influence the results for domains they wish to whitelist. To mitigate this issue, we have devised a formula with which the system determines the number of redundant nodes to choose for a particular task. Only after the system reaches consensus, the results are being validated and stored.

Since the number of nodes participating in the system is variable, we have designed a formula that will output the recommended number of nodes for the work to be distributed to.

We have devised the following formula:

$$\sum_{k=\lceil \frac{n}{2} \rceil}^{n} C_k^n (1-p)^k p^{n-k} \geq c$$

allows for the calculation of the minimum number of redundant nodes n needed to achieve a desired confidence level C that the majority of nodes evaluating a domain are honest, given a node compromise probability p. Thus, for a probability of node compromise of 0.1 (10%) and a desired confidence level of 0.95 (95%) we will need a number of at least 3 nodes.

5.3.4 Evaluation Results Consensus

When the finished results of the working nodes are being retrieved, it is important to establish consensus among the results to:

- Account for the usual variances in score due to:
  - Data timing differences: Nodes may have slightly different views of the data due to updates occurring between evaluations.
  - Local data variations: Nodes may have access to different subsets of data, such as cache states.
  - Network latency: Delays in data retrieval can affect the freshness of the data used in evaluation.
- Account for the possibility of malformed results from malicious devices.

To counter these differences, we have established a consensus mechanism that is based on the application of Interquartile Range (IQR)[27] over the list of scores generated by the nodes. We have chosen this method for its ability to effectively reduce the influence of extreme values (outliers), leaving only the legitimate scores to be registered.

5.3.5 Domain Score Decay Applicator

An important component of the reevaluation process is represented by the domain score decay applicator (DSDA) which is scheduled to run each day to properly adjust the decay factor for evaluated domains. The decayed scores produced by the Domain Score Decay Applicator are utilized by the Domain Evaluation Scheduler to prioritize domains for re-evaluation. Domains with lower scores (indicating higher decay) are given higher priority, ensuring that resources are allocated to domains with the most uncertainty.

```
DECAY_CONSTANT = 0.5
MIN_SCORE = 0.0
MAX_SCORE = 1.0

for domain in domain_repository.get_all_domains():
    # Calculate effective domain age
    effective_domain_age = max(domain.domain_age, 1)

    # Calculate decay denominator and decay factor
    decay_denominator = effective_domain_age + domain.
        revalidation_count
    decay_factor = 1 - (DECAY_CONSTANT / decay_denominator)

    # Ensure decay factor is between 0 and 1
    decay_factor = max(min(decay_factor, 1), 0)

    # Update domain score with decay factor
    decayed_score = domain.score * (decay_factor ** domain.
        days_since_validation)
    decayed_score = max(min(decayed_score, MAX_SCORE), MIN_SCORE)

    # Update the domain score and save changes
    domain.score = decayed_score
    domain_repository.update(domain)
```

Listing 5.2: Domain Score Decay Algorithm.

5.4 Storage System

For the storage aspect of the system we needed to find a solution that allows for fast insertion of data and also permits a flexible structure of the ingested data. Among multiple available solutions, we choose MongoDB[1] for its performances and also for its ability to scale.

5.5 Distributed Computing Network

We propose a novel domain reputation evaluation system grounded in a distributed computing architecture. Specifically, our approach leverages the computational resources of mobile devices within a distributed network. This architecture mitigates the dependency on centralized high-performance computing resources by distributing the workload across a broad network of volunteer devices. By harnessing the collective computational capacity of mobile devices, the proposed system aims to provide a scalable, cost-effective, and efficient solution for real-time domain reputation evaluation.

The management of such a distributed architecture also demands scalable infrastructure to handle device registration, task assignment, and result collection, as well as secure methods to protect data privacy and prevent unauthorized access. Despite these challenges, the potential benefits of a distributed system, including cost-efficiency, scalability, and resilience, make it a promising approach for large-scale computational tasks such as domain reputation evaluation.

The main components of the system are:

- Core Server;
- Worker Nodes;
- Task Distribution;
- Consensus Mechanisms;

[1]`https://www.mongodb.com/`, [Accessed on 02 October 2023]

- Work Assimilation.

5.6 Conclusions

The proposed distributed computing network effectively addresses the limitations of centralized systems in domain reputation evaluation by leveraging decentralized worker nodes to enhance scalability, efficiency, and cost-effectiveness while reducing dependency on high-performance infrastructure.

6 | Evaluation of Proposed Architecture

This chapter outlines the objectives, requirements, technological choices, implementation workflow, challenges faced, and the features successfully demonstrated through a Proof of Concept (PoC). It provides a concrete foundation for validating the system's design and paves the way for further development and deployment.

6.1 Design Objectives

By implementing the proof of concept system we aimed to validate several critical aspects of the proposed distributed domain reputation system. The specific design objectives were:

- Deploy the infrastructure to assess system requirements and document the process.
- Enable domain feature extraction and machine learning prediction on Android devices without modifying the underlying platform.
- Compile and deploy the BOINC wrapper and Worker Application on the Android devices.
- Implement a robust validation mechanism for task replication across multiple nodes to ensure the security of the results.
- Integrate validated results into the ADDReS Nexus management system.
- Demonstrate end-to-end functionality of the distributed system.
- Address technical challenges and document solutions.
- Evaluate system performance and resource utilization.

6.2 Technological Stack

In this section, we provide an in-depth examination of the selected technological stack employed for the proof-of-concept implementation. Furthermore, we elucidate the rationale underpinning each decision, highlighting how these choices align with the system's design objectives and operational requirements.

- **ADDReS Nexus:**
  - **Web Component**: Provides a visual interface for interacting with the system. It incorporates multiple access levels, enabling users to engage with the stored domain data according to their permissions.
  - **Scripts**: Includes the following key scripts used by the system:
    - *Domain Scheduler*: Manages the scheduling of domain evaluations.
    - *Score Decay Applicator*: Implements the decay mechanism for domain reputation scores.
    - *Work Validator*: Ensures the integrity and accuracy of results from distributed nodes.
    - *Work Assimilator*: Integrates validated results into the system's database.
  - **Storage**: Responsible for storing essential system data, including:
    - User credentials and access tokens.
    - Extracted domain features.
    - Evaluation scores for domain reputation.
- **Distributed Computing Infrastructure:**
  - **BOINC Client**: The software installed on volunteer devices to receive and execute work units.
  - **BOINC Server**: The central server that manages the distribution of work units and collection of results.
  - **BOINC Project Applications**: Specific applications that define the tasks performed by the BOINC clients.
  - **Work Units and Result Files**: Units of work sent to clients and the corresponding result files returned after execution.
  - **Scheduler**: A component that assigns work units to available clients and ensures efficient task distribution.
  - **BOINC Manager (User Interface)**: A user-facing application that allows participants to monitor and manage their contributions to the distributed system.
  - **Web-Based Administration and Community Tools**: Online tools for managing the BOINC project, engaging with the community, and providing project statistics.

6.2.1 Proof of Concept Foundation

To develop a reproducible and distributable proof of concept, Docker[1] has been selected as a lightweight containerization platform. This choice allows us to ensure the system

[1] `https://docs.docker.com`, [Accessed on 29 March 2024]

can be reproduced and distributed reliably. Docker works by bundling everything an application needs—like its code, runtime, libraries, and settings—into containers that behave consistently across any environment. This makes Docker an ideal foundation for building proof-of-concept (PoC) systems composed of multiple services.

6.3 Implementation Workflow

The implementation of the proof of concept (PoC) involved a series of methodical steps to overcome technical challenges and achieve the design objectives outlined earlier. This section provides a detailed account of the development process, highlighting key activities and milestones that led to the successful execution of the distributed domain reputation system using BOINC and Android devices.

6.3.1 Developing the ADDReS Nexus Web Service

To manage the system's domain scheduling and validation results, MongoDB was integrated using PyMongo[2], while user administration continued to rely on Django's default SQLite database. This hybrid approach allowed us to leverage MongoDB's flexibility and performance for handling domain-related data while maintaining the simplicity and robustness of Django's built-in user authentication and administration features.

This configuration allowed the system to achieve the best of both worlds. MongoDB provided the scalability and flexibility needed for managing domain scheduling and validation at scale, while Django's Admin interface and SQLite ensured a seamless and secure experience for user management. This approach also positioned the system for future scalability, as the MongoDB-backed components could evolve independently, supporting more complex data operations or integrations without disrupting the core Django application.

Finally we have created a Dockerfile describing all the necessary environment specifications, dependencies versions and bundled the application into a Docker image along with its dependencies and configuration files. This allows us to easily integrate the Nexus web service alongside the rest of the infrastructure components.

6.3.2 Setting up Nexus and BOINC Environments with Docker

The foundational environment for the system is established by deploying the Docker service on a Linux-based virtual machine. The specific Linux distribution is inconsequential, as Docker operates directly with the kernel, which provides the necessary system-level

[2]`https://pymongo.readthedocs.io/en/stable/`, [Accessed on 29 March 2024]

abstractions for containerization. We then proceed to download the BOINC docker infrastructure [3] and modify its docker-compose.yml configuration file to incorporate our Nexus platform. This allows the services to reside on the same network and also consolidates the project under the same configuration.

After executing the docker compose up command from the Docker Compose utility, we end up with the deployed infrastructure. If the process was successful we are now able to interact with the Nexus web server on port 80 and with the BOINC management console on port 8081.

Using Docker provided an isolated environment that simplified the setup process and ensured consistency across different systems. Docker containers can be easily started, stopped, and scaled, offering flexibility during development and testing.

6.3.3 Worker Application Development

In this section we will explore the development process of the worker application, the challenges encountered when trying to deploy the first version on the Android mobile device and the solutions found to overcome the limitations.

The initial version of the worker application was implemented in Python, leveraging its robust library ecosystem for tasks like WHOIS queries, DNS retrieval, network analysis, and domain data collection, which are essential for feature extraction. It also integrates a pre-trained XGBoost machine learning model, managed with Scikit-learn, to predict domain reputation scores. While Python's flexibility and tool support are ideal for development, its reliance on an interpreter presents a challenge on Android devices, which lack a default Python environment. Building a Python interpreter for Android would conflict with the project's design principle of ensuring system accessibility without requiring modifications to user devices.

The Go programming language was selected for this project due to its simplicity, efficiency, and ability to meet the unique requirements of the system. One of Go's most compelling features is its capacity to compile all necessary libraries and code into a single, portable binary. This characteristic eliminates the need for external dependencies or complex runtime environments, significantly simplifying deployment across diverse platforms. Furthermore, Go supports cross-compilation, allowing applications to be built for various architectures and operating systems, including Android. This cross-platform compatibility is particularly advantageous for projects that target heterogeneous environments or mobile platforms.

To successfully build the application, first, it is imperative to download the Android NDK [4] which is a toolset that lets you implement parts of your app with native code. In our

[3] `https://github.com/marius311/boinc-server-docker`, [Accessed on 12 April 2024]
[4] `https://developer.android.com/ndk`, [Accessed on 22 April 2024]

case it will allow us to build a binary compatible with the Android's aarch64 architecture. Running the following bash script will produce a compatible binary:

```bash
#!/bin/bash

export NDK=/Users/dev/Library/Android/sdk/ndk/23.1.7779620
export CC=$NDK/toolchains/llvm/prebuilt/darwin-x86_64/bin/aarch64-linux
    -android21-clang
env GOOS=android GOARCH=arm64 CGO_ENABLED=1 CC=$CC go build -o
    worker_arm64
```

Listing 6.1: Bash script for building Android worker application.

After uploading to an Android emulator and running our program we can confirm that our worker application is able to execute successfully.

6.3.4 BOINC Application Wrapper

BOINC provides a C++ API for workunits to interact with its infrastructure, handling tasks such as starting/stopping processes, reporting progress, accessing files, and error reporting. However, integrating this API directly into a worker application tightly couples it with the BOINC platform, complicating future migration to alternative systems. To maintain flexibility, the "gridification" model, as proposed by Mateos et al.[28], was adopted, utilizing a wrapper application that incorporates the BOINC API and acts as a communication intermediary between the worker application and the BOINC server. Since BOINC does not supply binaries for the aarch64 architecture or Android, the wrapper was cross-compiled using the Android NDK's Clang toolchain. To prepare the worker application and its wrapper for secure distribution, they were uploaded to a structured directory on the BOINC server and signed using cryptographic keys. This process involved generating keys, signing the binaries, and updating the server configuration to make the application accessible to volunteer nodes. These measures ensured compliance with BOINC's security requirements, preserving the system's integrity and enabling efficient deployment.

6.3.5 Remote Job Submission

"Remote job submission" refers to the process where jobs are submitted to a BOINC server by scripts or programs operating externally, without requiring login access to the server. This is facilitated through the use of Web RPCs (Remote Procedure Calls), which allow for the creation of batches and jobs, as well as the management of input and output files. This method of submission is distinct from local submission, which involves direct interaction with the server.

Remote job submission introduces several key differences compared to local submission.

First, job submitters must possess a user account on the BOINC project. Submissions are linked to this account, and submitters must provide their credentials. Additionally, users need to be granted appropriate access rights and quotas to create jobs.

Jobs are organized and submitted in “batches,” even when only a single job is required; in such cases, a batch with one element must be created. Unlike local jobs, remote submissions do not utilize an assimilator to process output files. Instead, the “retire batch” operation is employed to indicate that the batch’s files and database records can be safely cleaned up.

The process of submitting jobs to the BOINC server is managed by the Nexus Domain Scheduler script. This script is responsible for determining the next set of domains that require validation or revalidation, based on the scheduling algorithm introduced earlier. Once the domains are identified, the script organizes them into a batch, which serves as the input for subsequent processing. By automating this task, the Nexus Domain Scheduler ensures efficient and systematic management of domain validation workflows within the BOINC framework.

6.3.6 Job Processing on Android Devices

The jobs are submitted to the Android devices with a replication factor of three, ensuring that each work unit is processed by at least three devices. This setup aligns with our statistical calculation indicating that, with a node compromise probability of 10%, assigning tasks to three devices achieves a 95% confidence level that the majority of devices processing any given task are honest.

To implement this, we configured the BOINC project settings to require a minimum quorum of three valid results (min_quorum=3) before considering a work unit complete. We also set the maximum number of error results and total results appropriately to handle any potential failures or discrepancies.

We instantiated three emulated Android devices using [Emulator Name], each configured with unique hardware profiles and connected to the BOINC project using individual user accounts. This ensured that each emulator acted as a distinct participant in the network. The BOINC client was installed on each emulator, and network settings were adjusted to enable communication with the BOINC server.

Upon deployment, the devices successfully registered with the project infrastructure and began receiving work units. The BOINC server distributed tasks in such a way that each work unit was sent to at least three devices. The devices processed the tasks using the Go-based worker application executed via the BOINC wrapper and returned the results to the server.

6.3.7 Results Validation

When processing workloads, discrepancies may arise due to differences in machine architecture, operating systems, network topology, or even malicious interference. To address this, a replication factor is employed, ensuring the aggregation of results that are statistically closer together. Worker applications return reputation scores along with JSON-formatted domain features, which are validated using an Interquartile Range (IQR) method to filter outliers. The mean of the validated scores is recorded as the definitive reputation score in the Nexus database, ensuring reliability and representativeness. Domain feature extraction similarly applies a consensus-based approach, retaining consistent data while discarding conflicting information, ensuring accuracy in the extracted features.

Variations in domain features may also arise due to differences in node locations, ISPs, proxy configurations, or blacklists in communication hardware. A consensus-based filtering mechanism mitigates these discrepancies, preserving only validated data for training and refining the machine learning model, which is subsequently updated for future workloads. All processes, including data validation, aggregation, and consensus establishment, are managed by a Python script integrated into the BOINC project's configuration. This script ensures robust and consistent validation during the consensus phase, enhancing the reliability of the distributed computing system.

6.3.8 Assimilation

The assimilation task is covered by the job submitter script using the BOINC API method get_output_files_url to retrieve the URLs of the output files that are being stored on the BOINC server and file_get_contents to obtain the results stored inside the files. The structure of the result is outlined as follows:

```
{
    "domain_age": 5,
    "domain_expiration_days": 365,
    "domain_length": 12,
    "dga_suspect": 0,
    "typosquatting_suspect": 0,
    "dns_ttl": "1800-86400",
    "dnssec": 1,
    "fast_flux_suspect": 0,
    "ssl": 1,
    "ssl_certificate_issuer": "Let's Encrypt",
    "ssl_certificate_valid_from": 120,
    "ssl_certificate_validity": 365,
    "registrar": "Namecheap",
    "ip_address": "192.0.2.1",
    "as_number": "AS12345",
    "is_blacklisted": 0,
    "is_ip_blacklisted": 0,
    "tls_version": "TLSv1.3",
    "strict_transport_security": 1,
    "content_security_policy": 1,
    "has_email_server": 1,
    "is_email_server_blacklisted": 0,
    "email_has_dmarc": 1,
    "email_has_spf": 1,
    "email_has_dkim": 1,
    "email_server_is_open_relay": 0
}
```

Listing 6.2: Sample worker response strcuture.

The contents of the file are stored in the Nexus database, ensuring they are available for future reference and for retraining the XGBoost model with an expanded dataset, thereby enhancing its predictive accuracy over time.

With the assimilation process successfully integrating the validated results into the Nexus database, we have completed the implementation of the distributed domain reputation system's proof of concept. This comprehensive implementation encompassed developing the worker application, setting up the BOINC infrastructure, processing jobs on Android devices, and ensuring data integrity through validation and assimilation.

Having established the system's foundational components, the next critical step is to evaluate its performance and effectiveness. In the following section, we will delve into the system evaluation, presenting the methodology employed, detailing the simulation setup, and analyzing the results obtained. This evaluation aims to assess the system's ability to meet the design objectives outlined earlier, examine its scalability and efficiency, and identify any areas for improvement.

6.4 System Evaluation

Evaluating the distributed domain reputation system is critical to validating its design and functionality. Due to the challenges of deploying the system with a large number of real-world participants, simulations were conducted to model its behavior under varying conditions. This evaluation involved analyzing performance, scalability, and effectiveness in achieving design objectives. Specifically, the simulation assessed the evaluation of 10,000 domains across a distributed network comprising 100 nodes, split evenly between desktop and Android devices. Desktop nodes, with their higher processing power and stable network connections, served as a performance benchmark, while Android nodes provided insights into mobile device participation, reflecting heterogeneous real-world conditions.

The simulation configuration allowed for a comparative analysis of resource utilization and task completion efficiency across node types, demonstrating the system's adaptability in diverse environments. Alongside evaluating worker nodes, the central server's load was monitored due to its pivotal role in job distribution, result validation, and database assimilation. These processes ensure data integrity and consistency while maintaining responsiveness to incoming queries. By identifying potential bottlenecks in the server's operations, the evaluation informed optimizations to enhance scalability and reliability, supporting the system's ability to manage real-world distributed computing workloads effectively.

6.4.1 CPU and Memory Impact

The evaluation of CPU and memory usage on desktop and Android devices demonstrated minimal resource consumption during job execution, with average completion times of 100 milliseconds for desktops and 200 milliseconds for Android devices, reflecting platform-specific processing capabilities. Profiling tools integrated into the worker application revealed negligible CPU usage differences between the two device categories and low memory allocation averaging 0.4 MB, confirming the application's lightweight and efficient design. While extended domain response times may occasionally lead to prolonged runtimes near the timeout limit, the application remains idle during these periods, consuming minimal resources. Optimizing domain response times or enhancing timeout management could further improve overall efficiency.

6.4.2 Network Impact

The network impact on nodes during the evaluation process is minimal due to the lightweight protocols used, such as WHOIS, DIG, and HTTP, which require only low bandwidth for routine operations. The most significant network usage occurs during the initial setup when the BOINC client downloads essential files, including the worker binary, wrapper application, and configuration XML. This setup is a one-time process; subsequent tasks involve negligible network usage as the BOINC client reuses locally stored files if no updates are required, thereby reducing redundant data transfers and optimizing overall system efficiency.

For the BOINC server, network activity primarily revolves around job file downloads by worker nodes and the upload of results. While experimental simulations showed synchronized network activity due to concurrent operations, real-world deployments are expected to have more distributed traffic patterns as devices register and request tasks asynchronously. This analysis highlights the need for server readiness to manage potential spikes in network activity, such as during high registration periods or simultaneous job completions. Anticipating these fluctuations is critical to maintaining server stability and ensuring the seamless operation of the distributed computing system. The proof of concept further validates the feasibility of the distributed architecture, showcasing its scalability, efficiency, and capacity to handle real-time domain reputation evaluation by leveraging decentralized computational power across diverse devices.

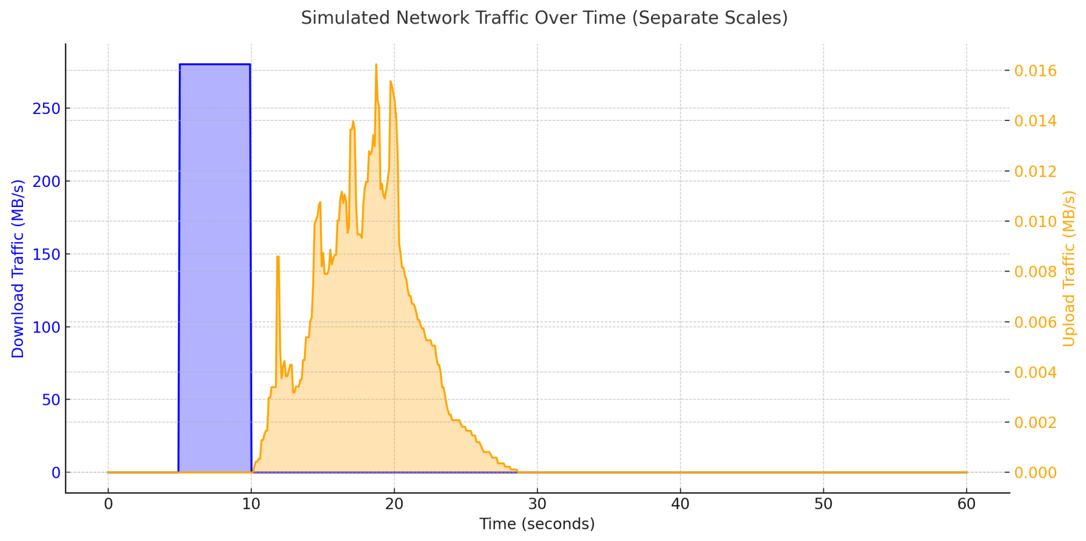

Figure 3: Simulated Network Traffic Over Time.

7 | Case Study on .ro Internet Domains

To date, our experiments have been conducted from an external vantage point, utilizing only publicly available datasets. While this approach enabled the collection of a subset of domain features, it inherently restricted the scope of our analysis to openly accessible information. However, through access to a privileged entity such as a Top-Level Domain (TLD) registry, we significantly expanded the depth and breadth of our research. This access unveiled insights into privately held domain features, including contact validation and historical domain records, which were previously beyond reach. By integrating these additional data points, we achieved a more comprehensive understanding of domain behavior and dynamics, enhancing both the accuracy and the robustness of our analyses. This allowed us to further expand our machine learning algorithm model achieving even more accuracy when predicting a domain's reputation score.

7.1 TLD Level Information Gathering

The WHOIS protocol for .ro domains is mandated by Romanian law[1] to redact any personal information pertaining to domains owned by natural persons. The regulation ensures compliance with data protection standards, allowing only the following information to be displayed:

- **Person Type**
- **Registration Date**
- **Expiry Date**
- **Domain Status**
- **Registrar Name**
- **Nameservers**

This restriction significantly limits the amount of information accessible for validating a .ro domain from a public perspective, posing challenges for comprehensive domain analysis and verification efforts.

[1]`https://eur-lex.europa.eu/legal-content/RO/TXT/PDF/?uri=CELEX:32016R0679`, [Accessed on 12 May 2024]

The contact validation flag, a component of privately held information, is established following a procedure designed to verify the identity of the domain's owner. For natural persons, this process typically involves submitting a personal identification document to confirm that the provided contact details are accurate and legitimate. In the case of companies, the risks are comparatively lower, as the domain registration process incorporates cross-referencing the company's identification details with data from authorized institutions. However, companies are still required to undergo the same verification procedure to ensure compliance.

The historical records maintained by the registry serve as a critical resource for understanding a domain's lifecycle, encompassing its initial registration, renewals, ownership transfers, and eventual deletion. These records provide comprehensive insights into the domain's past, enabling more accurate assessments of its behavior and reputation. From a public perspective, however, access to such information is limited. When a domain becomes available for registration, the primary indication of its prior registration is its presence on public blacklists. This lack of transparency often results in challenges during the validation of deleted domains.

By examining the historical records provided by the registry, it becomes possible to identify domains exhibiting Fast Flux behavior. These records offer detailed logs of domain reset operations, enabling the detection of patterns indicative of malicious activity. Additionally, the logs reveal the associated hosts and highlight whether the domain uses sub-domains as nameservers for other domains—a characteristic commonly linked to Fast Flux networks. This comprehensive visibility facilitates the identification of such domains with greater accuracy and efficiency, contributing to enhanced monitoring and mitigation efforts.

A streamlined version of the proposed architecture was implemented, primarily constrained by privacy considerations associated with involving volunteer participants. The deployment was carried out using a lightweight Kubernetes environment, Minikube[2], which served as the infrastructure for hosting both the Nexus and BOINC servers. Additionally, it facilitated the operation of worker nodes responsible for computing domain reputation scores based on the extracted domain features. This approach allowed for a controlled and efficient evaluation of the architecture while adhering to privacy and data protection standards.

The objective was to demonstrate the effectiveness of the machine learning model by enhancing its performance through the incorporation of additional information provided by the top-level domain (TLD). The newly included domain features, extracted as part of this refinement process, are detailed in Table 5.

Integrating the newly extracted features into the XGBoost model significantly enhanced its predictive capability, resulting in a 5% improvement in distinguishing between ma-

[2]`https://minikube.sigs.k8s.io/docs/start/`, [Accessed on 23 June 2024]

Feature	**Data type**	**Description**
Is contact valid?	Boolean	This flag
Nameserver changes	Numeric	Nameserver changes in the past year or since registration
Registrar transfer	Numeric	Number of times a domain had been transfered from a registrar to another
Ownership transfer	Numeric	Number of times a domain had it's ownership changed
Resolved domains	Numeric	The number of domains that the are being resolved by a nameserver hosted

Table 5: TLD advanced domain features.

licious and legitimate domains. Among the added features, the registrant validation flag had the most substantial impact, serving as a highly effective discriminator between legitimate and potentially malicious domains.

7.2 Conclusions

The implementation of this streamlined version of the architecture within the Romanian TLD facilitated the collection of valuable insights into domain features that are not publicly accessible. This demonstrates a significant opportunity for TLD operators to adopt a more proactive role in combating cybercrime by leveraging the advantage of possessing more granular and detailed information. This enhanced capability positions TLDs as critical players in the development of robust domain reputation systems.

8 | Conclusions

As we conclude this work, it is an opportune moment to synthesize the key findings, assess the contributions made to the field, and envision the pathways for future exploration and innovation.

8.1 Contributions

The proposed architecture introduces a novel approach to harnessing the computational potential of idle mobile devices for the purpose of establishing domain name reputations. By leveraging distributed computing principles, the system demonstrates the feasibility of transforming decentralized, underutilized resources into a cohesive and scalable solution for addressing computationally intensive tasks. While this work primarily focuses on presenting the architecture and validating its core principles, the results achieved during the course of this thesis provide compelling evidence of its practicality and effectiveness. It is our hope that this work will inspire further research and development in this area, encouraging both academic and industrial efforts to refine, implement, and expand upon the architecture to realize its full potential in real-world applications.

An additional significant contribution is the development of the reputation score decay function, which introduces a novel approach to the evaluation and periodic reevaluation of domain names. This methodology underscores the importance of providing a second opportunity for domain names to be registered by legitimate users, facilitating the eventual removal of such domains from blacklists. This approach aims to balance the need for cybersecurity with the potential for legitimate use of previously compromised or flagged domains.

Furthermore, this work has demonstrated that a modern and secure programming language, such as Go, can be effectively utilized to develop complex workunits compatible with resource-constrained platforms, such as the Android operating system. This highlights Go's versatility and suitability for addressing the challenges of cross-platform compatibility and efficient resource utilization, particularly in distributed computing environments.

8.2 List of Publications

1. Dragoș Smada, Mihail Dumitrache, Carmen-Ionela Rotună, Cristian-Alexandru Gheorghiță - The impact of internet domain name status on their reputation (Article, RRIA);

2. Mihail Dumitrache, Carmen Ionela Rotună, Alexandru Gheorghiță, Adrian Victor Vevera, Ionuț Sandu, Dragoș Smada - A Domain Reputation System Architecture Description Using TOGAF (Article, SIC, ISI);

3. Cristian-Alexandru Gheorghiță, Dragoș Smada, Adrian-Victor Vevera, Mihail Dumitrache, Ionuț-Eugen Sandu, Carmen-Ionela Rotună - Blacklists and whitelists in the framework of a domain reputation system (Article, RRIA);

4. Blockchain-based Decision Support System for Water Management - Bogdan-Ionuț Pahonțu, Diana-Andreea Arsene, Alexandru Predescu,Mariana Mocanu, Alexandru Gheorghiță (Article, SIC, ISI);

5. Carmen Ionela Rotună, Alexandru Gheorghiță, Ionut Sandu, Mihail Dumitrache, Meda Udroiu, Dragoș Smada - A Generic Architecture for Building a Domain Name Reputation System (Article, SIC, ISI);

6. Mihail Dumitrache, Ionuț-Eugen Sandu, Adriana-Meda Udroiu, Cristian-Alexandru Gheorghiță - Theoretical considerations about establishing the Internet domain reputation (Article, RRIA);

7. Alexandru Gheorghiță, Ionuț Petre - Securely Driving IoT by Integrating AIOps and Blockchain (Article, ROCYS);

8. Carmen Rotună, Alexandru Gheorghiță, Alin Zamfiroiu, Dragoș Smada - Smart City Ecosystem Using Blockchain Technology (Article, Informatică Economică);

9. Radu Boncea, Ionuț Petre, Victor Vevera, Alexandru Gheorghiță - Machine Learning Based Methods Used for Improving Scholar Performance (Article, eLearning & Software for Education);

10. Carmen Rotună, Carmen Cîrnu, Alexandru Gheorghiță - Implementing smart city solutions: Smart city map and city drop (Article, Calitatea Vieții);

11. Carmen Rotuna, Dragoș Smada, A Gheorghiță - Smart city applications built on big data technologies and secure IoT (Article, Ecoforum Journal);

12. Alexandru Gheorghita, Monica Anghel - Serious Games: An Oxymoron? (Proceedings, ICVL, ISI);

8.3 List of Projects

1. Core Program ”Advanced Research Based on Emerging and Disruptive Technologies - Support for the Society of the Future,” Objective: The Internet of the Future, Communications, Mobile Technologies; PN 2338 02 01 - ”Architecture – Intelligent Monitoring Platform for Internet Domains through the Development of a Dynamic Reputation Assessment System (TLDRep)” - Project Team Member; Deliverable Contribution;

2. Core Program ”Advanced Technologies and Services for the Development of the Information Society – TEHSIN,” Objective: 01 – Advanced Technologies for e-Services; PN 09 23 01 10 – ”Electronic Services Based on Cloud Computing Infrastructure” - Project Team Member; Deliverable Contribution;

3. Core Program ”Methods for Analyzing the Impact of Social Media on Governance Processes”; PN 09 23 06 01 – ”Electronic Services Based on Cloud Computing Infrastructure” – Project Team Member; Deliverable Contribution;

4. Sectoral Program ”Methods for Analyzing the Impact of Social Media on Governance Processes”; 145/2015 – ”Proposals for Solutions in Implementing the European Interoperability Framework at the National Level – Examples of Good Practices from EU Member States” – Project Team Member; Deliverable Contribution;

5. PN.802 - Expansion of Services Provided by the Online Platform for Evaluating Technical-Scientific Literature;

6. Online e-Participation System for Smart City Project Initiatives, PN 603;

7. Study on Adaptive Systems for Early Recognition of Cyber Attacks on State Resources - CS 76/19.06.2018;

8. New Research in Modeling and Optimizing Complex Systems with Applications in Industry, Business Environment, and Cloud Computing - PN101/2018;

9. PN101/2019 - Research on Advanced Policies and Solutions for Securing Critical Infrastructures Against Cyber Attacks;

10. PN 301/2019 - Non-Invasive Monitoring and Evaluation System for the Health of Elderly People in an Intelligent Environment Ro - SmartAgeing;

8.4 Future Work

For future work, we plan to extend the scope of the system by exploring the inclusion of additional device categories, such as Internet of Things (IoT) devices, into the computational pool. This expansion aims to further leverage the growing ecosystem of connected devices, enhancing the system's scalability, diversity, and overall computational capacity.

Additionally, future efforts will focus on addressing the challenge of machine learning model size by implementing a more lightweight alternative, namely LightGBM [1]. LightGBM is specifically designed to reduce memory usage while maintaining high performance, offering faster and more efficient model training. This adaptation is expected to improve the system's compatibility with resource-constrained devices and enhance overall operational efficiency without compromising predictive accuracy.

Finally, we aim to explore the possibility of decentralizing the storage system by implementing a blockchain-based solution across the participating nodes. Such an approach would not only eliminate the reliance on a centralized server managing a traditional database but also provide an immutable and inherently more secure framework for data storage. By leveraging blockchain technology, the system would benefit from enhanced transparency, tamper resistance, and decentralized consensus, addressing potential vulnerabilities associated with centralization and further strengthening the overall robustness and trustworthiness of the architecture.

8.5 Lessons Learned

One of the most profound lessons I have learned during the course of this work is encapsulated in a quote from Bjarne Stroustrup, shared with me by a colleague: "If you think it's simple, then you have misunderstood the problem." This single sentence resonates deeply and succinctly captures the essence of the challenges encountered throughout the documentation and implementation phases of this project.

Tasks that initially appeared straightforward, such as compiling a binary for a Linux-derived operating system like Android, revealed themselves to be far more complex than anticipated. These underestimated challenges served not only as obstacles but also as valuable opportunities for growth, fostering a deeper understanding of the intricacies

[1]`https://lightgbm.readthedocs.io/en/stable/`, [Accessed on 12 May 2024]

involved in system development. Such hurdles, while demanding, enrich the journey and inspire a continued pursuit of knowledge and exploration in this ever-evolving field.

Bibliography

[1] Landers, Joerg Weissmann, L Steinbrinker, Uwe Kraemer, Stefan Weinert, and H.-J Lüddeke. Notos: an electronic questionnaire structures individualised diabetes management. In *NOTOS: an electronic questionnaire structures individualised diabetes management*, 09 2013.

[2] Leyla Bilge, Sevil Sen, Davide Balzarotti, Engin Kirda, and Christopher Kruegel. Exposure: A passive dns analysis service to detect and report malicious domains. *ACM Transactions on Information and System Securi*, 16, 04 2014. doi: 10.1145/2584679.

[3] Manos Antonakakis, Roberto Perdisci, Wenke Lee, Nikolaos Vasiloglou, and David Dagon. Detecting malware domains at the upper dns hierarchy. In *Detecting malware domains at the upper DNS hierarchy*, pages 27–27, 08 2011.

[4] Michael Shirts and Vijay Pande. Screen savers of the world unite. *Science (New York, N.Y.)*, 290:1903–4, 01 2001. doi: 10.1126/science.290.5498.1903.

[5] Hitesh Bheda. Application processing approach for smart mobile devices in mobile cloud computing. *International Journal of Software Engineering and Knowledge Engineering*, 3:1046–1054, 08 2013.

[6] Joshua Jaffe and Luciano Floridi. Ransomware: Why it's growing and how to curb its growth. *Applied Cybersecurity & Internet Governance*, 11 2024. doi: 10.60097/ACIG/192959.

[7] Elahe Soltanaghaei and Mehdi Kharrazi. Detection of fast-flux botnets through dns traffic analysis. *Scientia Iranica*, 22, 01 2016.

[8] Valentin Manescu, Neghina Alexandra, Andreea Barbu, Ganciu Rodica, and Gheorghe Militaru. Analysis of ssl certificates trends and extended validation ssl usage for e-commerce websites and internet of things. *UPB Scientific Bulletin, Series C: Electrical Engineering*, 83, 12 2021.

[9] HuffPost. Domain theft: Who owns your website?, 2014. URL `https://www.huffpost.com/entry/domain-theft_n_5877510`. Accessed: 2023-01-25.

[10] Paul Mockapetris and Kevin J Dunlap. Development of the domain name system. In *Symposium proceedings on Communications architectures and protocols*, pages 123–133, 1988.

[11] Aminollah Khormali, Jeman Park, Hisham Alasmary, Afsah Anwar, Muhammad Saad, and David Mohaisen. Domain name system security and privacy: A contemporary survey. *Computer Networks*, 185:107699, 2021.

[12] Deepak Kumar Vishwakarma. Domain name generation algorithms. *Masaryk University*, 2017.

[13] Frank Swiderski and Window Snyder. *Threat modeling*. Microsoft Press, 2004.

[14] Yunyi Zhang, Mingming Zhang, Baojun Liu, Zhan Liu, Jia Zhang, Haixin Duan, Min Zhang, Fan Shi, and Chengxi Xu. Cross the zone: Toward a covert domain hijacking via shared {DNS} infrastructure. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 5751–5768, 2024.

[15] U Steinhoff, A Wiesmaier, and R Araújo. The state of the art in dns spoofing. In *Proc. 4th Intl. Conf. Applied Cryptography and Network Security (ACNS)*. Citeseer, 2006.

[16] K Jansson and Rossouw von Solms. Phishing for phishing awareness. *Behaviour & information technology*, 32(6):584–593, 2013.

[17] Andi Fitriah Abdul Kadir, Raja Azrina Raja Othman, and Normaziah Abdul Aziz. Behavioral analysis and visualization of fast-flux dns. In *2012 European Intelligence and Security Informatics Conference*, pages 250–253. IEEE, 2012.

[18] Zubair Jeelani. An insight of ssl security attacks. *International Journal of Research in Engineering and Applied Sciences*, 3:52–61, 2013.

[19] Huiju Lee, Jeong Do Yoo, Seonghoon Jeong, and Huy Kang Kim. Detecting domain names generated by dgas with low false positives in chinese domain names. *IEEE Access*, 2024.

[20] Tong Anh Tuan, Hoang Viet Long, and David Taniar. On detecting and classifying dga botnets and their families. *Computers & Security*, 113:102549, 2022.

[21] Craig Beaman, Ashley Barkworth, Toluwalope David Akande, Saqib Hakak, and Muhammad Khurram Khan. Ransomware: Recent advances, analysis, challenges and future research directions. *Computers & security*, 111:102490, 2021.

[22] Dimitrios Georgoulias, Jens Myrup Pedersen, Morten Falch, and Emmanouil Vasilomanolakis. A qualitative mapping of darkweb marketplaces. In *2021 APWG Symposium on Electronic Crime Research (eCrime)*, pages 1–15. IEEE, 2021.

[23] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 785–794. ACM, August 2016. doi: 10.1145/2939672.2939785. URL `http://dx.doi.org/10.1145/2939672.2939785`.

[24] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[25] Lean Yu, Rongtian Zhou, Rongda Chen, and Kin Keung Lai. Missing data preprocessing in credit classification: One-hot encoding or imputation? *Emerging Markets Finance and Trade*, 58(2):472–482, 2022.

[26] Santhanam Ramraj, Nishant Uzir, R Sunil, and Shatadeep Banerjee. Experimenting xgboost algorithm for prediction and classification of different datasets. *International Journal of Control Theory and Applications*, 9(40):651–662, 2016.

[27] Jiawei Han and Micheline Kamber. Data mining: Concept and techniques (s˜ ond edition), 2006.

[28] Cristian Mateos, Alejandro Zunino, and Marcelo Campo. A survey on approaches to gridification. *Software: Practice and Experience*, 38(5):523–556, 2008.