University of the Peloponnese
Department of Informatics and Telecommunications
Ac. Vlachou, 22100, Tripoli
Tel. 2710-230177
Fax. 2710-372160

A System for Targeted Dark Web Crawling and Content Analysis

Christina Michalaki
Stamatina Gkouna

M.Sc. Thesis

Tripoli, June 2026

A System for Targeted Dark Web Crawling and Content Analysis

Christina Michalaki

Stamatina Gkouna

M.Sc. Thesis

Software and Database Systems Lab

Department of Informatics and Telecommunications, University of the Peloponnese

University of the Peloponnese
Department of Informatics and Telecommunications
Ac. Vlachou, 22100, Tripoli
Tel. 2710-230177
Fax. 2710-372160

A System for Targeted Dark Web Crawling and Content Analysis

Christina Michalaki
Stamatina Gkouna

M.Sc. Thesis

Supervisor:

Prof. Christos Tryfonopoulos, University of the Peloponnese

Examination committee:

Prof. Nicholas Kolokotronis, University of the Peloponnese

Assist. Prof. Nikolaos V. Platis, University of the Peloponnese

Acknowledgements

First of all, we would like to express our deepest gratitude to our supervisor, Prof. Christos Tryfonopoulos, for his invaluable guidance, continuous support, patience and understanding throughout the duration of this thesis. His insightful feedback and academic expertise was instrumental to the successful completion of this work.

We would also like to sincerely thank the members of our examination committee for their time and critical evaluation of our research.

Last but not least, we would like to thank our families and friends for their continuous support, encouragement and understanding during this demanding period of academic and life transitions.

Περίληψη

Στη σημερινή εποχή, το Διαδίκτυο αποτελεί τη βασική υποδομή που υποστηρίζει την παγκόσμια επικοινωνία και τις οικονομικές δραστηριότητες σε όλο το σύγχρονο ψηφιακό περιβάλλον. Ωστόσο, κάτω από την επιφάνειά του, το Dark Web λειτουργεί ως ένα μη ανιχνεύσιμο δίκτυο που επιτρέπει σε παράνομες δραστηριότητες να παραμένουν κρυφές χάρη στην ανωνυμία που προσφέρει. Η απουσία κεντρικής οργάνωσης σε συνδυασμό με την αστάθεια των ιστότοπων δημιουργούν ένα απρόβλεπτο περιβάλλον που καθιστά εξαιρετικά δύσκολη την εξαγωγή δεδομένων με αξιόπιστες μεθόδους. Στην παρούσα εργασία, παρουσιάζουμε μια αρχιτεκτονική ανίχνευσης που εντοπίζει και ανακτά δεδομένα με αυτοματοποιημένες μεθόδους παίρνοντας πρόσβαση μέσω του Tor, προκειμένου να πραγματοποιήσει μια διεξοδική ανάλυση. Μέσω αυτής της ανάλυσης, αποκαλύπτει τις τάσεις στην τεχνολογία, τις προτιμήσεις κρυπτονομισμάτων που φαίνεται να διαμορφώνουν αυτές τις υπηρεσίες και τα γλωσσικά μοτίβα μεταξύ της βάσης χρηστών. Το σύστημα επιτρέπει έτσι στους αναλυτές να μετατρέπουν μη δομημένα δεδομένα του Διαδικτύου σε οργανωμένες πληροφορίες, οι οποίες τους βοηθούν να κατανοήσουν καλύτερα το οικοσύστημα του Dark Web.

Abstract

Nowadays, the Internet serves as the core infrastructure which supports global communication and economic activities throughout the modern digital environment. However, behind the world's surface, the Dark Web functions as an untraceable digital network which enables elicit operations to stay hidden through its anonymity. The absence of central organization along with website volatility create an unpredictable environment which makes data extraction through reliable methods extremely difficult to achieve. In this thesis, we present a crawling architecture which detects and retrieves data from Tor's hidden network in order to perform a thorough analysis. Through this analysis, we reveal the current and growing trends in technology, cryptocurrency preferences that appear to shape these services, and language patterns among the user base. Thus, the implemented system allows analysts to convert unstructured internet data into organized information, which helps them to get a better understanding of the Dark Web ecosystem.

Contents

List of Figures

List of Tables

List of Abbreviations

Acronym	Definition
ACHE	Automated Crawler for Hyperlinks and Entities
API	Application Programming Interface
BFS	Breadth-First Search
BIOS	Basic Input/Output System
CPU	Central Processing Unit
CTI	Cyber Threat Intelligence
CVE	Common Vulnerabilities and Exposures
DDoS	Distributed Denial of Service
DNS	Domain Name System
DOM	Document Object Model
GUI	Graphical User Interface
HTML	Hypertext Markup Language
HTTP	HyperText Transfer Protocol
JSON	JavaScript Object Notation
JVM	Java Virtual Machine
LTS	Long Term Support
ML	Machine Learning
NoSQL	Not Only SQL
OS	Operating System
PoC	Proof of Concept
PPM	Pages Crawled Per Minute

RAM	Random Access Memory
REST	Representational State Transfer
SOC	Security Operations Center
SOCKS	Socket Secure
SQL	Structured Query Language
SVM	Support Vector Machine
TCP	Transmission Control Protocol
Tor	The Onion Router
UI	User Interface
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VIDA	Visualization, Imaging, and Data Analysis Center
VM	Virtual Machine
WSL	Windows Subsystem for Linux
XML	Extensible Markup Language

Chapter 1

Introduction

This thesis presents a specialized architecture for the targeted crawling, harvesting, analysis and visualization of unstructured content from the Dark Web (Tor network) [8]. In this chapter, we establish the particular difficulties which hidden services present because they operate with anonymity and volatility and we present our method to retrieve data while mapping the underground ecosystem.

1.1 The Dark Web

The Internet functions as an advanced network system which distributes information through multiple levels which extend past what standard search engines can display. Standard browsers enable users to access the web through its surface content which makes up only a small portion of the entire web system. Users need to identify between public content and protected data which requires authentication or special access methods because digital information hides in invisible spaces.

The web hierarchy consists of three distinct sections based on accessibility and anonymity. The Surface Web is the portion indexed by standard search engines (e.g., Google, Bing). Deep Web contains the unindexed content such as databases and private networks, while Dark Web is a subset of the Deep Web that requires specific software to access [11, 16].

1.1.1 Surface, Deep and Dark Web Characteristics

The architectural differences between the Surface Web, Deep and the Dark Web impose strict constraints on data harvesting. Table 1.1 outlines these critical distinctions.

Table 1.1: Comparative Analysis: Surface, Deep and Dark Web

Feature	**Surface Web**	**Deep Web**	**Dark Web**
Definition	Publicly accessible pages indexed by standard search engines.	Content not indexed by search engines, usually requiring authentication.	A subset of the Deep Web intentionally hidden on overlay networks.
Access Method	Standard Browsers (Chrome, Firefox, Edge).	Standard Browsers (requires Login/VPN/Auth).	Specialized Software (Tor Browser, I2P).
Addressing	DNS (Standard Domains like .com, .org).	DNS (Standard Domains, Intranets, Subdomains).	Cryptographic Hashes (e.g., .onion).
Indexing	Fully Indexed (Google, Bing, Yahoo).	Not Indexed (Blocked by robots.txt or password).	Not Indexed (Inaccessible to standard crawlers).
Anonymity	**Low:** Users are tracked via IP and cookies.	**Moderate:** Users are identified by accounts (e.g., Bank ID).	**High:** IP addresses are masked via routing.
Content	News, E-commerce, Blogs, Marketing websites.	Medical records, Banking, Academic databases, Intranets.	Illegal marketplaces, Whistleblowing, Hacking forums.
Est. Size	Smallest portion ($\sim$4% of the Web).	Largest portion ($\sim$90-95% of the Web).	Small subset (Hard to measure, $<$1%).

Research indicates that the life cycle of a typical dark web marketplace can be as short as a few weeks [19, 27]. Hidden services are often hosted on personal computers, temporary virtual private servers, or compromised infrastructure, resulting in significantly lower uptime compared to the 99.9% availability of Surface Web servers. This creates a "moving target" problem, where a crawler must revisit pages frequently to verify their continued existence before attempting to harvest data.

Universal crawlers on the Surface Web utilize hyperlinks from high-traffic hubs (like Wikipedia or news portals) to discover new content naturally. In contrast, there are no central indexing authorities or Search Engine Optimization (SEO) tools on the Dark Web. Hidden services are intentionally isolated to maintain opacity, preventing crawlers from using standard graph traversal. Instead, the harvesting process often requires "seed

lists" from specialized wikis or forums, manually curated to ensure effectiveness [1, 2].

1.2 Problem Statement

While there is an urgent need for monitoring Dark Web traffic, its intelligence extraction system faces significant technical challenges and operational challenges [17]. Unlike the Surface Web, which is built for deliverability, the Dark Web is designed for obfuscation. The specific problems addressed by this thesis are categorized as follows.

1.2.1 The Discovery and Indexing Challenge

The most fundamental hurdle is the lack of a centralized registry. On the Surface Web, DNS (Domain Name System) broadcasts the existence of websites. On the contrary, hidden Services in the Tor network operate with hash-based addresses (.onion) but those remain hidden from public view [8]. Since there is no Google for the Dark Web, security scholars frequently find themselves depending on fragmented lists, manual discovery or inefficient crawling methodologies that don't really capture the whole length or breadth of the network [3, 6].

1.2.2 Volatility

Dark Web sites are notoriously unstable. A significant percentage of onion services have an uptime of less than 24 hours, according to studies [4, 9, 27]. This extreme volatility is mainly shown in two behaviors: migration and sudden disappearance of sites. On the one hand, administrators often perform URL migrations to avoid law enforcement operations or to reduce Distributed Denial of Service (DDoS) attacks launched by rival gangs [23]. On the other hand, many malicious actors conduct "exit scams", a common phenomenon where a marketplace suddenly disappears along with the escrowed funds of users and reappears later with a completely new infrastructure and identity [16]. This volatility makes it difficult to maintain a time-based dataset. A single site visit from a crawler results in no future discovery of that site which produces intelligence gaps.

1.2.3 Technical Barriers

Crawling the Dark Web requires navigating a hostile environment.Access requires tunneling traffic through the Tor SOCKS5 proxy, which causes major delays and network connection breakdowns compared to standard HTTP requests [8, 11]. These delays are caused by the fact that data packets are going through a series of encrypted relays, which affects the throughput of automated tools. Thus, a crawler operating in this environment has to implement aggressive timeout settings, robust retry strategies and session management in order to ensure operational stability in the face of the underlying network unreliability.

Finally, the dark web hidden services implement powerful anti-crawling protection systems like CAPTCHAs, cryptographic challenges and invite-only authentication to block automated scrapers [7, 23]. These obstacles require substantial computational resources or manual human intervention to overcome.

1.3 Motivation

The motivation behind this research derives from the rapid shift in the global threat landscape. The cybercrime industry now operates as a service-based model which people refer to as Cybercrime-as-a-Service (CaaS) [23]. The Dark Web marketplaces and forums operate as central hubs which support a large underground economy that enables people to exchange illegal items including digital assets and physical goods. These anonymous platforms serve to allow sophisticated threat actors to operate complex digital storefronts.

In light of this operational maturity, there is a clear need for systematic and automated approaches that provide a holistic view of the Dark Web landscape. The research aims to map what exists in these anonymous networks, to understand the dynamics and profiles of the actors operating behind them and to monitor emerging infrastructure and financial trendsby translating unstructured data into meaningful statistical analytics and visual representations. [12].

1.4 Our Contribution

In this work, we focus on the data harvesting and ecosystem analysis aspects of the Dark Web and present an architecture that is able to provide a robust crawling and visualization infrastructure for hidden services (Onion sites). Our approach uses a crawling methodology in order to guide the harvest toward high-value services, effectively overcoming the noise and instability of the Tor network. This is achieved through a combination of filtering algorithms (for initial seed verification from directories like Ahmia) and regex-based crawling (utilizing the ACHE framework[1] in a Dockerized environment [2, 15]).

The retrieved content is stored in an efficient NoSQL datastore (Elasticsearch[2]) and is aggregated for visual inspection in order to extract meaningful patterns regarding the technological and financial state of the underground economy. Those patterns are then represented using visual dashboards via Grafana[3]. In summary, this method converts raw, unstructured web content into structured visual intelligence, enabling the mapping of the Dark Web's ecosystem.

1.5 Thesis Structure

The rest of the thesis is organized to cover both the theoretical framework and the technical implementation of the proposed system. Specifically, in **Chapter 2** we present the related work on Web crawlers and discuss about the different architectures and typologies. **Chapter 3** details the design of the proposed solution. In that section, the engineering choices and containerized deployment blueprints of our proposed pipeline are discussed. **Chapter 4** presents the core findings derived from the collected dataset. This chapter displays the visualized data, revealing a deep understanding of the linguistic demographics, technical infrastructures and financial behaviors of the Dark Web ecosystem. Finally, in **Chapter 5** we summarize the work done in the thesis. Limitations experienced during the research are shown in that section and possible future

[1] `https://ache.readthedocs.io/`
[2] `https://www.elastic.co/elasticsearch/`
[3] `https://grafana.com/`

prospects for extending the system are suggested.

Chapter 2

Background

To understand the architectural decisions made in this thesis, it is essential to trace the historical evolution of Web Crawling technologies. The transition from Surface Web indexing to Dark Web mining represents a fundamental shift in algorithmic requirements [3, 12].

The current chapter contains an essential historical assessment of web crawling techniques. The research starts by studying the original Universal Crawlers which includes Apache Nutch[1] and Heritrix[2] to understand their operational principles and their restrictions when operating in networks with high latency and anonymity. The discussion then shifts to the next generation script-based frameworks which included Scrapy[3] as an example of flexible systems that failed to provide sufficient stability. Finally, we present the theoretical framework of ACHE system, positioning it as the most sophisticated method for targeted Dark Web harvesting [2, 15].

2.1 Tor Architecture

Accessing the Dark Web requires going through the Tor (The Onion Router) network. Tor is an overlay network that was developed by the U.S. Naval Research Laboratory. It is meant to keep TCP-based applications private. The Tor network operates by sending

[1] `https://nutch.apache.org/`
[2] `https://github.com/internetarchive/heritrix`
[3] `https://scrapy.org/`

traffic through multiple volunteer-operated relays which function differently than the Surface Web because packets travel straight from clients to servers. The main focus of Tor exists in privacy protection rather than in achieving high-speed data transfer rates [6, 8].

The main innovation of Tor exists in its Onion Routing system which uses multiple encrypted layers for its operation. Data in Tor is routed through a random path of relays and the Tor client software builds a network path, the Circuit. The client needs to get active relay data from trusted Directory Authorities to create this circuit. The system chooses three network nodes to create a telescoping path which encrypts the data packet through three encryption processes for each node in the sequence. As depicted in Figure 2.1, a standard circuit consists of three nodes [8]:

1. **Guard Node:** The entry point to the network. This is the only node that knows the client's (crawler's) real IP address. However, it cannot see the content of the traffic or the final destination. It takes off the top layer of encryption to show the address of the next hop.

2. **Middle Node:** This node acts as a bridge. It receives encrypted data from the Guard and passes it to the Exit. The Middle Node maintains complete privacy for communication because it does not have any information about the starting point (Client) or the ending point (Hidden Service).

3. **Exit Node:** The final hop before the destination.It decrypts the innermost layer of encryption and delivers the payload to the destination. The Exit node enables users to access .onion services through its connection to the point which establishes the handshake without revealing the crawler's IP address to the server [8].

This architecture ensures perfect forward secrecy: no single node in the circuit knows the complete path.

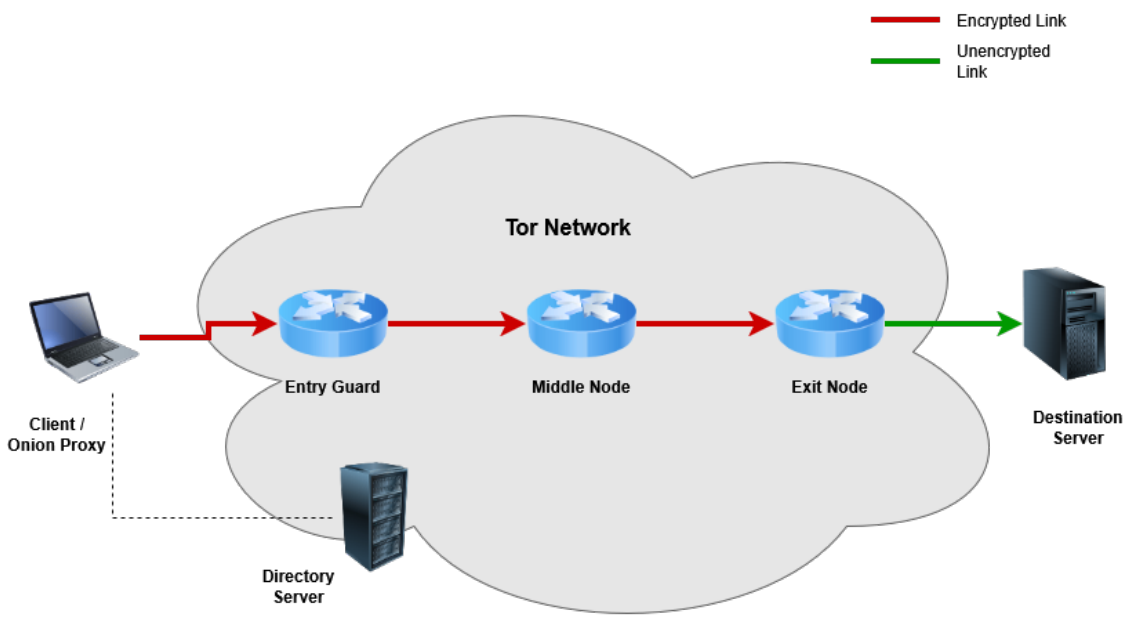


Figure 2.1: The Tor Circuit and Onion Routing Mechanism

2.2 Crawler Architectures

The crawling process can be represented through graph traversal algorithms which operate on the World Wide Web structure where pages function as vertices and hyperlinks serve as directed edges. The crawler determines its traversal sequence through a *Frontier* structure which helps it achieve the most efficient path between nodes. The system performs a fetch-and-parse operation at each node (URL) to discover neighboring nodes which represent new links. The state management system that appears in Figure 2.2 represents a fundamental element of this architecture because it removes all previously explored website addresses (visited URLs). The basic algorithmic structure remains simple but the system requires advanced engineering solutions to handle storage growth and maintain system reliability [18].

The literature distinguishes between various crawler architectures based on their scope and traversal strategy. Generally, crawlers are categorized as **Universal Crawlers** and **Focused Crawlers** [3, 18]. Universal Crawlers are designed to index the entire web (e.g., Googlebot). Coverage matters most to them, yet they struggle when handling large amounts of online data. On the other hand, as utilized in this

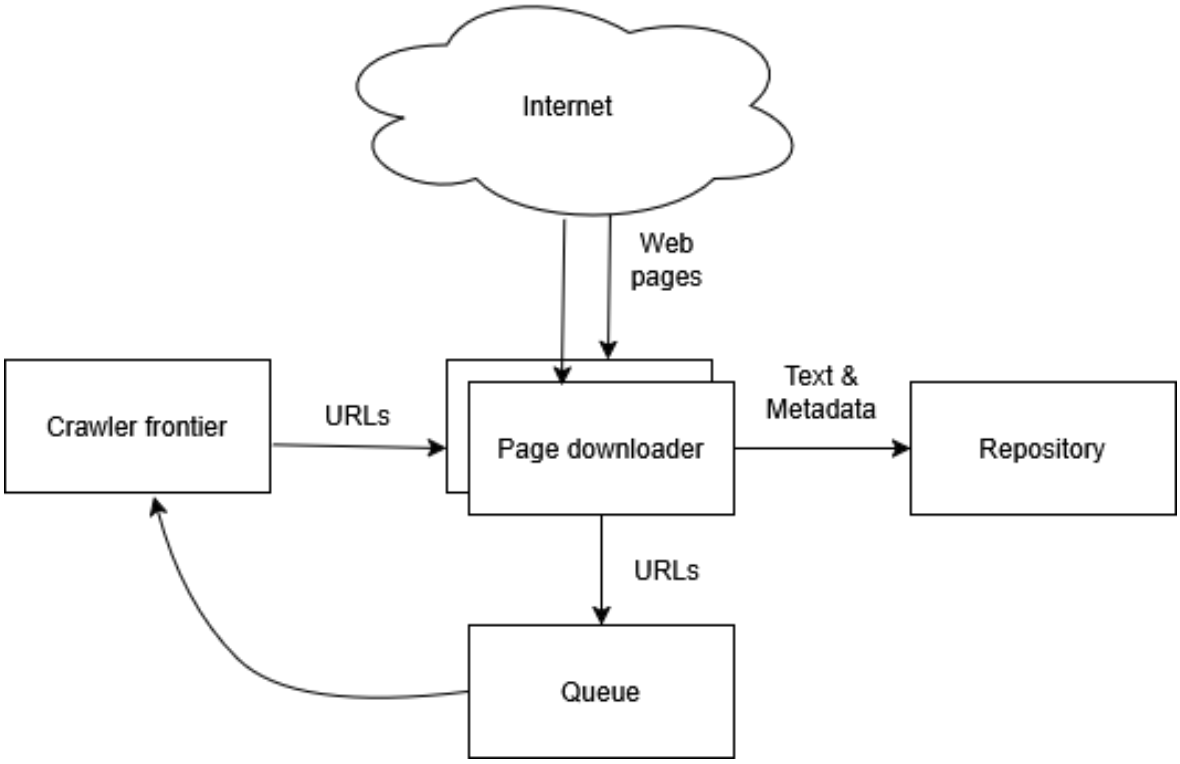


Figure 2.2: Basic Crawler Architecture

thesis, Focused Crawlers seek to harvest only a subset of the web relevant to a specific topic defined by keywords or classifiers [2].

Table 2.1 presents the main distinctions between these methods which demonstrate that Focused Crawler provides better results for Dark Web investigations [12].

Table 2.1: Comparison: Universal vs. Focused Crawlers

Feature	**Universal Crawler**	**Focused Crawler**
Goal	Index the entire web graph.	Index specific topics only.
Resource Usage	Extremely High (Bandwidth/Storage).	Optimized/Low.
Suitability for Tor	Low (Too slow for Tor latency).	High (Targeted fetching).

2.3 The First Generation: Universal Crawling Architectures

The early literature on web crawling, driven by the explosive growth of the World Wide Web in the late 1990s, focused on a single objective: **Coverage** [5]. Systems like the *Mercator* crawler and later *Apache Nutch* were designed to download as much of the web as possible.

2.3.1 Breadth-First Search (BFS)

Universal crawlers function through the use of graph traversal algorithms which they execute. Modeling the web as a directed graph $G = (V, E)$, where V are pages and E are hyperlinks, these systems typically employ a **Breadth-First Search (BFS)** strategy [18].

The BFS algorithm operates through a First-In-First-Out (FIFO) queue which stores all discovered URLs. The algorithm provides the best results for graph structure exploration in static graph environments, yet it fails to deliver optimal performance when used for Dark Web exploration because it contains two major weaknesses: **Traversal Inefficiency** and **Spider Traps**. BFS treats all links as equal. Users need to wait 5-10 seconds for the Tor network to process HTTP requests, so they should only click on essential links. Moreover, universal crawlers become trapped in endless loops which emerge from dynamic calendar systems and session ID mechanisms that Dark Web forums use to protect themselves from bots.

2.3.2 Apache Nutch

Apache Nutch operates as the primary system which enables organizations to conduct extensive web crawling operations at scale. The system operates on Java technology which brought the MapReduce programming model to manage large-scale data processing needs.

Nutch divides its crawling operations into three separate cycles. The Generator

first creates the fetch list and determines which URLs will be fetched. Second, the Fetcher is responsible for the actual content retrieval from the identified targets. Finally, the DbUpdater performs the graph updates by modifying the system's database to represent the newly found link structures and nodes.

The system architecture of Nutch depends on fast network connections because it does not support SOCKS5 proxy routing efficiently. The system enforces "politeness" rules through its established time-based structure which creates defined periods between successive requests. The performance of Nutch threads becomes poor because they stay idle for extended periods when the Tor network experiences unpredictable latency variations [12]. Furthermore, Nutch requires a massive infrastructure, which is excessive for the smaller, but more complex, Dark Web graph.

2.4 The Second Generation: Modular Frameworks

The development of more detailed data mining requirements led scientists to abandon their Java-based crawling systems for Python-based frameworks which provided better flexibility. The dominant tool in this era is **Scrapy**.

2.4.1 Event-Driven Architecture

Scrapy differs from Nutch by utilizing an asynchronous, event-driven architecture. The system enables concurrent operation through this approach which avoids the performance costs that result from using numerous threads.

Scrapy introduced the **Middleware** concept which enables developers to add their own logic to the request-response process through the pipeline. The introduction of this tool brought a major breakthrough to Dark Web scraping operations because scientists gained the ability to create their own middleware solutions which enabled them to change Tor User Agents and solve particular HTTP errors including 502 Bad Gateway.

2.4.2 The Bottleneck

The framework nature of Scrapy prevents it from serving as a standalone solution. Scrapy does not come with a built-in persistent URL frontier. The system loses its current state when the crawler experiences a crash, which happens frequently in unstable Tor network conditions. Moreover, the researcher needs to develop manual link filtering logic because Scrapy requires this to achieve effectiveness on Dark Web platforms. The process of writing particular Regular Expressions (Regex) becomes necessary to prevent the crawler from accessing User Profile pages in forums. The method proves ineffective for large-scale onion service identification because it cannot handle thousands of unidentified onion services.

2.5 Focused Crawling & ACHE

The limited functionality of BFS (Nutch) and Scrapy's need for user participation made us choose **Focused Crawling** as our primary method to extract Dark Web data [2]. The concept changes the objective from achieving complete coverage to achieving relevant results.

2.5.1 Theoretical Basis: Best-First Search

Focused crawlers replace the FIFO queue of BFS with a Priority Queue. The system directs its restricted bandwidth toward links which seem to provide important information through its link filtering mechanism (the system follows links containing "Bitcoin" or "Database" but avoids "Copyright" and "CSS" links).

2.5.2 ACHE

ACHE (Automated Crawler for Hyperlinks and Entities) serves as the following development stage. The system operates as an automated version of the "Best-First" method which it uses to perform searches within particular domains [2].

The system combines Java's strong robustness from Nutch with Scrapy's adaptable target definition capabilities to address the three main difficulties of the "Dark Web Triad": **Volatility, Latency** and **Anonymity**. In order to address Volatility, the system maintains a stable frontier which Scrapy lacks. The system allows crawl interruptions which result in no loss of data during the resume process [15]. Moreover, the Link Classifier removes most unimportant network traffic before the system starts connection attempts; thus, it accelerates the crawling process by dismissing unnecessary data [2]. Finally, Anonymity is being addressed by enabling users to access ACHE through its native SOCKS5 proxying functionality which operates directly from the core configuration without requiring unstable middleware wrappers.

The evolutionary process shows that ACHE functions as a required adaptation of crawling technology which must operate within the dangerous conditions of the Tor network [3, 12]. Table 2.2 provides an overview of the structural, operational and performance characteristics of the discussed crawler generations to give a complete picture of their architectural evolution over time.

2.6 Relevant Approaches

In order to provide context for the architectural choices of our system, it is important to examine the prominent frameworks developed by Koloveas et al. [12, 13]. The mentioned works are the main academic benchmarks for dark web data collection and laid the foundation methodology for multi-tier web harvesting. Yet, although we share the goal of harvesting Dark Web data, our architecture shifts the focus to a visualization-centric execution pipeline.

The first framework is a comprehensive, multi-level crawler architecture proposed by Koloveas et al. [12] that aims to discover and collect data across three different web layers at the same time: the Clear Web, the Social Web and the Dark Web. The basic philosophy of this system was the proactive collection of specific Internet of Things (IoT) ecosystem cyber threat indicators. Their work showed that automated discovery routines, when run in a timely manner, could be effective in identifying corporate leaks

Table 2.2: Evolutionary Comparison of Web Crawling Frameworks

Metric / Feature	**First Generation (e.g., Apache Nutch)**	**Second Generation (e.g., Scrapy)**	**Focused Generation (e.g., ACHE)**
Core Objective	Maximum graph coverage & page volume.	Flexible scripting & modular scraping.	Contextual relevance & dynamic threat discovery.
Traversal Strategy	Breadth-First Search (BFS) using FIFO queue.	Manual/User-defined path iteration.	Best-First Search guided by link classifiers.
State Management	Centralized Database (requires massive cluster).	Volatile memory (loss of frontier state on crash).	Persistent frontier with native pause/resume support.
Tor Proxy Integration	Non-native (poor thread reuse over high latency).	Middleware wrappers (prone to connection leaks).	Native SOCKS5 routing integrated at core config.
Target Optimization	Unoptimized (treats login/error pages equally).	Manual Regex filtering (high maintenance overhead).	Automated Machine Learning & regex page classifiers.
Infrastructure Cost	Extremely High.	Moderate (Requires custom orchestration).	Lightweight (Low computational footprint).

and leveraging developments in underground forums before they turned into active attacks on the surface web.

The next framework *inTIME* (intelligent Threat Intelligence Mining Engine) [13] was the structural evolution of the above-mentioned multi-tier architecture that included automated machine-learning classifiers in the harvesting and data processing loop. The *inTIME* ecosystem was able to go beyond the rigid boundaries of keyword-based filtering, gaining the ability of real-time semantic assessment and automatic categorization of unstructured dark web textual data. As a result, it could transform chaotic stacks of raw data into organized and actionable information.

A representative benchmark in dark web market analysis is the architectural domain introduced by Wegberg et al. [25] that structured the logistical and financial workflows

of anonymous illicit markets. Leveraging machine learning for automated data analysis across a range of dark web marketplaces, Rao et al. [20] proposed a targeted framework based on these analytical models with the use of modern computational techniques. They work on supervised learning algorithms to classify illicit vendor profiles and predict product categories from highly unstructured textual data.

In conclusion, the above models proposed are both important advances in data classification and evidence compilation. However, they tend to rely on static, post-incident batch processing or historical data dumps. Live onion services are transient and highly volatile and this historical dependence is a major operational bottleneck in dealing with this. In contrast, our proposed architecture implements the ideas to move from static batch computation to an active pipeline. Our framework ingests raw dark web noise directly into an optimized Elasticsearch schema and dynamically visualizes indicators via Grafana, turning chaotic hidden network data into actionable knowledge that progresses our fundamental understanding of the Dark Web ecosystem.

Chapter 3

System Design and Installation

The proposed architecture consists of two phases. **Phase I (Filtering & Validation)** functions as a basic pre-processing module that removes unimportant nodes before starting the resource-consuming crawling process. **Phase II (Deep Harvesting)** Utilizes the ACHE framework to perform containerized crawling for content extraction while the system uses NoSQL indexing for backend operations [12]. The main purpose of this architecture involves automated processing of unstructured data which exists on the Tor network, starting from its detection and authentication through to data processing and visualization.

3.1 System Prerequisites and Environment Setup

Before deploying the harvesting pipeline, a secure and isolated software environment must be set up. Because of the risks associated with accessing unverified .onion services, the architecture was not deployed directly on a host machine but rather within a controlled virtualized environment.

3.1.1 Virtualization and Security Considerations

The experimental testbed operated from a personal workstation which used a Type-2 Hypervisor to establish a Virtual Machine (VM). The guest operating system selected for this implementation is **Ubuntu Linux (20.04 LTS)**.

The decision to utilize a Virtual Machine was driven by various security requirements. The process of Dark Web crawling requires users to interact with hidden services which could contain dangerous content including exploit kits and malware. The VM environment provides protection to our system because it keeps any possible security breach limited to the guest operating system which prevents both the host system and network from getting infected. Moreover, virtualization enables users to achieve system integrity through their capability to create system snapshots. In the event of a system failure or security breach during the crawling process, the environment can be instantly reverted to a clean state without data loss or the need for formatting the physical hardware. Finally, the VM enables administrators to establish complete network interface control which directs all network traffic through the Tor interface thus protecting against IP address exposure (deanonymization).

3.1.2 Docker Engine Installation

The Docker Engine serves as the essential base requirement which our architecture needs to function. Docker provides a lightweight virtualization layer that allows the ACHE crawler and Elasticsearch database to run as isolated "containers" sharing the host's OS kernel.

Requirements:

- **Operating System:** Linux (Ubuntu 20.04 LTS recommended) or Windows 10/11 with WSL2 (Windows Subsystem for Linux) enabled.
- **Virtualization:** BIOS-level virtualization support (VT-x/AMD-v) must be enabled to support the Docker daemon.

The environment is initialized by installing the Docker Desktop or Engine[1]. This is critical because ACHE's dependencies (Java Runtime, Tor libraries) are pre-packaged in its Docker image.

```
# Verification of successful installation
$ docker --version
```

[1] https://docs.docker.com/engine/install

```
Docker version 20.10.7, build f0df350
```

Listing 3.1: Verification of successful Docker installation

3.1.3 Additional Dependencies and Libraries

Beyond the Docker infrastructure, the following dependencies need to be installed in order to facilitate the execution of the custom filtering algorithms, manage the project repositories and ensure stable connectivity with the Tor network before the containerized environment is deployed.

The python script operates directly on the host machine. The system needs particular libraries to execute HTTP requests through SOCKS5 and to process HTML data.

```
$ sudo apt install python3 python3-pip -y

$ pip3 install requests pysocks beautifulsoup4
```

A local Tor instance is required for the filtering script to validate URLs independently of the Dockerized Tor proxy used in Phase II.

```
$ sudo apt install tor -y
$ sudo service tor start
```

3.2 System Architecture and Components

The system design implements a distributed modular pipeline which solves two main problems by enabling both anonymous network discovery and handling extensive data collections. It operates through containerization which enables separate execution of individual functional units that maintain high-speed connectivity to other services.

The system architecture, illustrated in Figure 3.1, consists of five components which interact to transform raw network traffic into actionable intelligence.

The Network Component (Tor Proxy) functions as the anonymous gateway to the Dark Web. Because standard DNS can't find .onion addresses, the system uses

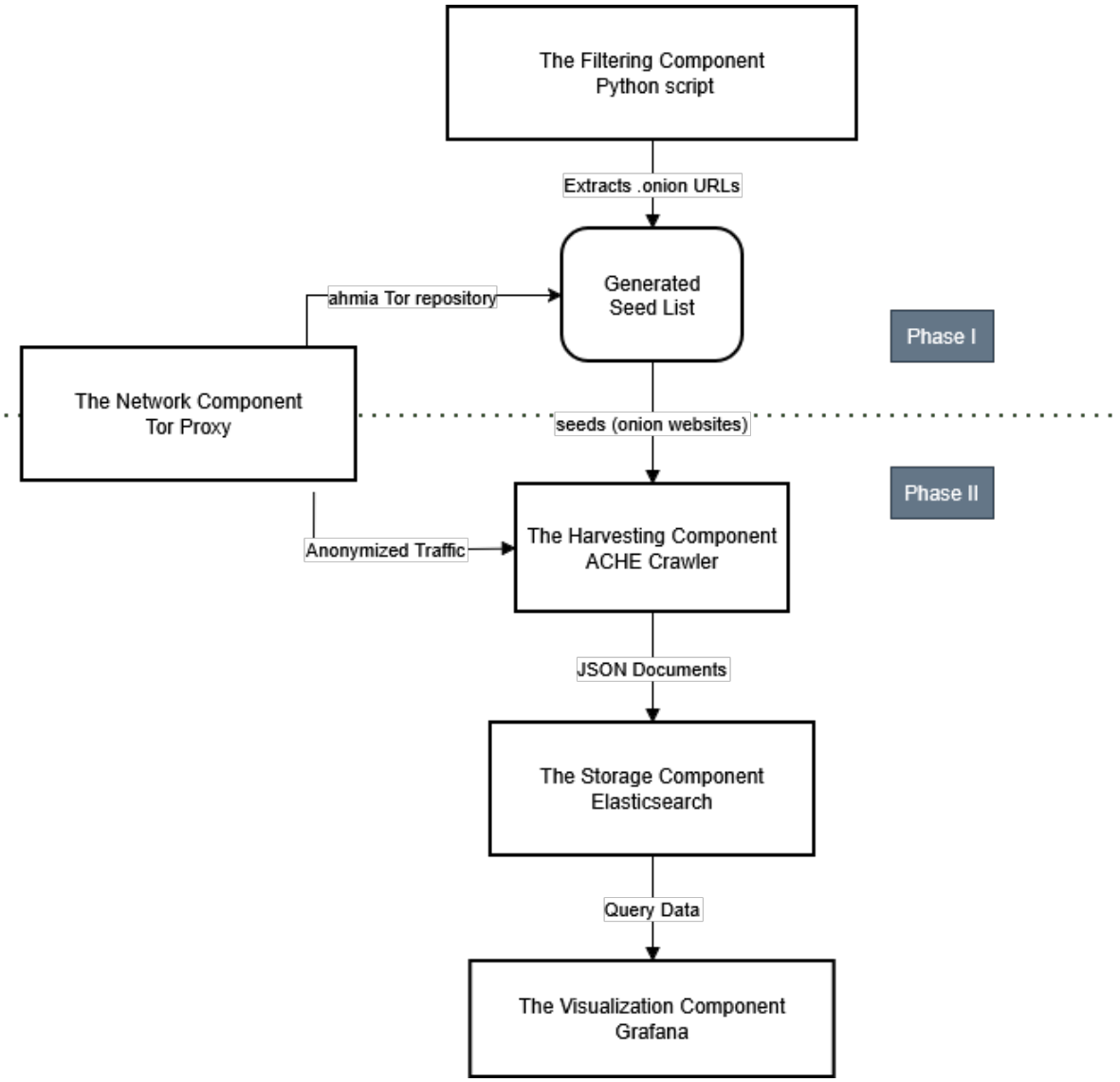


Figure 3.1: System Architecture Diagram

a SOCKS5 proxy interface, which is usually on port 9050. It encrypts traffic in layers and sends it through the Tor network, which keeps the crawler's identity secret.

The Filtering Component (Filtering Script) acts as the first step (Phase I) which executes seed collection and filtering before any crawling process starts. It utilizes a custom Python script to consume massive lists of unverified URLs from public directories of Dark Web. Its role is to validate the liveness of services and filter them based on keywords, producing a high-quality "seed list" to feed the crawler.

The Harvesting Component (ACHE) functions as the core operational section

which makes up the system. The business logic of Focused Crawling operates at this particular layer [2, 15]. The system manages the crawl frontier through Regex filter applications to extract vital information. The system uses the Network Layer to retrieve pages and stores valid results in the Storage Layer.

The Storage Component (Elasticsearch) functions as a distributed NoSQL datastore which operates as the system memory. The system performs immediate indexing operations to index semi-structured JSON documents which originate from the Harvesting Layer. The system enables users to perform complete text searches and execute intricate data combination operations on their massive database of crawled information which exceeds one billion words.

The Visualization Component (Grafana) serves as the user interface. This layer queries the Storage Layer to produce intelligence. The system converts raw data into visual dashboards (see Section 4) which allows analysts to track the ecosystem condition through trend analysis without needing to view the original log data.

3.3 Phase I: Mass Filtering and Seed Generation

The first major component of our methodology addresses the "noise" problem. The directories Ahmia and DarkFail present thousands of `.onion` addresses, yet research indicates that more than 80% of these addresses become unavailable or useless at random points in time. Launching a heavy crawler directly on such a list would result in significant resource wastage. The Python filtering bot, which we built as a custom solution, serves to address this issue.

The bot operates on a list of raw URLs (approximately 23,000 hosts from the Ahmia directory as shown in Figure 3.2). Its execution flow is as follows:

1. The system performs a **Connection Handshake** operation which creates a socket connection to the `.onion` address through the local Tor proxy that operates on port 9050.

2. The system operates with a 10-second timeout period which enables it to detect non-responsive services that will become inactive.

3. The bot performs **Keyword Scanning** after it receives a successful HTTP 200 OK response.

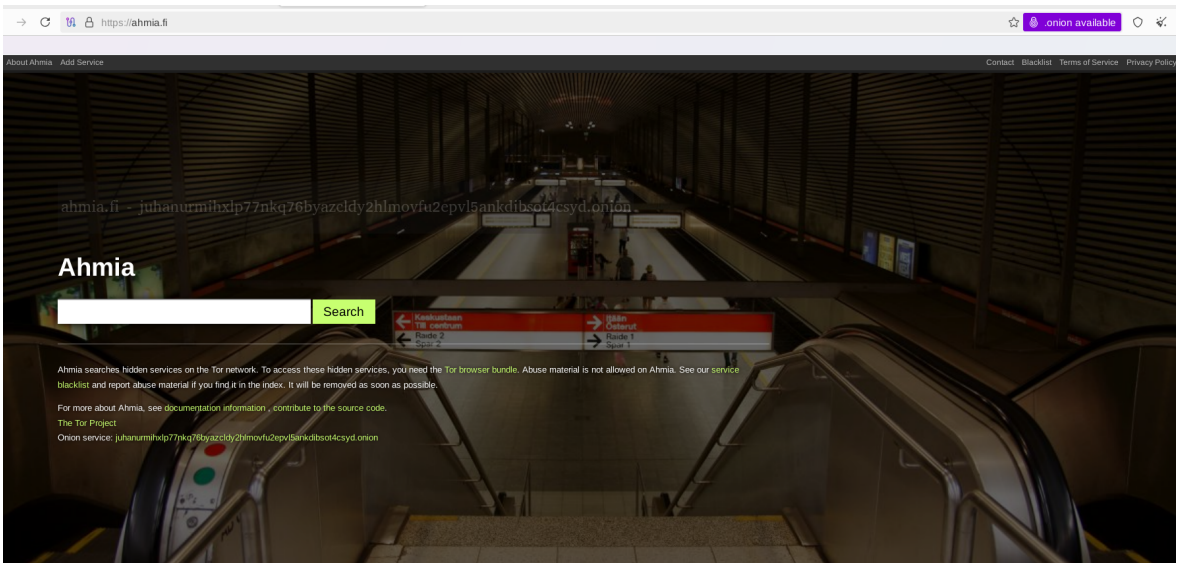


Figure 3.2: Ahmia.fi interface

4. The bot uses HTML parsing to extract research scope defining high-value keywords from the page:

Table 3.1: Categorization example of Keywords used in Phase I Filtering

Category	**Keywords & Patterns**
Financial Assets	Bitcoin, BTC, Monero, XMR, Ethereum, USDT, Wallet, Mixer, Escrow, Laundry, Cash App, PayPal Transfer
Data Leakage	Database, Fullz, SSN, CVV, Dumps, Credentials, Passport Scan, ID Card, Combolist, Email Access, Dox
Security Threats	Exploit, Zero-day, Ransomware, Botnet, Malware, RAT, DDoS, C&C, Rootkit, CVE, Vulnerability

Output: The process decreases the initial number of ∼23,000 raw hosts to a verified list which contained 350 high-relevance URLs. The final set of verified URLs is serialized into a plain text file. This file serves as the primary input (seed frontier) for the ACHE crawler in Phase II. The format adheres to the standard requirement of line-separated Uniform Resource Identifiers (URIs).

As demonstrated in Listing 3.2, the output contains valid Tor V3 addresses (56 characters), which offer stronger cryptography compared to the deprecated V2 addresses [8].

```
http://2222222757uebc4nj4aqtrjxsr3ywf7w6lcks47uvxqdjk5b5h3m7bid.onion
http://2222222wfycju37jin5tcjvisvanageq2uf3udmvwsdxidqcatay4kyd.onion
http://2222224x2424q7mef5g73ynt2c7ihr7kongwhgd7ee4smqm7kkamtcyd.onion
http://22222ejcfuf6sfwrrj7fdqrue5p3o6shdq3nl42jnaurd7obyveamhad.onion
http://22222qszh7l5qdadebpmtaqf2edbdandh2x3wc6ryofv7heplwxfuqd.onion
http://22222tx355brbp6uyue5xty2zcazfmgbcv3ucsxlxr5d66aiwfa4zad.onion
http://22222vtmbmeiqelg362kpbxexy3h6kucj2hxzviqmfjv7l6t6hjuxoad.onion
http://22222fwwemqtxt2vvs2qoxkhom5ad5inaastxbh6gycx7lqxihrl2yqd.onion
http://22224q7whi7onerypu3q3qf6lrrg3kkv2zp56mgxhbrd3t4gk4logwad.onion
http://22225z3rkesk3huhrsvorvvgzprrocipp6if4ksdssrp4vqbxndhycqd.onion
```

Listing 3.2: Sample content of the generated seeds file

3.4 Phase II: Deep Harvesting with ACHE

For the intensive task of deep crawling, we utilized ACHE (Automated Crawler for Hyperlinks and Entities), which was developed at the Visualization, Imaging and Data Analysis Research Center (VIDA)[2] of New York University. ACHE operates as a specialized web crawler which enables link prioritization according to their connection strength to the specified domain. To this end, ACHE provides a full feature set. On the core crawling side, the tool performs regular crawling of a fixed list of websites, recrawls sitemaps to discover new pages and seamlessly crawls hidden services using Tor proxies. Its discovery mechanism is further improved by automatic link prioritization for discovering new relevant websites, based on flexible configuration of different page classifiers including regular expression models. Indexing the crawled pages directly with Elasticsearch also enables data ingestion and exploration with a web interface built to search for content in real-time. Finally, a web-based user interface for real-time monitoring of the crawler are provided to support system oversight.

[2] `https://vida.engineering.nyu.edu/`

3.4.1 ACHE Installation

For the crawling component, we utilize the official ACHE Docker image provided by the developers. As outlined in the ACHE installation guide[3], the use of the Docker image `bitdiscovery/ache` is preferred over compiling from source code, as it guarantees a stable build environment.

```
$ docker run -p 8080:8080 vidanyu/ache:latest
```

Listing 3.3: Running the latest pre-built ACHE image

Upon execution, Docker automatically retrieves the latest layers from the registry and initializes the crawler service.

Alternatively, to enable low-level modifications to the crawler's source code, the image can be built manually from the Git repository[4].

```
$ git clone https://github.com/ViDA-NYU/ache.git
$ cd ache
$ docker build -t ache .
```

Listing 3.4: Building ACHE from source code

The file organization for the `darkweb_crawl` module is illustrated below:

```
darkweb_crawl_project/
|-- config/                    # Configuration files directory
  |-- data/                      # Mapped volume for storing harvested data
    |-- data_target/               # Elasticsearch indices location
  |-- docker-compose.yml         # Orchestration of ACHE, Tor Proxy and ES
  |-- ache.yml                 # Main crawler configuration with Tor enabled
  |-- tor.seeds              # The list of .onion URLs to be crawled
                                   (created in Phase I)
```

[3] https://ache.readthedocs.io/en/latest/install.html
[4] https://github.com/VIDA-NYU/ache

3.4.2 ache.yml

The behavior of ACHE is governed by the `ache.yml` file. Based on the official documentation[5] [15], we tuned specific parameters to optimize for the Tor environment:

3.4.2.1 Tor Integration

Compared to the clear web, the Tor network is naturally slow and unstable [6]. If a standard crawler times out (usually after 30 seconds), it won't be able to get hidden services. To mitigate this, we raise the connection timeouts to 200,000 ms.

```
crawler_manager.downloader.torproxy: http://torproxy:8118
crawler_manager.downloader.tor.socket_timeout: 200000
crawler_manager.downloader.tor.connection_timeout: 20000
crawler_manager.downloader.tor.connection_request_timeout: 20000
```

3.4.2.2 Storage Configuration

To guarantee data safety while enabling real-time analysis, we save data indexed in Elasticsearch and to the local file system (as a backup) [19]:

```
target_storage.data_formats:
  - ELASTICSEARCH
  - FILES
  target_storage.data_format.elasticsearch.rest.hosts:
  - http://elasticsearch:9200
```

3.4.2.3 Content Filtering

Since the objective is to gather textual intelligence, downloading media files wastes bandwidth and storage. We enable filtering to avoid the collection of binary data which would not be useful.

```
crawler_manager.downloader.external_links.filter.enabled: true
crawler_manager.downloader.external_links.filter.mime_types:
  - image/jpeg
```

[5] https://ache.readthedocs.io/en/latest/tutorial-crawling-tor.html

```
  - image/png
  - application/zip
  - application/pdf
  - video/mp4
```

Listing 3.5: Filtering non-textual content

3.4.3 docker-compose.yml

The configuration below highlights two important aspects, command execution and dependency management. The `-e tor` flag instructs the crawler to modify its internal networking stack so that all traffic is routed through the SOCKS5 proxy, while the `depends_on` directive ensures that the database and proxy services are both fully operational before the crawl initiates.

```
ache:
  image: vidanyu/ache
  command: startCrawl -c /config/ -s /config/tor.seeds -o /data -e
   tor
  volumes:
    - ./data-ache/:/data
    - ./:/config
  depends_on:
    - torproxy
    - elasticsearch
```

Listing 3.6: ACHE Service Configuration

3.4.3.1 Database Performance Tuning

Processing large amounts of raw text demands substantial computing power from Elasticsearch [19]. System demands grow when handling extensive unstructured data sets, often leading to "Out of Memory" (OOM) crashes. To reduce that effect, certain kernel adjustments were made:

To mitigate this, we applied specific kernel-level optimizations, which include 4GB

of RAM specifically to the Java Virtual Machine (JVM) via `ES_JAVA_OPTS`. Moreover, the `memlock` limit was disabled (set to -1) in order to prevent the OS from swapping the database memory to the disk, which would degrade the performance.

```
elasticsearch:
  image: docker.elastic.co/elasticsearch/elasticsearch:8.9.0
  environment:
    - discovery.type=single-node
    - bootstrap.memory_lock=true
    - ES_JAVA_OPTS=-Xms4g -Xmx4g
  ulimits:
    memlock:
      soft: -1
      hard: -1
  ports:
    - 9200:9200
```

Listing 3.7: Elasticsearch Resource Allocation

3.4.3.2 Network Gateway

Lastly, a lightweight container connects to the Tor network and listens on port 8118 for HTTP requests from ACHE.

```
torproxy:
  image: dperson/torproxy
  ports:
    - "8118:8118"
    - "9050:9050"
  restart: always
```

Listing 3.8: Tor Proxy Service

3.4.4 Running the Crawler

After the configuration phase, we can run the following command to start the crawling process.

```
docker-compose up -d
```

As depicted in Figure 3.3, the initialization process successfully launces three services that depend on each other. The output from the terminal shows that the Tor Proxy, Elasticsearch and ACHE containers are all in the Running state.

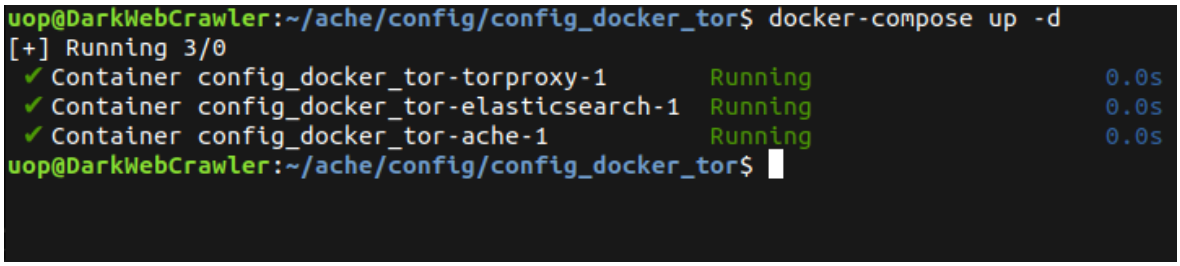


Figure 3.3: Running the crawler

To stop the crawl, we can use the following command-line argument:

```
docker-compose down
```

3.4.5 Elasticsearch

The backend of our architecture is Elasticsearch, a distributed, RESTful search and analytics engine [19]. The system operates as the official database which stores all collected harvested information.

3.4.6 Schema

ACHE automatically structures the crawled web pages into JSON documents. Each document in the `ache-data` index contains the following fields:

- `url`: The full .onion address of the resource.
- `domain`: The root domain, used for aggregation.
- `title`: The HTML `<title>` tag.
- `text`: The main body content, stripped of HTML tags and scripts.

- `retrieved_at`: A timestamp indicating when the crawl occurred.
- `response_headers`: The HTTP headers returned by the server.

We selected Elasticsearch over relational databases (like MySQL) for three main reasons. Firstly, the system employs an inverted index which enables users to search for keywords throughout massive text databases at high speeds. It also processes web data with its unpredictable structure through flexible table definitions. Finally, Elasticsearch's Aggregation Framework allows users to perform "Group By" operations through its query execution system.

3.4.7 Index Configuration and Mapping

Data is organized into a single index named `darkweb_crawler`. To optimize query performance and aggregation speed, we defined a strict **Index Mapping** (Schema):

```
PUT /darkweb_crawler
{
  "mappings": {
    "properties": {
      "domain": { "type": "keyword" },
      "url": { "type": "keyword" },
      "title": { "type": "text", "analyzer": "standard" },
      "html_content": { "type": "text", "index": false },
      "cleaned_text": { "type": "text", "analyzer": "english" },
      "timestamp": { "type": "date" },
      "content_hash": { "type": "keyword" },
      "detected_entities": {
        "type": "nested",
        "properties": {
            "type": { "type": "keyword" },
            "value": { "type": "keyword" }
        }
      }
    }
  }
```

```
}
```

Listing 3.9: Elasticsearch Index Mapping for Dark Web Data

Index mapping configuration is an important architectural layer to guarantee data integrity and reduce storage overheads. Rather than relying on the engine's default dynamic type inference, which often leads to indexing memory constraints, each field in the `darkweb_crawler` index is mapped to its particular operational role. The `domain` and `url` fields are mapped as non-tokenized `keyword` types to enable high-speed terms aggregations and filtering in the Grafana interface without the overhead of full-text analysis. The `title` field is mapped as type `text` with the analyzer `standard` to allow partial-match queries. The heavy `html_content` field is set to `index: false` for performance reasons. This setting stores the raw source code permanently for forensic verification but completely skips inverted index generation to avoid structural bloat (e.g. skipping the indexing of thousands of redundant `<div>` tags and embedded scripts). The parsed body text in `cleaned_text` is processed with an explicit `english` analyzer to enforce morphological stemming and stop-words removal (e.g., striping common words like "the" or "and" and reducing terms like "hacking", "hacked" and "hacks" to their common root "hack"), greatly improving keyword intelligence queries across the dominant language of the dataset. The `timestamp` and `content_hash` fields also serve as the required temporal anchors for time-series dashboards.

Finally, the `detected_entities` object block is explicitly defined as a `nested` type, a structural requirement that preserves the strict internal relationship between an indicator's classification `type` (e.g. Crypto Wallet, Email) and its exact string `value`, which is populated by the ingestion pipeline that utilizes a set of targeted Regular Expressions (Regex). These patterns scan the cleaned text field to systematically extract important indicators, such as cryptocurrency wallet addresses and threat identifiers.

Specifically, the pipeline applies predefined patterns such as the ones shown below:

- **Ethereum / ERC-20 Wallets:** Matched using `\b0x[a-fA-F0-9]{40}\b`, enabling the aggregation of USDT/USDC transactions discussed in Section 4.5.
- **Bitcoin Wallets:** Identified via `\b(1|3|bc1)[a-zA-HJ-NP-Z0-9]{25,39}\b`.

- **Monero Wallets:** Filtered using `\b4[0-9AB][1-9A-HJ-NP-Za-km-z]{93}\b`.
- **Vulnerability Identifiers (CVEs):** Extracted via `\bCVE-\d{4}-\d{4,7}\b` to feed the Threat Intelligence Feed.

The extracted enable high-speed bucket aggregations in Elasticsearch that are then seamlessly rendered in Grafana dashboards.

3.4.8 Grafana

The final stage of the deployment involves configuring the dashboard interface to visualize the collected data. The analytical frontend of Grafana[6] uses Elasticsearch indices to produce real-time metrics through its query functionality. Once the Docker containers are operational, the Grafana web interface is accessible through port 3000 on the host machine via the link http://localhost:3000. By default, Grafana is not connected to any database. To enable visualization, the Elasticsearch container must be added as a trusted **Data Source**.

The most important step in this configuration requires the analyst to define the `Time Field Name` `retrieved_at`. ACHE automatically timestamps every crawled page with this field. The system needs this exact timestamp format to display time-based graphs in Grafana which shows the page harvesting speed in real-time through "Crawled Pages per Minute" metrics.

[6] `https://grafana.com/`

Chapter 4

Data Analysis and Visualization

This chapter presents the analytical findings derived from the deployment of the proposed harvesting architecture. Having established a robust pipeline for data collection in Chapter 3, we now shift our focus from the mechanisms of collection to the interpretation of the collected data.

The analysis is structured around four key pillars, corresponding to the visualizations implemented in the Grafana dashboard. First, infrastructural forensics investigates the specific technologies utilized to support the underground economy. Second, linguistic profiling examines who the actors are and their geopolitical origins. Third, the crypto-economy pillar evaluates money flows and shifting financial preferences [14]. Finally, threat analysis provides an assessment of what is actually being transacted beyond surface-level perception.

All charts and tables presented here can be generated via the Grafana dashboard, which queries data directly from the Elasticsearch index.

4.1 Dataset Overview

Before delving into specific trends, it is essential to quantify the volume of the analyzed dataset. The data, stored as semi-structured JSON documents, represents a snapshot of the active Tor network.

The Phase I (Filtering) process proved its value: Starting with roughly 23,000 basic

web entries pulled from Ahmia's directory, we managed to pinpoint around **350 key seeds.** If the ACHE tool had proceeded through all 23,000 originals without pause, that probably would not have worked and unrelated material would have drowned out real insights. More than **52,000 distinct pages** were handled by the system, revealing around **1,200 unique domains**, including sub-domains and hidden paths.

4.2 Operational Metrics

Before interpreting the content, we need to study the crawl behavior. The Real-Time Crawl Status panel provides a window into the structural integrity of the Dark Web during our harvesting process.

4.2.1 Throughput and Latency Correlation

The graph in Figure 4.1 displays **Page Crawling Rates** which were monitored over a multi-day observation period.

Figure 4.1: Pages Crawled over a multi-day observation period.

Both system-level and operational execution elements contribute to this rate's dynamic character. Active crawling sessions when the agent processed high-density targets, including multi-threaded forum archives, are indicated by the prominent green peaks (reaching more than 500 pages each time bucket). On the other hand, the extended zero-throughput intervals show times when the crawler completed its allocated seed queue and was momentarily stopped before being restarted with updated seed batches. Additionally, Tor's circuit rotation mechanism, where route assignment to slower or overcrowded exit relays increased response latencies beyond 5000ms [6], as

well as hidden service unavailability were the main causes of short-term rate swings during active execution windows.

4.2.2 Top Domains Analysis

To understand the structural topology of the harvested dataset, we analyzed the distribution of pages per unique `.onion` domain. As illustrated in the "Top 15 Biggest Onion Websites" bar chart (Figure 4.2), the page distribution is heavily skewed toward a small set of highly active hidden services, a common structural trait in web graphs.

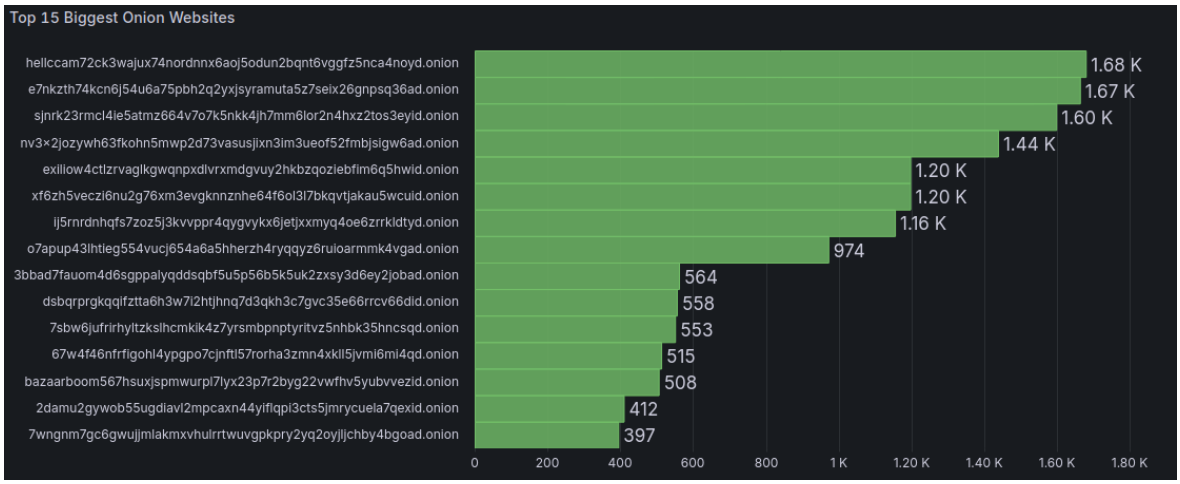


Figure 4.2: Top 15 Biggest Onion Websites by Page Count

The chart reveals that the harvested dataset is highly concentrated within a small number of domains, which account for the vast majority of total collected pages. The top three identified services each yielded approximately **1.60K to 1.68K unique pages**.

Regarding the crawler depth validation, the large number of pages which the system retrieved from individual hosts during the Top 15 hosts validation process proved the effectiveness of the Phase II (Deep Harvesting) methodology. The ACHE crawler demonstrated its capability to retrieve content which exists past basic web pages because it navigated through complex internal directory structures to obtain Deep Web information.

4.3 Infrastructural Forensics

The Panel depicted in Figure 4.3 aggregates the HTTP Server headers returned by the crawled services. The system enables us to evaluate the operational configurations maintained by hidden service administrators. [23].

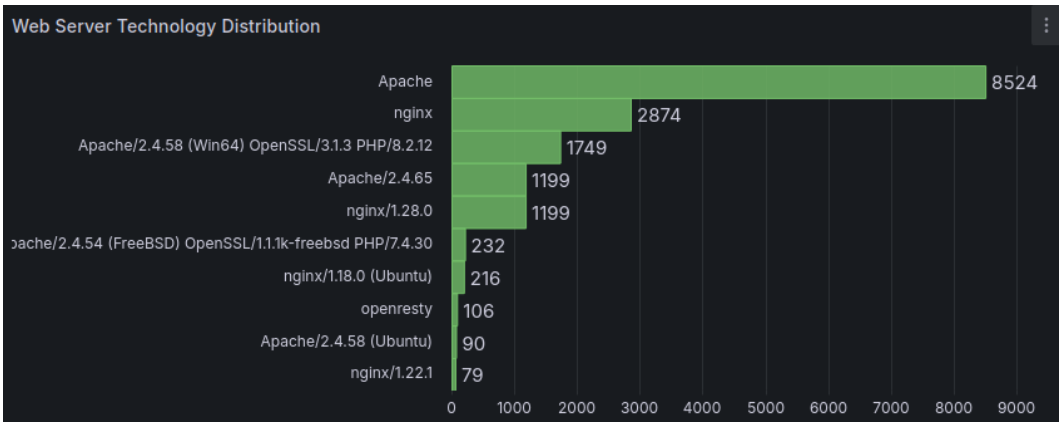


Figure 4.3: Web Server Technology Distribution

As illustrated, the web server landscape is heavily dominated by various deployments of **Apache**, along with instances of **Nginx**. Additionally, a smaller subset utilizes `OpenResty`, a Web platform usually used by administrators to implement DDoS mitigation logic, custom reverse proxying and dynamic routing.

Critical system telemetry, such as software versions, underlying programming language runtimes (e.g., PHP 8.2.12 and PHP 7.4.30) and target operating system architectures, are revealed when HTTP headers are explicitly disclosed. In addition to Unix-like systems like `FreeBSD` and Linux versions like `Ubuntu`, the dataset also shows a notable presence of `Microsoft Windows` environments.

This extensive leakage of server banners suggests that dark web operators frequently neglect to take basic server hardening precautions. Law enforcement and threat intelligence analyst can benefit from network reconnaissance, as this enables possible CVE matching and server fingerprinting.

4.4 Demographic and Linguistic Profiling

One of the primary challenges in Dark Web intelligence is Attribution - determining who is operating a service and their geographical locations. The process of geolocating Onion services through standard network measurement becomes impossible because it lacks the ability to determine physical locations (country, city, ISP) of servers. Consequently, Linguistic Fingerprinting becomes the closest viable proxy for attribution.

4.4.1 Linguistic Distribution of Harvested Services

By analyzing UTF-8 character encodings, HTML language attributes and textual vocabulary within the harvested web pages, we can gain a clearer picture of the demographic structure within the underlying ecosystem.

Figure 4.4 shows that English is the most common language, but it also shows how people in different parts of the world use their own languages.

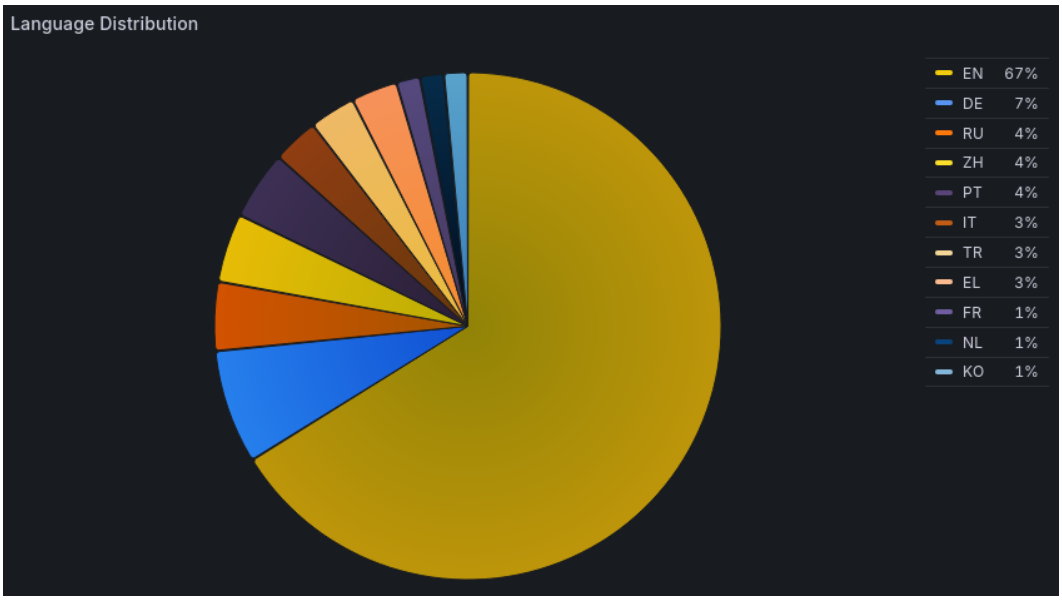


Figure 4.4: Linguistic Distribution of Harvested Services

The data shows **English** language stands as the leading language because it represents two-thirds of the complete ecosystem. This does not necessarily imply that **67%** of criminals are native English speakers. Rather, it proves that English is the *lingua franca* of the underground economy [16]. To achieve liquidity in illicit markets (i.e.,

to find buyers for stolen data or drugs), vendors from all nationalities are forced to operate in English. The language serves as the global trade connection which connects all businesses across the world.

German stands as the second most common language used in the harvested dark web sites. The **7%** user base maintains a small yet highly active localized community. Surprisingly, **Russian, Chinese and Portuguese** share a percentage (**4% each**) in our harvested dataset. The remaining percentage consists of various languages, indicating that while cybercrime is global, the infrastructure is concentrated in a few key linguistic hubs.

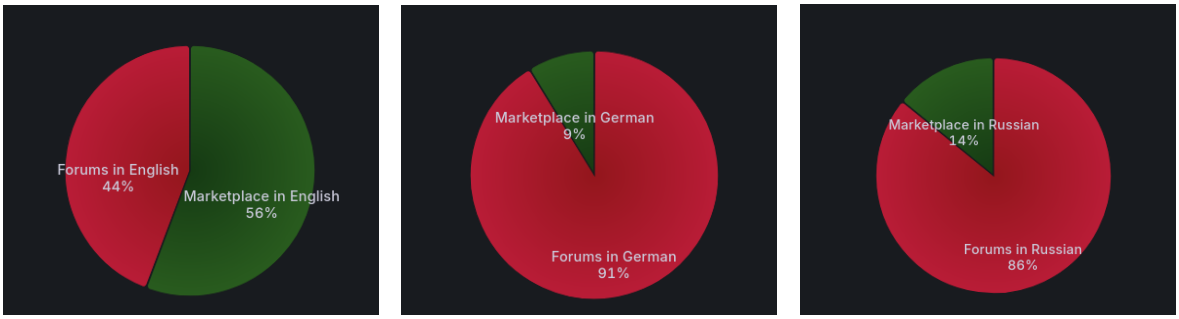
4.4.2 Ecosystem structure across the three dominant languages

In an effort to broaden the linguistic and spatial research and enhance comprehension of particular regional traits, comprehensive visualization models were created, as illustrated in Figure 4.5, for the three predominant languages identified in the research above.

As illustrated, the **English-speaking** ecosystem is fairly balanced but leans distinctively towards commercialization. This is because 56% of the services identified serve as Marketplaces and 44% serve as Forums. This supports the hypothesis that English language is the **global "storefront"** of the Dark Web. Vendors use English as a common language in a Marketplace to reach as many international customers as possible, no matter where they actually come from.

The German and Russian ecosystems, on the other hand, strongly favor social interaction and discussion over direct commerce [10, 26]. The **German-speaking** ecosystem is mainly comprised of Forums (91%). Similarly, forums make up 86% of the **Russian-language** landscape, while automated Marketplaces account for 14%. This indicates that the German-speaking and Russian-speaking Dark Web is more of a place for people to share ideas, protect their privacy and talk about technical issues (like Fraud/Hacking boards) in their own language than a place to buy illegal goods.

Finally, it is important to emphasize that this linguistic breakdown shows the interface language of the hidden services, not the operators' physical location or nationality.



English Ecosystem | German Ecosystem | Russian Ecosystem

Figure 4.5: Analysis of ecosystem structure (Marketplace vs. Forum) across dominant languages.

As previously discussed, English functions as the lingua franca of the cybercrime economy [16]. As a result, many English-speaking marketplaces are actually managed by non-native actors who use English to expand their market reach and to share ideas and information on English-speaking websites, thereby concealing their true demographic origin behind a globalized façade.

4.5 The Crypto-Economy

The Dark Web ecosystem depends on financial transactions as its essential operational foundation. Unlike surface web e-commerce, which relies on regulated payment gateways, the Tor network relies exclusively on cryptocurrencies [14].

Figure 4.6 monitors the occurrences of cryptocurrency wallet addresses through Regex pattern matching. The visualization functions as an economic indicator which tracks the size of the hidden economic sector.

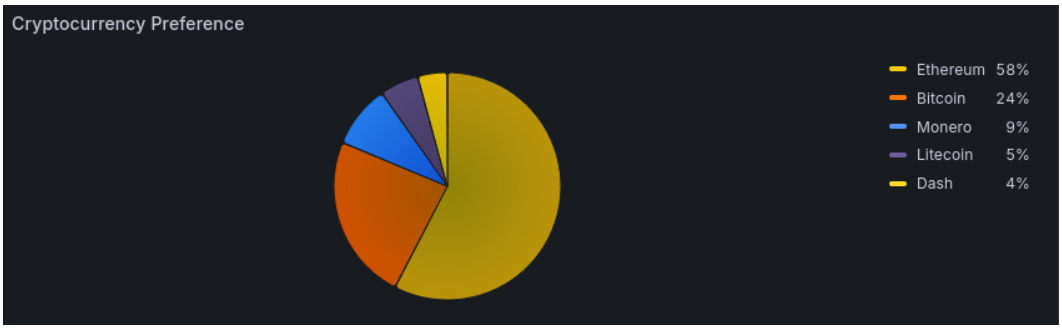


Figure 4.6: Cryptocurrency Wallet Distribution

The most striking finding is the dominance of **Ethereum (58%)**, which has dethroned Bitcoin. However, a deeper technical analysis of the addresses (all beginning with `0x...`) suggests that this does not primarily represent transactions in Ether currency, but rather in **ERC-20 Tokens**, specifically **Tether (USDT)** and **USD Coin (USDC)**. Dark Web vendors operate their businesses with small profit margins. The extreme volatility of Bitcoin (which can fluctuate 10% in a day) is a business risk. The criminals achieved price stability through their switch to USDT which operates on Ethereum while keeping their operations free from censorship.

Bitcoin (BTC) used to be the only cryptocurrency on the Dark Web, but now it only makes up **24%** of all detected addresses. Bitcoin's public blockchain feature called "Traceability" and its price swings make it less useful for big businesses that want to use the cryptocurrency [21]. It is mostly used as a way for new buyers, who don't know how to buy Altcoins yet, to get into the market.

Monero (XMR) maintains a stable market position which accounts for **9%** of the total market value. The XMR market has less trading activity than other markets, but it is the most secure market in the economy. The content correlation analysis shows that Monero is the main cryptocurrency for High-Risk Services that use Malware-as-a-Service and Contract Hacking [23]. The ETH/USDT pair is used for "Business Stability," while Monero is used for operational security, which helps businesses hide their activities.

The presence of **Litecoin and Dash** represents the "budget" sector. These currencies are the best choices for small purchases, because the fees for Ethereum and Bitcoin transactions would be higher than the value of the item.

4.6 Threat Landscape Analysis

Having analyzed the actors (Language) and the currency (Crypto), this final analytical section focuses on the actual threats and commodities exchanged within the ecosystem [24]. The Vulnerability Distribution Panel and Real-Time Threat Feed in Grafana allowed us to create a map which showed the operational abilities of the adversaries.

4.6.1 Vulnerability and Tool Distribution

Figure 4.7 classifies the extracted textual artifacts and threat intelligence results according to the particular attack tools and vulnerability types that are mentioned throughout the crawled hidden services.

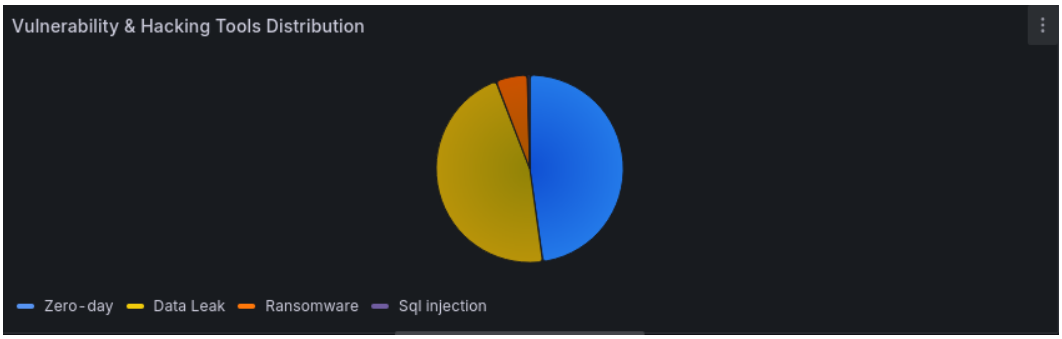


Figure 4.7: Vulnerability and Hacking Tool Distribution

The majority of content is driven by listings under **Zero-day attacks** and **Data Leaks, Dumps and Fullz (Full Identity Packages)**. The steady monetization of identity packages and corporate database leaks highlights a sophisticated underground supply chain where stolen data is quickly aggregated, verified and resold for financial fraud [17]. However, a deeper qualitative analysis indicates a significant prevalence of fraud within the public "zero-day" listings. Many of them are either scams aimed at unsuspecting buyers or recycled exploits for older vulnerabilities. Genuine Zero-day markets are likely insulated in invite-only forums and are not visible in public crawls.

The dangerous threats of **Ransomware** [17] also appear in the system. Our text analysis indicates that Ransomware is sold as a standalone binary. The last tier contains Tools which include **SQL Injection** (SQLi), appearing with an extremely low percentage that is almost invisible in the distribution diagram. This minimal presence does not mean that SQL injection exploits aren't found in the underground economy, but that such highly targeted tools and databases are much more likely to be isolated in private, invite-only forums.

4.6.2 The Operational Threat Intelligence Feed

While the previous charts offer strategic insight (long-term trends), the Threat Intelligence Feed (Figure 4.8) provides tactical value.

Threat Intelligence Feed (Raw Critical Findings)

retrieved	domain	text	title	topPrivateDomain	url
2025-12-03 03:14:	e7nkzth74kcn6j54u6	Publié le 21 juin 201	Le capitalisme en rés	e7nkzth74kcn6j54u6	http://e7nk
2025-12-03 03:14:	xf6zh5veczi6nu2g76	Вам необходимо об	Пробив фнс	xf6zh5veczi6nu2g76	http://xf6zl
2025-12-03 03:14:	e7nkzth74kcn6j54u6	Publié le 10 février 2	Blocage du collège T	e7nkzth74kcn6j54u6	http://e7nk
2025-12-03 03:14:	e7nkzth74kcn6j54u6	Publié le 15 novembr	AG étudiante contre l	e7nkzth74kcn6j54u6	http://e7nk

Figure 4.8: Real-Time Threat Intelligence Feed

The live monitoring interface shows a full stream of extracted telemetry, including item titles, source `.onion` URLs, website content, timestamps and extracted entity indicators. One of the operational strengths of this panel is the dynamic search and filter capability. Threat analysts can interactively query the ingested feed with targeted keywords, domain names or threat vectors to monitor emerging risks. This interactive functionality bridges the gap between passive harvesting and active Security Operations Center (SOC) workflows.

From a Brand Protection and Asset Monitoring perspective, analysts can apply persistent filters for corporate identifiers, proprietary project names, or sensitive keywords such as `"Company X Database"` or corporate email domains. This allows real-time detection of exfiltrated corporate data, breach logs or leaks of employee credentials before they escalate or widely circulate.

Alternatively, in a Vulnerability Management context, security operations can isolate specific vulnerability identifiers e.g., searching directly for `"CVE-2021-44228"` to follow market activity around specific exploits. Analysts see an unexpected spike in filtered listings as an indication that a Proof-of-Concept (PoC) exploit or weaponized execution script is starting to propagate across underground channels and that patch management prioritization needs to be addressed immediately.

Chapter 5

Conclusions and Future Work

This concluding chapter combines our main findings with suggestions on how the system can be further enhanced by leveraging new technologies.

5.1 Summary of Contributions

In this work, we presented a system for harvesting and analyzing the Dark Web ecosystem. We developed a focused data collection system which combined Tor proxies with the ACHE crawler to overcome the restrictions of manual research and basic web scraping methods, while Elasticsearch and Grafana were used in order for the visualization environment to be implemented. This system converts raw hidden service entities into actionable threat telemetry by fusing heterogeneous data streams into interactive, customizable visual elements, allowing analysts to visually inspect complex dark web environments, trace behaviors and quickly identify illicit activities and patterns.

The research shows that the underground economy operates as a professional system which uses technical methods to stay secure while its financial operations continue to develop, demonstrating that continuous, automated monitoring is essential for understanding and mitigating modern cyber threats.

5.2 Limitations of the Research

While the proposed architecture successfully met its design goals, certain limitations were identified during the experimental phase. The ACHE crawler, configured as an unauthenticated agent, could not penetrate “gated" communities. Users need to provide login information and either invite codes or paid subscription fees to access high-end forums and marketplaces. Consequently, our dataset likely under-represents the most advanced tier of cybercrime [10, 23]. Moreover, modern onion services use complex CAPTCHA challenges which operate as automated systems to prevent scrapers from accessing their content. The system solved its basic problems but advanced image-based CAPTCHAs stopped ACHE from performing successful crawls on specific high-value targets.

5.3 Future Work

To address these limitations and further enhance Dark Web data extraction capabilities, the following directions are proposed for future research:

5.3.1 Authentication Modules and Persona Management

A promising future work direction lies in exploring the integration of authentication modules to handle persistent session cookies. By potentially leveraging a managed pool of virtual identities, the system could get into the "Deep Dark Web". Such an approach would probably need human-behavior simulation, such as randomized cursor movements and dynamic keystroke latencies, in order to defeat heuristic bot detection mechanisms [17, 23]. Further research into automated Two-Factor Authentication (2FA) token handling could also be a viable method of gaining access to restricted forums.

5.3.2 AI-Driven CAPTCHA Solving

Given the rapid surge and evolution of AI technologies, it is becoming a necessity to embed intelligent solving mechanisms into the harvesting pipeline. Traditional approaches often do not work against modern security controls, and the use of sophisticated AI capabilities could help the system to navigate complex visual CAPTCHA and dynamic challenge logic. Since basic Optical Character Recognition (OCR) is insufficient nowadays, newer methods trained on fake images made by Generative Adversarial Networks (GANs) may slip past visual CAPTCHAs [7].

5.3.3 Cross-Platform Intelligence

The cybercrime ecosystem is becoming more and more fragmented, moving beyond the Tor network. The future architecture needs to have a multi-platform system that lets harvesters get data from Telegram channels [22] and alternative darknet protocols. Criminals are moving to these platforms to avoid police attention related to Tor and take advantage of the end-to-end encryption features of modern messaging apps. In order for a unified ingestion pipeline to work, there would need to be separate protocol handlers that could connect to the MTProto API[1] for Telegram and the I2P router console[2] to combine data from these different sources into one data lake.

5.3.4 Blockchain Transaction Correlation

Since the current research identifies cryptocurrency wallets found on Hidden Services, a logical extension would be the integration of a Blockchain Analytics module [21]. This would allow researchers to follow the money trail by linking the scraped Bitcoin addresses to transactions on the public ledger. Using clustering heuristics like common-input analysis, it would be possible to figure out how money moves from illegal marketplaces to exchanges [14]. This could provide the capability to figure out the actual revenue of the criminal enterprises being watched and to explore potential cash-out points.

[1] `https://core.telegram.org/mtproto`
[2] `https://geti2p.net/`

Bibliography

[1] M. W. Al-Nabki, E. Fidalgo, E. Alegre, and I. de Paz. Tooldog: A Dedicated Dark Web Crawler and Dataset Generator. In *Proceedings of the 14th International Conference on Soft Computing Models in Industrial and Environmental Applications (SOCO 2019)*, pages 255–265. Springer, 2019.

[2] L. Barbosa, K. Pham, C. Silva, M.R. Vieira, and J. Freire. Structured Open Web Data Harvesting. In *Proceedings of the 2016 International Conference on Management of Data (SIGMOD)*, pages 2169–2172, 2016.

[3] Jesper Bergman and Oliver B. Popov. Exploring Dark Web Crawlers: A Systematic Literature Review of Dark Web Crawlers and Their Implementation. *IEEE Access*, 11:35914–35933, 2023.

[4] A. Biryukov, I. Pustogarov, and R.P. Weinmann. Content and Popularity Analysis of Tor Hidden Services. In *Proceedings of the 2014 IEEE 34th International Conference on Distributed Computing Systems Workshops*, pages 188–193, 2014.

[5] P. Boldi and S. Vigna. The WebGraph Framework I: Compression Techniques. In *Proceedings of the 13th International Conference on World Wide Web*, pages 595–602, 2004.

[6] A. Chaabane, P. Manils, and M.A. Kaafar. Digging into Anonymous Traffic: A Deep Analysis of the Tor Anonymizing Network. In *Proceedings of the 2010 4th International Conference on Network and System Security*, pages 167–174, 2010.

[7] C. Connolly, D. Wall, and R. Chaudhary. Automated CAPTCHA Solving in

Dark Web Investigations: A Forensic Perspective. *Forensic Science International: Digital Investigation*, 44:301–312, 2023.

[8] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The Second-Generation Onion Router. In *Proceedings of the 13th USENIX Security Symposium*, pages 303–320, 2004.

[9] C. Guitton. A Review of the Available Content on Tor Hidden Services: The Case against Minors. *Aggression and Violent Behavior*, 18(4):353–364, 2013.

[10] R. Hoheisel, T. Meurs, M. Junger, E. Tews, and A. Abhishta. Dark Web Dialogues: Analyzing Communication Platform Choices of Underground Forum Users. In *2024 APWG Symposium on Electronic Crime Research (eCrime)*, pages 1–15. IEEE, 2024.

[11] E. Jardine. The Dark Web Dilemma: Tor, Anonymity and Online Crime. *Global Commission on Internet Governance Paper Series*, (21), 2015.

[12] P. Koloveas, T. Chantzios, C. Tryfonopoulos, and S. Skiadopoulos. A Crawler Architecture for Harvesting the Clear, Social, and Dark Web for IoT-related Cyberthreat Intelligence. In *2019 IEEE World Congress on Services (SERVICES)*, pages 3–8, 2019.

[13] Paris Koloveas, Thanasis Chantzios, Sofia Alevizopoulou, Spiros Skiadopoulos, and Christos Tryfonopoulos. inTIME: A Machine Learning-Based Framework for Gathering and Leveraging Web Data to Cyber-Threat Intelligence. *Electronics*, 10(7):818, 2021.

[14] M. Moser, R. Bohme, and D. Breuker. An Inquiry into Money Laundering Tools in the Bitcoin Ecosystem. In *Proceedings of the eCrime Researchers Summit (eCrime)*, pages 1–14, 2013.

[15] New York University. *ACHE Documentation: Focused Web Crawler*, 2024. Accessed: 2026-01-24.

[16] Fawn T. Ngo, Catherine D. Marcum, and Scott Belshaw. The dark web: What is it, how to access it, and why we need to study it. *Journal of Contemporary Criminal Justice*, 39(2):160–166, 2023.

[17] E. Nunes, N. Diab, A. Gunn, E. Marin, V. Mishra, V. Paliath, J. Robertson, J. Shakarian, A. Thart, and P. Shakarian. Darknet and Deepnet Mining for Proactive Cybersecurity Threat Intelligence. In *Proceedings of the 2016 IEEE Conference on Intelligence and Security Informatics (ISI)*, pages 7–12, 2016.

[18] C. Olston and M. Najork. Web Crawling. *Foundations and Trends in Information Retrieval*, 4(3):175–246, 2010.

[19] P. De Pascale, I. Amerini, and F. Nencini. CRATOR: A Modular Dark Web Crawler for Dynamic Marketplaces. *Future Internet*, 16(2), 2024.

[20] U. N. Pratham Rao, Rachit Raam Guruprasad, Omkaar J. Shetty, V. Sarasvathi, and Gauri Sameer Rapate. Data Analysis of Dark Web Marketplaces using Machine Learning. In *Proceedings of the International Conference on Signal Processing, Information and Communication Systems (ICSPIS)*, pages 1–6, 2024.

[21] F. Reid and M. Harrigan. An Analysis of Anonymity in the Bitcoin System. In *Security and Privacy in Social Networks*, pages 197–223. Springer, 2013.

[22] R. Ricaldi, T. Marjanov, and A. Hutchings. Uncovering the Trust Signals Supporting Telegram's Cybercrime Economy. In *Proceedings of the 2025 APWG Symposium on Electronic Crime Research (eCrime)*. IEEE, 2025.

[23] S. Samtani, R. Yu, H. Zhu, M. Patton, and H. Chen. Exploring Hacker Assets in Underground Forums: Identifying Key Hackers and Asset Transfer Networks. *IEEE Transactions on Information Forensics and Security*, 12(6):1337–1352, 2017.

[24] M. Schäfer, M. Fuchs, M. Strohmeier, M. Engel, M. Liechti, and V. Lenders. BlackWidow: Monitoring the Dark Web for Credit Card Fraud. In *2019 11th International Conference on Cyber Conflict (CyCon)*, pages 1–20. IEEE, 2019.

[25] Rolf van Wegberg, Rianne Verburgh, Gijs de Jong, Matti Obbink, and Michel van Eeten. How cybercriminals outsource work: an empirical analysis of illicit commodities on darknet marketplaces. *Empirical Software Engineering*, 23(4):2210–2235, 2018.

[26] A. V. Vu, J. Laite, A. Caines, and A. Hutchings. POSTCOG: A Tool for Interdisciplinary Research into Underground Forums at Scale. In *Proceedings of the 2022 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 402–409. IEEE, 2022.

[27] P. Practical Y. Yannikos, J. Heeger Navas and E. Vasilomanolakis. Prophet: Forecasting the Lifespan of Hidden Services. In *Proceedings of the 2022 IEEE International Conference on Cyber Security and Resilience (CSR)*, pages 345–352, 2022.