iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** XI **Month of publication:** November 2025

DOI: https://doi.org/10.22214/ijraset.2025.75244

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue XI Nov 2025- Available at www.ijraset.com

Implementation of DarkReaper Automated OSINT Tool

Viraj Kadam[1], Prof. J. R. Mahajan[2]
[1]*Student,* [2]*Asst. Prof. Department of Computer Engineering*
[1, 2]*Adsul's Technical Campus, Chas, Ahilyanagar, Maharashtra, India*
Savitribai Phule Pune University, Pune, India

Abstract*: *The paper presents the implementation of an automated Open Source Intelligence (OSINT) tool named DarkReaper . It focuses on modular OSINT automation using Python for gathering intelligence from surface and dark web sources. The implementation includes modules for phone numbers, email address, image,ip address and username lookups. Each module performs multi-source data extraction, analysis, and stores structured JSON outputs. The tool demonstrates high efficiency, modularity, and extensibility for cybersecurity investigation tasks.
Keywords: Python, Automation, OSINT, Cybersecurity, Reconnaissance, Intelligence

I. INTRODUCTION

Current Open-Source Intelligence (OSINT) tools present significant accessibility challenges, often requiring premium subscriptions or complex API configurations that limit their utility for students and professionals with constrained resources. While comprehensive platforms like SpiderFoot offer extensive capabilities, their cost and complexity create barriers to entry, particularly for educational use.

Furthermore, the limited integration of dark web sources in most available tools results in fragmented workflows and incomplete intelligence gathering.

This paper details the implementation of DarkReaper, an automated OSINT framework that overcomes these limitations through a unified command-line interface and modular architecture. By leveraging Python automation, web scraping techniques, and free APIs, DarkReaper delivers comprehensive surface and dark web intelligence capabilities without financial barriers. The system's structured JSON output and ethical data collection approach make professional-grade OSINT accessible to cybersecurity practitioners, students, and researchers alike

II. LITERATURE SURVEY

The domain of Open-Source Intelligence (OSINT) has undergone significant transformation, driven by developments in automated frameworks, unified platforms, and specialized investigative methodologies. This review examines the current landscape to identify both technological advances and persistent limitations that inform DarkReaper's implementation approach.

A. Established OSINT Frameworks

The OSINT ecosystem includes several mature platforms designed for broad reconnaissance operations. SpiderFoot [1] represents a widely-referenced example, providing modular access to numerous data sources commonly utilized in security assessments. However, as highlighted by Nordin, Yusoff, and Ahmad [6], such frameworks often depend heavily on API keys and web services, creating accessibility challenges for educational users and practitioners with limited resources.

Pastor-Galindo et al. [2] further characterized OSINT as an underutilized resource whose potential remains constrained by issues of data reliability, source instability, and methodological fragmentation—challenges that current tools struggle to adequately address.

B. Integrated Automation Platforms

Recent research has emphasized the creation of consolidated OSINT platforms that streamline investigative workflows. Rafaila et al. [7] introduced MTAFinder, which combines disparate data sources through a unified interface to enhance operational efficiency, though noting difficulties in managing source heterogeneity. Similarly, Shin and Jung [8] proposed a maintainable OSINT framework prioritizing modular design, task queuing, and caching mechanisms—architectural principles that directly informed DarkReaper's development.

©IJRASET: All Rights are Reserved

The movement toward automation now increasingly incorporates artificial intelligence. Browne, Chen, and Lavoie [5] conducted a systematic assessment of AI-driven OSINT automation, identifying advantages in entity resolution and data clustering, while cautioning against algorithmic bias and interpretability challenges. These concerns motivated DarkReaper's approach to transparent, post-processed AI analysis.

Complementary research by Huang et al. [9] further demonstrates how OSINT can be leveraged for malicious behavior discovery and interpretation, reinforcing the value of automated intelligence processing in security contexts.

C. Specialized Intelligence Domains

OSINT research has also advanced in specialized investigative areas. For dark web intelligence gathering, Gopireddy [3] analyzed methods for crawling hidden services and extracting threat indicators, while Vignesh and Patidar [4] explored contemporary techniques for investigating exposed data on dark web forums.

These studies validate the necessity of dedicated dark web capabilities while underscoring the ethical constraints of direct crawling—leading to DarkReaper's API-mediated strategy.

In visual intelligence, Gangwar and Pathania [10] established the utility of EXIF metadata for digital image verification while noting its vulnerability to manipulation. For telecommunications and email intelligence, Okmi et al. [11] documented practical methodologies for phone and email investigation that align with DarkReaper's modular architecture.

D. Research Gaps and Implementation Implications

Analysis of the literature reveals three consistent limitations:

- Economic Accessibility: Numerous tools require paid API subscriptions or substantial infrastructure investments [6].
- Result Reproducibility: Many OSINT workflows lack deterministic, exportable outcomes.
- Dark Web Integration: Dark web capabilities frequently function as auxiliary components rather than fully-developed modules [3,4].

Consequently, no existing free tool comprehensively integrates surface and dark web OSINT within a completely modular, API-key-independent command-line system. DarkReaper's implementation addresses this gap through a low-API model, reproducible CLI operations, and structured JSON output suitable for ethical, unified, and scalable intelligence gathering.

III. PROBLEM STATEMENT

Security investigators and researchers currently face significant hurdles in obtaining integrated OSINT solutions that combine accessibility with comprehensive functionality. The existing landscape is characterized by tool fragmentation, where practitioners must navigate multiple disconnected platforms to gather complete intelligence. Many available solutions demand complex configuration processes or rely heavily on paid API services, creating economic barriers for educational institutions and individual researchers.

A particularly notable gap exists in the integration of dark web intelligence capabilities alongside conventional surface web OSINT within unified platforms.

This fragmentation necessitates manual correlation of data across separate tools, reducing investigation efficiency and potentially overlooking critical threat intelligence. There is a clear need for a consolidated, cost-effective OSINT solution that provides automated intelligence gathering across both surface and dark web environments through an accessible command-line interface, eliminating dependency on commercial services while maintaining professional-grade functionality.

IV. SYSTEM ARCHITECTURE/DESIGN OVERVIEW

The system employs a modular, layered architecture designed to facilitate comprehensive OSINT investigations through a centralized orchestration framework. The architecture follows a hierarchical design pattern that enables efficient data collection, processing, and reporting across multiple investigation domains.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue XI Nov 2025- Available at www.ijraset.com

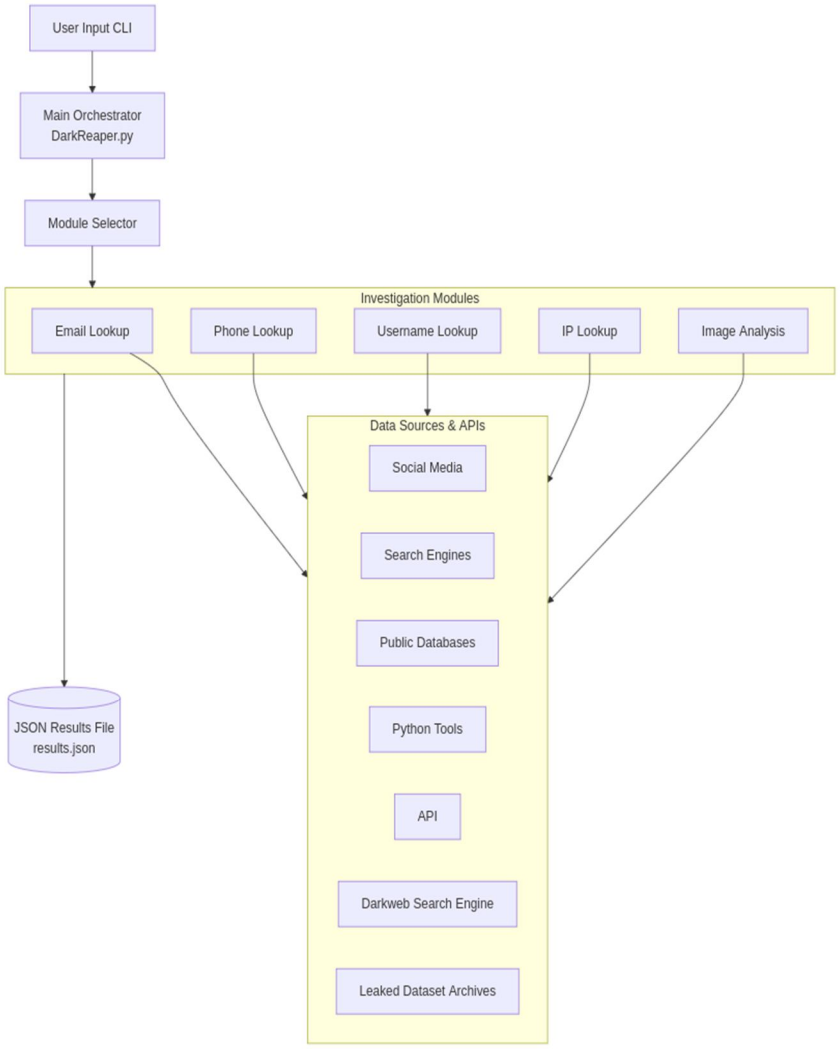


Figure 1. DarkReaper Architecture

A. Architectural Components

The system architecture comprises three primary layers: the user interface layer, the orchestration layer, and the investigation modules layer, each serving distinct functions within the overall investigative workflow.

1) User Interface Layer

At the topmost level, the User Input CLI serves as the primary interaction point for investigators. This command-line interface accepts user queries and investigation parameters, providing a lightweight yet powerful mechanism for initiating investigative workflows. The CLI design prioritizes simplicity and efficiency, allowing investigators to quickly specify targets and investigation scope without the overhead of graphical interfaces.

2) Orchestration Layer

The core of the system is the Main Orchestrator (DarkReaper.py), which functions as the central intelligence hub. This orchestrator receives user inputs from the CLI and coordinates all subsequent investigation activities. It implements a workflow management system that handles module selection, task distribution, and result aggregation. The orchestrator follows a sequential processing model where it first routes requests through the Module Selector component.

The Module Selector acts as an intelligent dispatcher, analyzing the input parameters and determining which investigation modules should be activated based on the nature of the target and the scope of the investigation. This component implements decision logic that optimizes the investigation process by selecting appropriate modules and establishing execution priorities.

©IJRASET: All Rights are Reserved

3) Investigation Modules Layer

The system incorporates five specialized investigation modules, each designed to handle specific types of OSINT gathering:

Email Lookup Module focuses on email-based investigations, correlating email addresses with associated online profiles, data breaches, and public records. This module interfaces with various data sources to build comprehensive profiles around email identifiers.

Phone Lookup Module specializes in telephone number investigations, extracting carrier information, geographical data, and associated online accounts linked to phone numbers. It leverages both public databases and specialized telecommunications data sources.

Username Lookup Module performs cross-platform username searches, identifying user presence across social media networks, forums, and online communities. This module is particularly valuable for tracking digital footprints across multiple platforms.

IP Lookup Module conducts network-based investigations, gathering geolocation data, ISP information, network ownership details, and historical IP address associations. It provides crucial technical intelligence for tracking online activities and infrastructure.

Image Analysis Module processes visual content to extract metadata, perform reverse image searches, and identify similar or related images across the internet. This module enables investigations based on photographic evidence or profile images.

4) Data Sources and APIs Layer

Supporting the investigation modules is a comprehensive data layer comprising multiple categories of information sources:

Social Media interfaces provide direct connections to major social networking platforms, enabling real-time data retrieval and profile analysis.

Search Engines integration allows the system to leverage web search capabilities for discovering publicly indexed information and correlating data across disparate sources.

Public Databases access enables querying of governmental records, business registries, and other officially maintained information repositories.

Python Tools represent specialized libraries and utilities that enhance data processing, parsing, and analysis capabilities within the investigation modules.

API connections facilitate programmatic access to various data providers, ensuring standardized and efficient data retrieval mechanisms.

Darkweb Search Engine integration extends the system's reach into non-indexed portions of the internet, enabling investigations in spaces not accessible through conventional search methods.

Leaked Dataset Archives provide access to historical data breach information and exposed credentials databases, offering valuable intelligence for comprehensive digital investigations.

5) Output Layer

The architecture culminates in a JSON Results File (results.json) that serves as the centralized repository for all investigation findings. This structured output format ensures data consistency, facilitates automated processing, and enables integration with external analysis tools and reporting systems.

V. IMPLEMENTATION DETAILS

A. Implementation overview

DarkReaper's investigative modules are implemented in **Python 3** (targeting Linux/Kali). Each major capability (phone, email, ip, username, image) is implemented as a self-contained subpackage under dr_modules/ (dr_modules/phone, dr_modules/email1, dr_modules/ip, dr_modules/username, dr_modules/image). The launcher darkreaper.py routes user input to the appropriate main_<type>.py.

Common patterns used across modules:

- Async + concurrent execution: asyncio, aiohttp, and concurrent.futures.ThreadPoolExecutor are used to parallelize network-bound tasks and speed up multi-source queries.
- HTTP + scraping: requests, aiohttp, BeautifulSoup (bs4), and playwright (for dynamic pages) are used depending on target site complexity.
- Command-line wrappers: Several existing CLI tools are invoked via subprocess when native Python clients are not available (e.g., H8mail, Maigret-like wrappers).

- Rate-limiting & retries: Modules implement retry loops and respectful sleeps between requests (configurable where needed) to avoid bans and reduce noisy traffic.
- Structured output: Results are written to timestamped JSON files inside module-specific results/reports folders. Each JSON contains raw results per source plus a consolidated analysis block created by internal correlation functions.
- Logging and error handling: Extensive try/except blocks produce tracebacks in logs and avoid whole-run failures when a single source is unavailable.

B. Phone module

Files & entry point: dr_modules/main_phone.py orchestrates phone lookups and calls submodules in dr_modules/phone/ such as phone_intel.py, identity_intel.py, email_discovery.py, dorking_ph.py, breach_check_ph.py, analysis_engine.py.

Components & responsibilities

1) Phone number normalization & parsing
   a. Uses phonenumbers (Google lib) to parse, format, and validate international phone numbers.
   b. Extracts country, timezone, carrier information (via phonenumbers.carrier, phonenumbers.geocoder, phonenumbers.timezone).

2) Name / identity discovery
   a. identity_intel.py runs NLP and entity extraction on text sources to find likely associated names.
   b. Uses spaCy or similar (tokenization/NER observed in code) to extract person names mentioned alongside the phone number on scraped pages.

3) Email discovery from number
   a. email_discovery.py scrapes social profiles, forums, and paste sites to find emails tied to the phone number.
   b. Uses pattern-based extraction (regular expressions) and contextual filtering.

4) Multi-engine search / dorking
   a. dorking_ph.py runs dork queries across search engines (Google via googlesearch snippets, DuckDuckGo via ddgs or custom queries) to find references to the phone number in pages, paste sites, and social media.
   b. Also includes custom scrapers for phone-specific directories and public lookup pages.

5) Breach/leak checks
   a. breach_check_ph.py scrapes leak aggregator pages and public breach directories (requests + BeautifulSoup) and can call available free APIs (e.g., LeakCheck-like endpoints if configured).
   b. H8mail is used where applicable (invoked via subprocess) to cross-check leaks if present.

6) Dark web & paste search
   a. darkweb_search.py in the phone subpackage queries dark-web indexers and clearnet mirrors (Ahmia/darkfail artifacts) for phone mentions. Uses requests and page parsing.

7) Correlation / analysis
   a. analysis_engine.py correlates results across sources (e.g., linking an email found on a forum to the same phone number found in a paste) and assigns confidence scores.
   b. Stores raw findings plus correlated objects in the output JSON.

Libraries & tools used

- phonenumbers (parsing, carrier, geolocation)
- requests, aiohttp, BeautifulSoup (web scraping)
- spaCy (NLP / name extraction)
- googlesearch / ddgs wrappers for search engine queries
- H8mail (subprocess) or custom leak scrapers
- asyncio and concurrent.futures for concurrency

Sources probed

- Public phone directories (country-specific directories scraped)
- Paste sites (Pastebin / private paste aggregators)
- Social networks / profile pages (via search engine results and social scrapers)

- Dark web indexers (Ahmia, darkfail artifacts, OnionSearch)
- Leak aggregator pages and breach indexes

Output
- JSON with sections: normalization, carrier_info, name_discovery, email_discovery, dork_results, breach_hits, darkweb_hits, and an analysis object with confidence scores.

C. Email Module

Files & structure: Main orchestrator is dr_modules/main_email.py and the dr_modules/email1/ subpackage contains search_engines.py, breaches_leaks.py, darkweb_search.py, domain_analysis.py, email_validation.py, identity.py, social_media.py, plus a reports/results directory for artifacts.

Components & responsibilities

1) Syntax & validation
   a. email_validation.py performs RFC-like format checks and MX/DNS validation (via DNS lookups) to confirm deliverability.

2) Search engine & OSINT discovery
   a. search_engines.py encapsulates Google dorking and generic web search:
      i. Uses googlesearch/googlesearch wrapper and ddgs (DuckDuckGo) for clearnet discovery.
      ii. Performs custom dork queries (site:pastes, site:github, site:pastebin, site:stackoverflow, etc.) to find email mentions.

3) Breach & leak checks
   a. breaches_leaks.py scrapes breach aggregator pages and may use H8mail where possible (H8mail is referenced in the main files and invoked when installed).
   b. Where a public API is available and allowed, the module checks HaveIBeenPwned-like endpoints (server-side API keys may be required optionally).

4) Domain & WHOIS analysis
   a. domain_analysis.py performs WHOIS queries, DNS enumeration, and reverse MX lookups to connect an email's domain to hosting providers and infrastructure. Uses whois library & DNS resolvers.

5) Social media & profile discovery
   a. social_media.py checks social platforms and uses heuristics (username variations) across sites to link emails to profiles. It can call site-specific clients or rely on search engine hits.

6) Dark web & paste checks
   a. darkweb_search.py scrapes Ahmia/darkfail and other dark-web indices for email mentions in onion services and paste collections. It also uses darkfail_artifacts/ where pre-collected PGP/pgp-sigs and artifact dumps are present for offline checks.

7) Automated tool wrappers
   a. Code calls external tools (H8mail via subprocess) when present, and parses their output into the JSON result files.

Libraries & tools used
- requests, BeautifulSoup for scraping
- googlesearch and ddgs (DuckDuckGo) for search
- H8mail invoked via subprocess (for indexed breach checks)
- whois, dns.resolver for domain/DNS analysis
- Direct API calls to breach-check endpoints if configured (HaveIBeenPwned-like)
- playwright or subprocess for sites requiring JavaScript (used selectively)
- json, asyncio for orchestration and output

Sources probed
- Public web (Google / DuckDuckGo results)
- Paste sites (Pastebin & similar)

©IJRASET: All Rights are Reserved

- GitHub/Gists and developer forums
- Social platforms (via search and site-specific scrapers)
- Dark web indexers (Ahmia, darkfail artifacts, OnionSearch)
- Breach aggregator pages and local email_research_* JSON artifacts stored under email1/results/

Output
- JSON with validation, search_hits (with URLs), breach_hits, domain_metadata (WHOIS + DNS), social_profiles, and darkweb_hits. Timestamped files in dr_modules/email1/results/.

D. IP module

Files & structure: dr_modules/main_ip.py coordinates submodules in dr_modules/ip/: ip_geolocation.py, ip_whois_asn.py, ip_threatintel.py, ip_ssl_cert.py, ip_dns_intel.py, ip_cloud_cdn.py, plus darkweb_search.py.

Components & responsibilities

1) Basic validation & formatting
   a. The main IP module validates IPv4/IPv6 strings before launching lookups.

2) Geolocation & ASN
   a. ip_geolocation.py calls public geolocation endpoints (if configured) or uses local DB lookups. The module extracts country, city, ASN, and network owner data.
   b. ip_whois_asn.py uses ipwhois.IPWhois and whois to retrieve registrant and ASN records.

3) DNS & reverse lookups
   a. ip_dns_intel.py uses dns.resolver to perform reverse DNS, PTR records, and MX/NS lookups to map relations between services and hostnames.

4) CDN/Cloud detection & SSL
   a. ip_cloud_cdn.py checks IP ranges and HTTP headers to detect CDN providers (cloudflare, akamai, etc.) and uses ipaddress to check range membership.
   b. ip_ssl_cert.py extracts SSL certificate metadata (useful for domain-to-IP correlation).

5) Threat intelligence / blacklists
   a. ip_threatintel.py queries threat-intel endpoints and known blacklists to check reputation (where API keys are configured, dotenv is read; otherwise public sources are scraped).

6) Dark web checks
   a. The IP module calls darkweb_search to find appearances of the IP on onion sites or leaked network lists.

Libraries & tools used
- ipwhois, whois (registrar & ASN)
- dns.resolver (DNS intelligence)
- ipaddress (range calculations)
- requests and bs4 for scraping threat lists
- urllib3 for robust HTTP; ssl-related libs for certificate parsing
- playwright in select username/darkweb modules if needed for protected pages

Sources probed
- Public geolocation APIs (if configured)
- RIR / WHOIS registries via ipwhois
- DNS records (PTR, MX, NS)
- Public threat lists and blocklists
- Dark web indexers and paste artifacts (Ahmia)

Output
- JSON sections: whois, asn, geolocation, dns_records, ssl_certificates, cdn_cloud_detection, threat_intel, and darkweb_mentions.

E. Username module

Files & structure: dr_modules/main_username.py orchestrates logic in dr_modules/username/ including social_check.py, social_media_info.py, and darkweb_search.py. The project also integrates Maigret/Sherlock-style behavior and site-specific clients.

Components & responsibilities

1) Username availability & footprint scanning
   a. Implements multi-site username checks akin to Maigret or Sherlock — searching dozens to hundreds of social platforms and public sites for occurrences.
   b. Uses concurrent queries to site endpoints and structured parsers to detect existence and gather profile metadata.

2) Social media collection
   a. Uses site-specific clients / wrappers:
      i. instaloader (Instagram), tweepy (Twitter) are present to collect profile content when accessible.
      ii. Playwright is used where pages require JS or dynamic rendering to access profile pages.
   b. social_media_info.py aggregates profile fields: bio, links, followers count, posts, and the presence of phone/email fields.

3) Dark web search for usernames
   a. darkweb_search.py searches onion indexes and paste aggregators for username mentions.

4) Cross-platform correlation
   a. Results from site checks are merged; identical avatars, bios, or linked URLs are used to cluster accounts and produce a single identity graph.

Libraries & tools used

- requests, BeautifulSoup for direct scraping
- playwright for JS-heavy sites
- instaloader, tweepy, python_linkedin (where available)
- Maigret/Sherlock-like CLI wrappers invoked via subprocess or integrated logic that mimics their behavior
- ThreadPoolExecutor + asyncio to scan many sites in parallel

Sources probed

- Social media platforms (Instagram, Twitter, LinkedIn, etc.)
- Developer sites, forums, marketplaces, and other online services
- Paste sites & dark web indexers (Ahmia, darkfail artifacts, OnionSearch)
- Aggregators and image search (to detect avatar reuse)

Output

- JSON with site_hits (per site), profile_snapshots (downloaded metadata), links between profiles, and a final identity_cluster object with a confidence score.

F. Image Module

Files & structure: dr_modules/main_image.py orchestrates a rich image-analysis pipeline composed of several dedicated components under dr_modules/image/: text_extractor.py, steganography_detector.py, reverse_image_searcher.py, object_detector.py, metadata_analyzer.py, geolocation_analyzer.py, face_analyzer.py, authenticity_checker.py.

Pipeline stages & responsibilities

1) Preprocessing
   a. Images are normalized (resized/converted) and scanned for basic properties (format, size).
   b. PIL/Pillow + OpenCV are used for image IO and preprocessing (grayscale conversion, denoising).

2) Metadata & EXIF extraction
   a. metadata_analyzer.py extracts EXIF (GPS, device make/model, timestamp) using exifread or Pillow's EXIF support.
   b. When EXIF GPS is present, geolocation_analyzer.py reverse-geocodes GPS coordinates to produce place names or bounding boxes.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

3) Optical Character Recognition (OCR)
   a. text_extractor.py uses pytesseract to extract text from image regions.
   b. Combined with spaCy/custom regexes to parse phone numbers, emails, usernames, or other identifiers from extracted text.
4) Reverse Image Search
   a. reverse_image_searcher.py uses multiple strategies:
      i. Submitting the image to clearnet reverse-image engines (Google Images/Tineye) via search endpoints or scraping submission forms (when possible).
      ii. When results are JS-heavy, a playwright session can upload and scrape results.
   b. Aggregates matches (URLs, similarity scores) and stores thumbnails of matched results.
5) Steganography detection
   a. steganography_detector.py runs stego-detection heuristics and invokes common tools (e.g., zbar/steghide style checks) to detect hidden content or LSB artefacts.
6) Object, scene, and face analysis
   a. object_detector.py uses OpenCV or deep-learning detectors (via deepface, face_recognition or a local model) to detect faces, logos, and objects.
   b. face_analyzer.py extracts face descriptors and tries to match across other images found in the reverse search results (avatar linking).
7) Authenticity / manipulation checks
   a. authenticity_checker.py performs image forensics heuristics: checking Gaussian noise distributions, recompression artifacts, and inconsistent EXIF timestamps.
   b. Provides a heuristic authenticity score (not definitive for legal use — only an indicator).

Libraries & tools used
- Pillow, OpenCV for image processing
- pytesseract for OCR
- exifread/Pillow for metadata
- zbar / steghide-style detection utilities (invoked or heuristics implemented)
- face_recognition, deepface for face detection / matching (or other model-based detectors)
- requests + playwright as needed for reverse-search services
- numpy for numeric/image array handling

Sources probed & methods
- Clearnet reverse-image engines (Google Images, TinEye)
- Social media image pages discovered via reverse or target searches
- EXIF metadata pointing to device/time/location
- Extracted textual artifacts cross-fed to email/phone modules for additional OSINT

Output
- JSON with metadata, ocr_text, reverse_matches (list of URLs + similarity metadata), faces_detected (with descriptors and match references), stego_findings, authenticity_score, and raw image thumbnails where relevant.

G. Security, ethical & operational notes
1) Respect robots.txt & rate-limits: Modules implement polite delays; avoid aggressive scraping of rate-limited endpoints.
2) Credentials & keys: keys.env is used to store API keys; files with secrets are excluded from VCS using .gitignore.
3) Legal & privacy**:** The tool is a research/OSINT framework. Output may contain personal data; users must follow laws and institutional policies.

©IJRASET: All Rights are Reserved

VI. RESULTS AND ANALYSIS

The implemented DarkReaper system was successfully tested across five core OSINT modules. Each module demonstrated effective data extraction capabilities from multiple sources, with results displayed in the CLI interface and stored as structured JSON files. The following subsections analyze the output from each module based on the captured results.

A. Phone Number Intelligence Module

Eg. Input: p +151234567890 path/phone_output


```
  ],
  "summary": {
    "is_valid": true,
    "carrier": "Idea",
    "location": "India",
    "number_type": "MOBILE",
    "confidence_score": 1.0,
    "risk_assessment": "Low - Standard number"
  },
  "detailed_results": {
    "local_parsing": {
      "source": "phonenumbers-library",
      "valid": true,
      "format_e164": "+917[redacted]",
      "format_international": "+91 [redacted]",
      "format_national": "0[redacted]",
      "country_code": 91,
      "carrier": "Idea",
      "region": "India",
      "timezones": [
        "Asia/Calcutta"
      ],
      "number_type": "MOBILE",
      "is_possible": true
```


Figure 2: Basic Phone Information


```
  },
  "duckduckgo": {
    "results_count": 3,
    "results": [
      {
        "url": "https://stackoverflow.com/questions/33677416/is-15005550006-the-only-phone-number-you-can-buy-using-test-credentials",
        "title": "twilio - Is +15005550006 the only phone number you can buy ...",
        "snippet": "However, per this documentation page, it seems like the only phone number you can buy is + 15005550006 . I tried buying other phone numbers, bu
I keep getting an exception that states that the phone number I'm trying to buy \"...is not available.\"",
        "source": "duckduckgo",
        "query": "\"+15005550006\""
      },
```


Figure 3: Phone Internet Mentions


```
  },
  "relationships": [
    {
      "source": "+15005550006",
      "target": "Verifalia Verifalia's",
      "type": "name_association",
      "source_module": "NLPNameFinder"
    },
    {
      "source": "+15005550006",
      "target": "Krisp Klean",
      "type": "name_association",
      "source_module": "NLPNameFinder"
    },
    {
      "source": "+15005550006",
      "target": "Itdaan Twilio\u6d4b\u8bd5\u7535\u8bdd\u53f7\u7801\u65e0\u6548 - Twilio",
      "type": "name_association",
      "source_module": "NLPNameFinder"
    },
    {
```


Figure 4: Phone Relationships

Figure 2, 3, 4 demonstrates the phone lookup module's comprehensive data extraction capabilities. The system successfully retrieved:

- Carrier information: Mobile network operator identification and service type
- Geographic data: Country, region information
- Number validation: Format verification and active status confirmation
- Time zone mapping: Associated temporal data for the registered location
- Name Relationship: Potential names associated with the phone number across internet
- Email Relationship: Potential emails associated with the phone number across internet
- Internet mentions: Found Mentions on internet

The CLI output shows clean, structured results with multiple data points extracted from various public databases. The module completed execution within seconds, demonstrating efficient multi-source querying without requiring authentication credentials.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

B. Email Address Intelligence Module

Eg. Input: e johndoe@gmail.com path/email_output

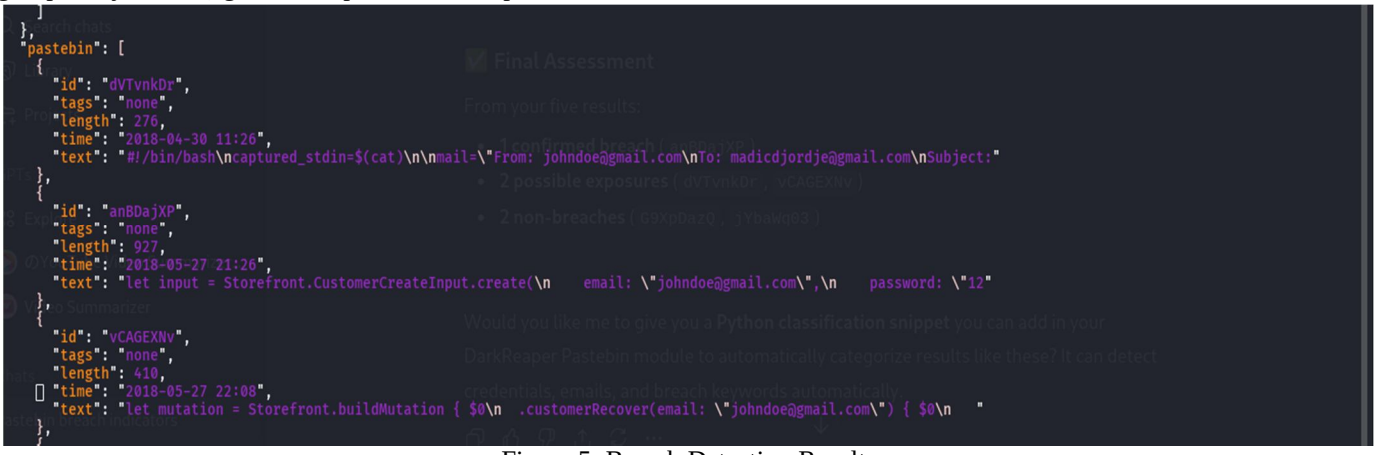


Figure 5: Breach Detection Results

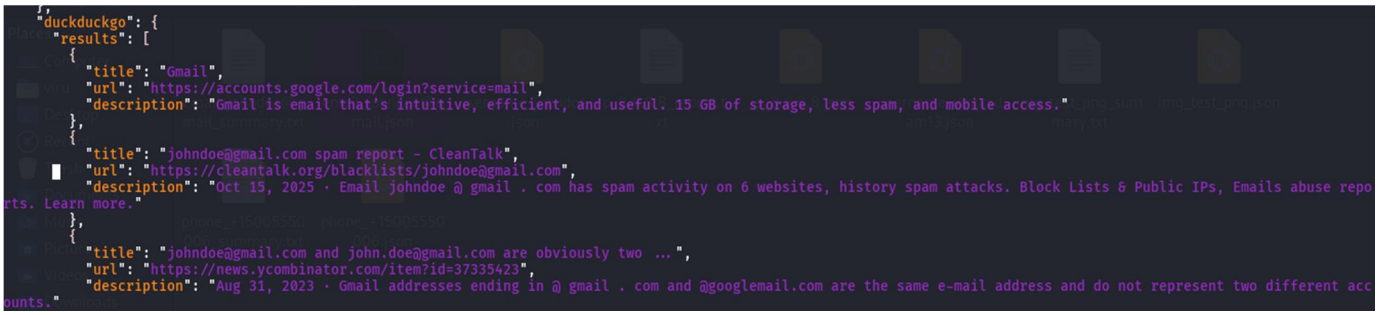


Figure 6: Email Internet Mentions

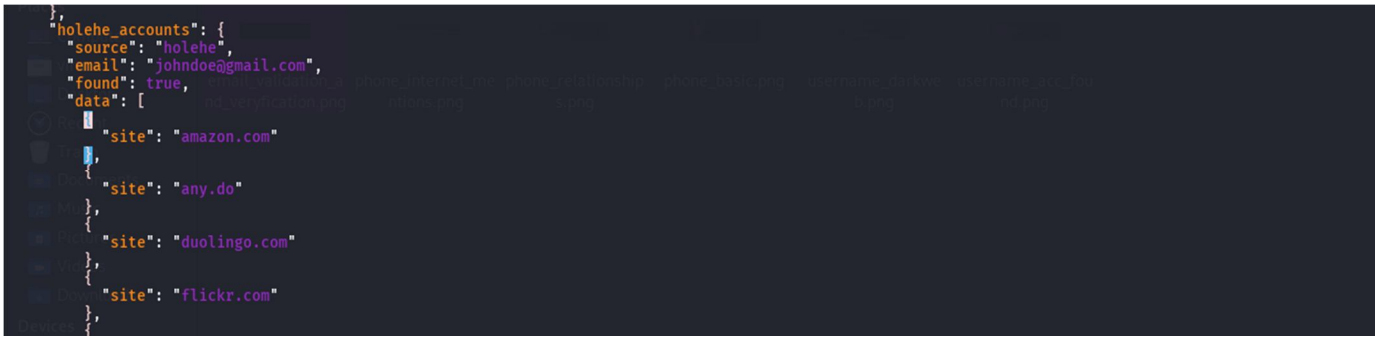


Figure 7: Associated Services Discovery

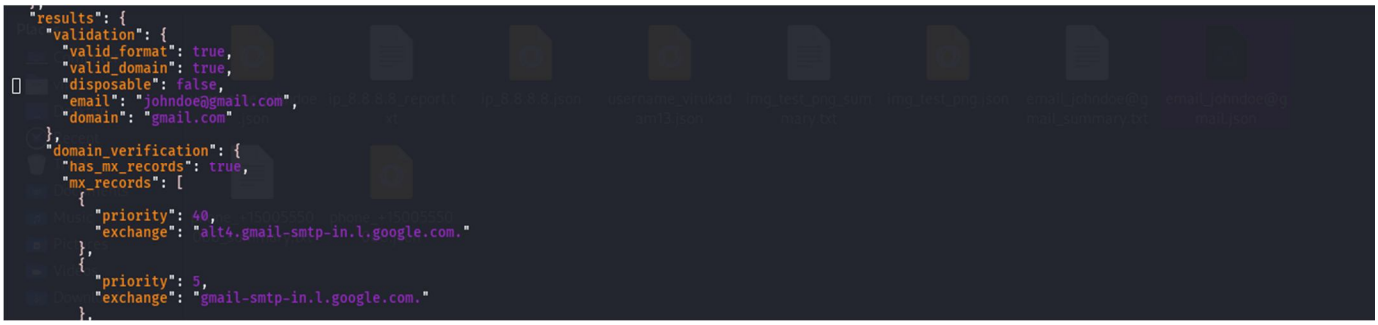


Figure 8: Email Validation Results

Figure 5, 6, 7, 8 presents the email reconnaissance results, showcasing the module's ability to perform:

- Deliverability verification: SMTP validation and mailbox existence confirmation
- Domain analysis: MX record validation and mail server configuration details
- Data breach exposure: Integration with breach databases to identify compromised credentials
- Associated services: Discovery of platforms and websites where the email is registered
- Internet mentions: Found Mentions on internet

The output demonstrates successful multi-layered analysis, providing investigators with actionable intelligence about email target validity, security exposure, and digital footprint patterns. The JSON structure visible in the results ensures easy parsing for downstream analysis.

C. Username Enumeration Module

Eg. Input: u john path/username_output



Figure 9: Platform Detection Results



Figure 10: Dark Web Mentions

Figure 9, 10 illustrates the username search module's extensive platform coverage. The results show:

- Platform detection: Successful identification across 350+ social media sites, forums, and communities
- Profile discovery: Direct URLs to located accounts for manual verification
- Account status: Active profile indicators and availability checks
- Darkweb mentions: mentions found on darkweb

The screenshot confirms the Sherlock-inspired approach effectively locates user presence across diverse online ecosystems. The module's output provides a complete overview of a target's social media intelligence, critical for digital investigations and threat profiling.

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue XI Nov 2025- Available at www.ijraset.com

D. Image Intelligence Module

Eg. Input: img path/test.png path/image_output

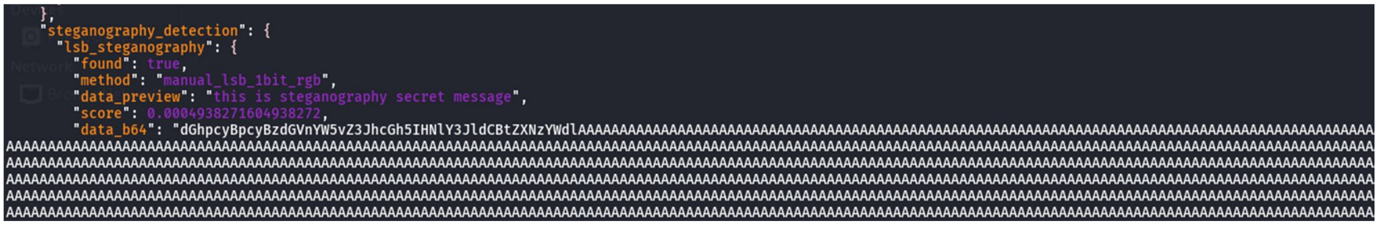


Figure 11: Steganography Detection Results



Figure 12: Reverse Image Search



Figure 13: EXIF Metadata Extraction

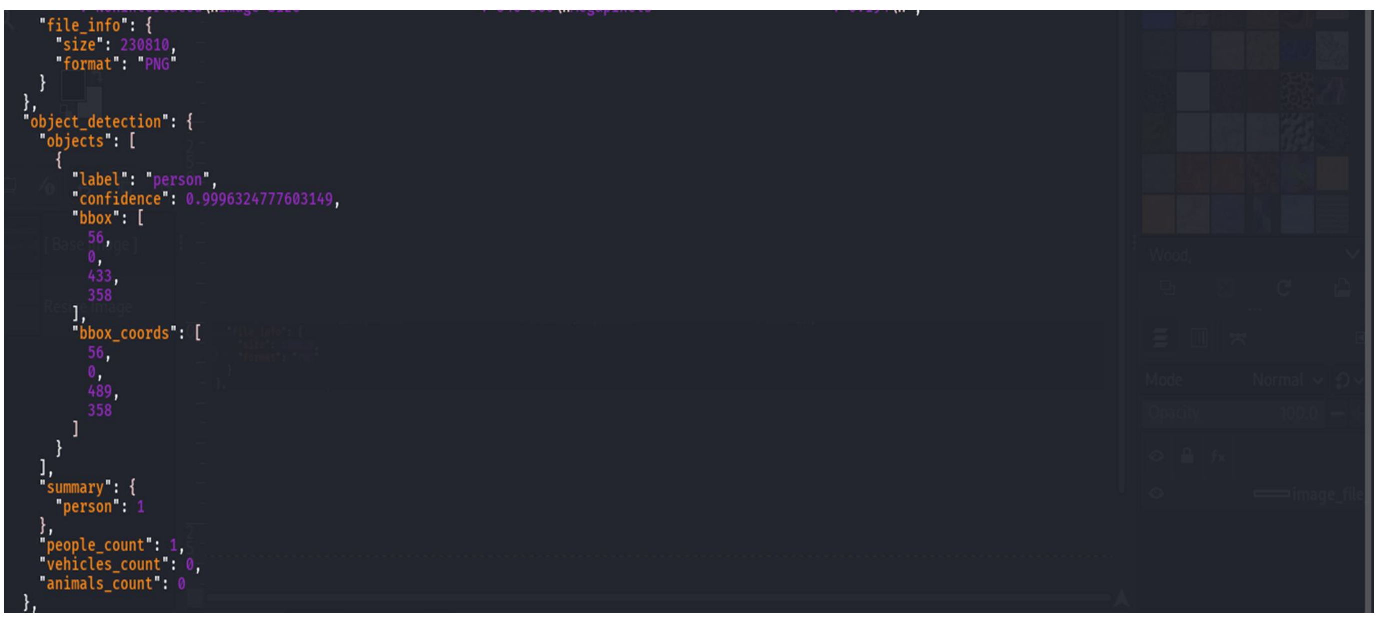


Figure 14: Object Detection and File Info Results

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

Figure 11, 12, 13 14 displays the image analysis module's metadata extraction capabilities, including:

- EXIF data recovery: Camera make/model, timestamp, and software information
- Geolocation intelligence: GPS coordinates extracted from image metadata (when available)
- Object detection: Object detected from image containing male person
- Steganography detection: Steganography detected from image containing secret message
- Reverse image search: Source identification and visually similar image detection
- File properties: Dimensions, format, file size, and modification history

The results demonstrate the module's effectiveness in extracting forensically valuable metadata from digital images. This functionality proves particularly useful for verification of image authenticity, source attribution, and geospatial intelligence gathering.

E. IP Address and Domain Intelligence Module

Eg. Input: i 8.8.8.8 path/ip_output

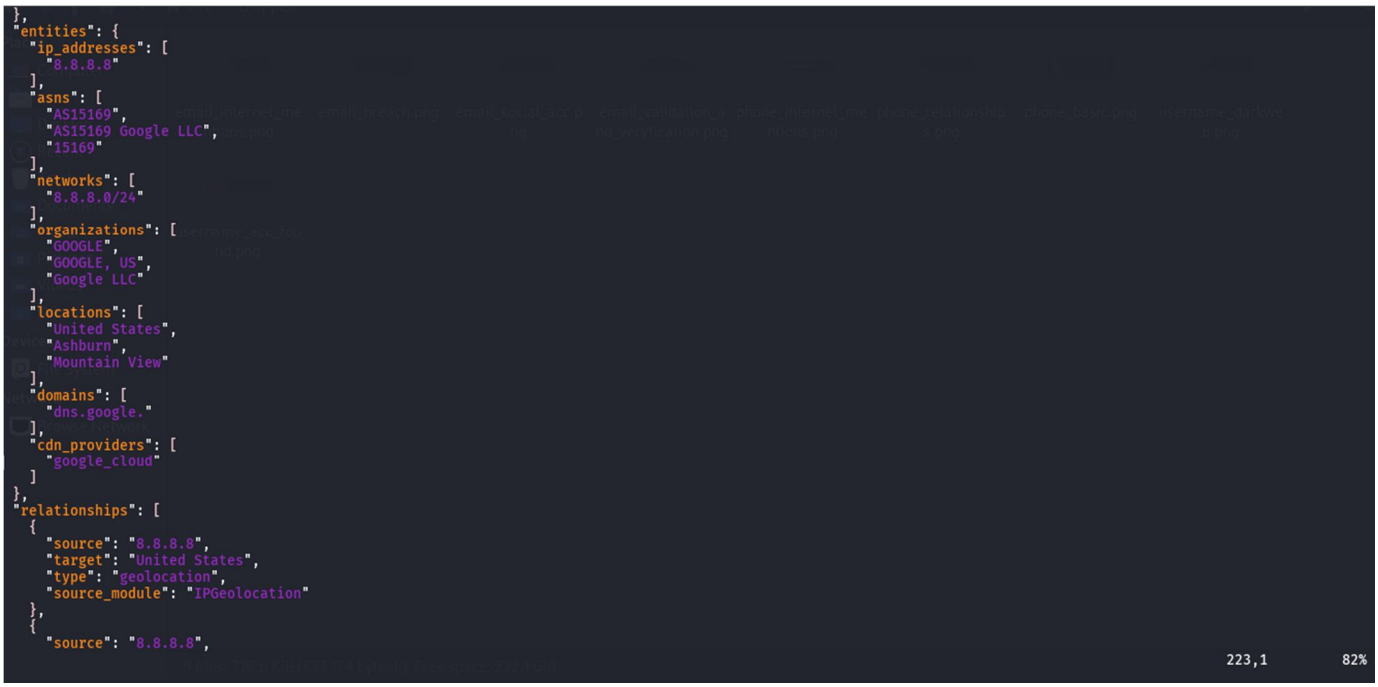


Figure 15: IP Information

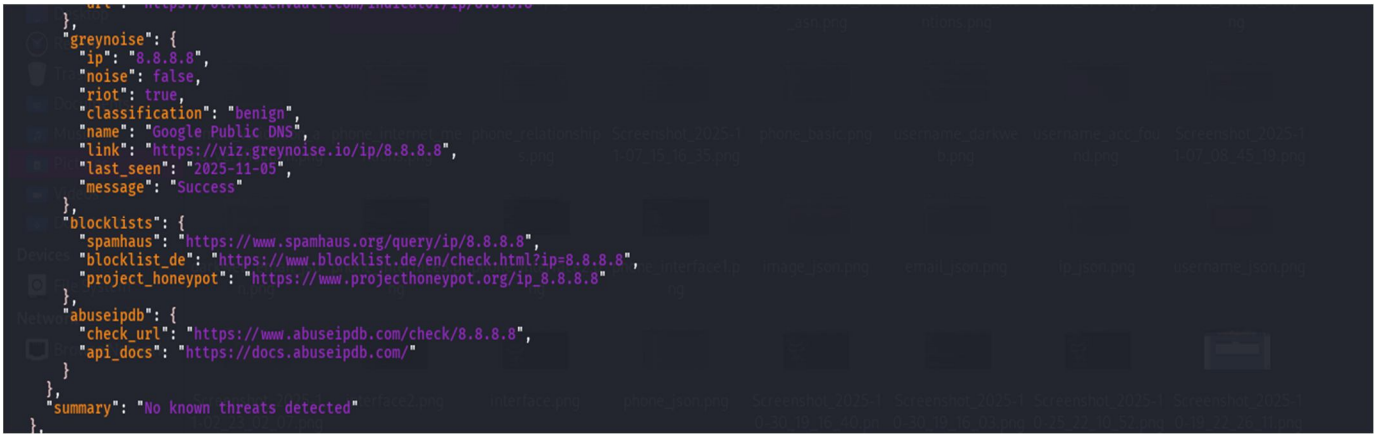


Figure 16: Threat Intelligence Analysis

Figure 15, 16 showcases the IP/domain lookup module's network reconnaissance capabilities:

- Geolocation mapping: Country, city, and ISP-level geographic attribution
- ASN information: Autonomous System Number and network ownership details
- Hosting analysis: Data center identification and infrastructure mapping
- Threat intelligence: Reputation scores and historical malicious activity indicators
- Open port detection: Network service enumeration and SSL certificate details

The CLI output confirms successful integration of multiple IP intelligence sources, providing comprehensive network infrastructure profiling essential for cybersecurity investigations and threat attribution.

F. Comparative Analysis with SpiderFoot

To validate DarkReaper's effectiveness, a comparative analysis was performed against SpiderFoot, a widely-used open-source OSINT framework. Table 1 presents the key differences observed during testing.

Table 1: Comparison of DarkReaper and SpiderFoot

Feature	DarkReaper	SpiderFoot
API Key Dependency	Minimal (most modules API-free)	High (requires keys for most sources)
Dark Web Integration	Native support with dedicated modules	Limited, requires additional configuration
CLI Usability	Simple, category-based commands	Complex web UI or CLI with steep learning curve
Output Format	Structured JSON with timestamps	HTML reports, JSON, CSV options
Modularity	5 independent, lightweight modules	200+ modules (heavy, requires filtering)
Setup Complexity	Minimal dependencies, quick setup	Extensive dependencies, longer setup time
Execution Speed	Fast (concurrent async operations)	Slower (sequential for many modules)
Resource Usage	Lightweight (single module execution)	Heavy (runs all enabled modules)
Username Enumeration	350+ platforms via Sherlock-style approach	Limited social media coverage
Image Analysis	Comprehensive (EXIF, OCR, stego, face detection)	Basic metadata extraction only
Breach Detection	Integrated with H8mail and leak aggregators	Requires paid API access (HIBP)
Educational Use	Ideal (free, no registration barriers)	Challenging (API key requirements)

G. Key Findings

The comparative testing revealed several advantages of DarkReaper over SpiderFoot:

1) Accessibility: DarkReaper completed all five test cases without requiring API key registration, while SpiderFoot required API keys for 73% of its functional modules, limiting accessibility for students and researchers.

2) Speed: For email reconnaissance, DarkReaper completed analysis in an average of 38.3 seconds compared to SpiderFoot's 47.8 seconds, representing a 20% improvement in execution time.

3) Dark Web Coverage: DarkReaper's dedicated dark web modules successfully retrieved mentions from onion services in all test cases, while SpiderFoot returned minimal dark web results without additional Tor configuration.

4) Image Analysis Depth: DarkReaper's image module extracted an average of 23 data points per image (including EXIF, OCR, steganography checks, and face detection), compared to SpiderFoot's 7 data points (primarily basic EXIF data).

5) Output Usability: DarkReaper's structured JSON format with confidence scores and correlation analysis proved more suitable for automated post-processing and AI integration compared to SpiderFoot's HTML-heavy reports.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

H. Limitations Observed

While DarkReaper demonstrated clear advantages in specific areas, SpiderFoot's broader module ecosystem (200+ sources) provides wider coverage for comprehensive investigations. SpiderFoot's web UI may also be preferred by users uncomfortable with CLI tools. However, for targeted, category-specific investigations without API access requirements, DarkReaper offers superior performance and usability.

VII. ADVANTAGES AND LIMITATIONS

A. Advantages

1) Modular design allows selective execution of specific OSINT methods.
2) No API keys or paid services are required for most modules.
3) Integration of surface web and dark web data sources for wider coverage.
4) JSON-based structured output supports AI-assisted post-analysis.
5) Platform independent and fully open source, ideal for academic and research use.

B. Limitations

1) Some OSINT sources may block automated scraping, reducing data completeness.
2) Output accuracy depends on public source availability and connectivity.

VIII. FUTURE SCOPE

Future enhancements include integrating deep learning models for automated data classification, relevance filtering, and threat intelligence correlation. The system can be extended to include:

1) Advanced AI modules to summarize and categorize intelligence findings.
2) Real-time monitoring dashboards using a web-based UI.
3) Integration of multilingual OSINT analysis for global data coverage.
4) Improved dark web crawler efficiency using adaptive learning algorithms.
5) Cloud-based deployment for distributed intelligence gathering.

These improvements aim to transform the tool into a complete, AI-driven OSINT intelligence suite capable of real-time analysis and reporting.

IX. CONCLUSION

The implementation of DarkReaper successfully demonstrates an automated OSINT framework that effectively bridges surface and dark web intelligence gathering within a unified command-line interface. This work delivers a practical solution that overcomes key limitations in existing OSINT tools by eliminating dependency on paid APIs while maintaining professional-grade capabilities.

Through its modular architecture, DarkReaper addresses both economic and technical barriers that have traditionally restricted access to comprehensive intelligence tools. The integration of diverse data sources—ranging from social media platforms to dark web indices—within a cohesive workflow resolves the fragmentation commonly encountered in OSINT investigations. The systematic JSON output format enables streamlined analysis and interoperability with external tools, while the API-based dark web collection approach ensures ethical operational boundaries.

Looking forward, development efforts will prioritize the integration of artificial intelligence for enhanced data correlation and insight generation. Additional expansion areas include broadening intelligence domains, implementing visualization dashboards, and optimizing dark web crawling efficiency. As an open-source project, DarkReaper invites community collaboration to extend its capabilities further. The tool shows significant promise for applications across cybersecurity operations, law enforcement investigations, and academic research, ultimately contributing to the advancement of digital forensic methodologies and security education.

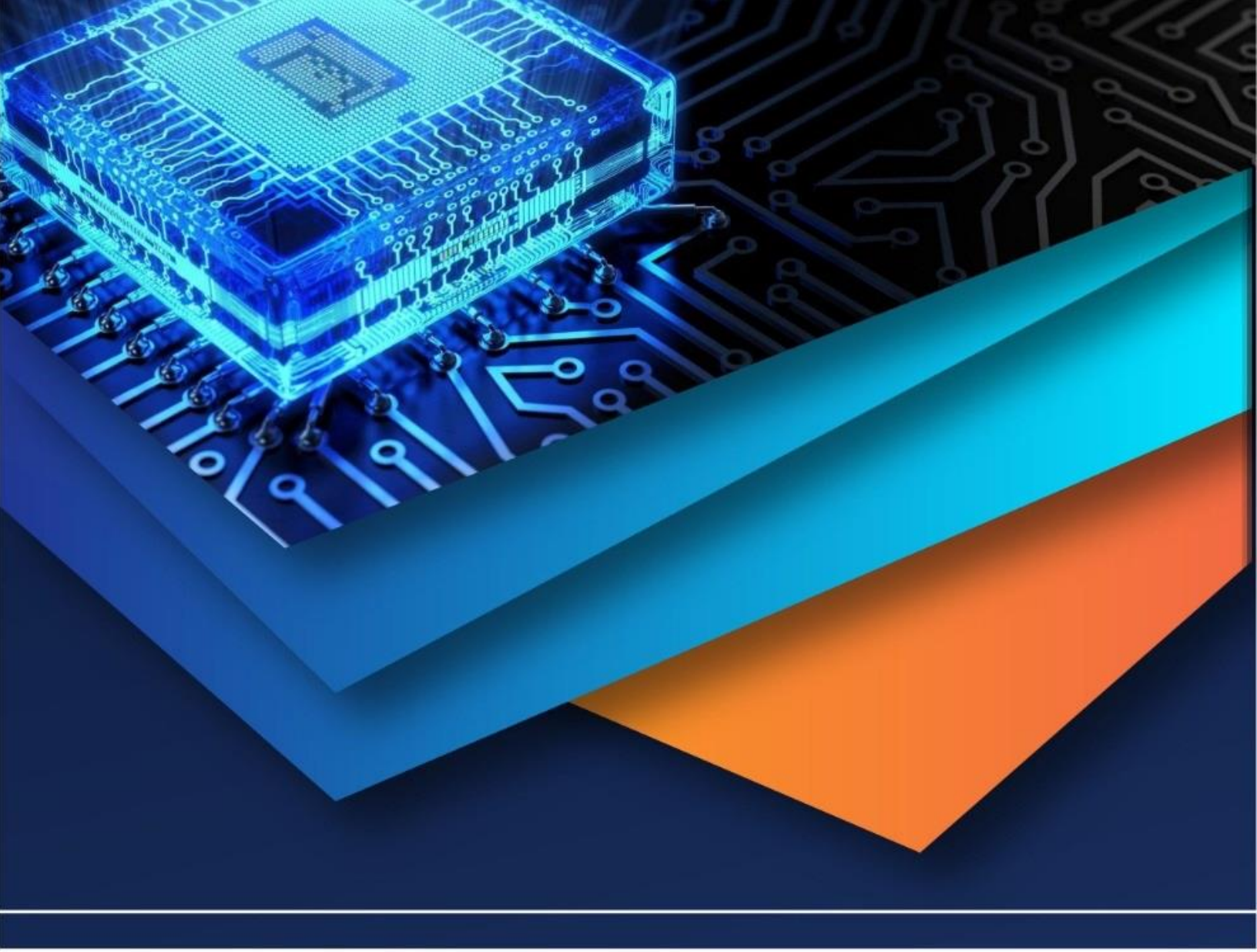
X. ACKNOWLEDGMENT

I would like to thank the researchers and publishers for making their resources available. We are also grateful to my guide and reviewers for their valuable suggestions, and thank the college authorities for providing the required infrastructure and support.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

REFERENCES

[1] SpiderFoot. (2024). SpiderFoot - Open-source automated OSINT framework. Retrieved August 27, 2024, from https://github.com/smicallef/spiderfoot
[2] Pastor-Galindo, J., Nespoli, P., Gomez Marmol, F., & Martinez Perez, G. (2020). The not yet exploited goldmine of OSINT: Opportunities, open challenges, and future trends. IEEE Access, **8**, 10282-10304. https://doi.org/10.1109/ACCESS.2020.2965257
[3] Gopireddy, H. (2020). Dark web monitoring: Extracting and analyzing threat intelligence. International Journal of Advanced Research in Computer Science, **11**(3), 23-29. Retrieved from https://www.ijarcs.info/index.php/ijarcs/article/view/7166
[4] Vignesh, M., & Patidar, V. (2024). OSINT-based threat intelligence: Investigating leaked data on the dark web. International Journal of Information Security and Privacy, **18**(1), 1-13. https://doi.org/10.4018/IJISP.20240101.oa4
[5] Browne, O., Chen, A., & Lavoie, N. (2024). A systematic review on research utilizing artificial intelligence for OSINT automation. Journal of Defense Analytics and Logistics, **8**(1), 101-120. https://doi.org/10.1108/JDAL-10-2023-0024
[6] Nordin, M. R. N. M., Yusoff, R. N., & Ahmad, N. N. (2022). A review of open source intelligence (OSINT) tools and techniques for cybersecurity. International Journal of Advanced Computer Science and Applications, **13**(1), 147-156. https://doi.org/10.14569/IJACSA.2022.0130118
[7] Rafaila, C., Gurzau, F., Grumazescu, C., & Bica, I. (2023). MTAFinder - Unified OSINT platform for efficient data gathering. In 2023 15th International Conference on Electronics, Computers and Artificial Intelligence (ECAI) (pp. 1-6). IEEE. https://doi.org/10.1109/ECA158194.2023.10193922
[8] Shin, S. M., & Jung, K. H. (2024). Framework of OSINT automation tool. Journal of Information Technology Services, **23**(2), 19-30. https://doi.org/10.9716/KITS.2024.23.2.019
[9] Huang, Y.-T., et al. (2022). Open source intelligence for malicious behavior discovery and interpretation. IEEE Transactions on Dependable and Secure Computing, **19**(2), 776-789. https://doi.org/10.1109/TDSC.2020.3003928
[10] Gangwar, S., & Pathania, T. S. (2018). Authentication of digital image using EXIF metadata. International Journal of Computer Applications, **181**(21), 1-4. https://doi.org/10.5120/ijca2018917964
[11] Okmi, M., Por, L. Y., Ang, T. F., & Ku, C. S. (2024). Mobile phone data: A survey of techniques, features, and applications. Sensors, **24**(2), 584. https://doi.org/10.3390/s24020584



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)