Proceedings of CONF-MLA 2025 Symposium: Intelligent Systems and Automation: AI Models, IoT, and Robotic Algorithms
DOI: 10.54254/2755-2721/2025.LD28446

Network Security Analysis Based on Web Crawler Technology

Jizhou Liu

Trinity Collegiate School, South Carolina, Darlington, USA
L2025326830AM@outlook.com

Abstract. With the increasing complexity of network security threats, web crawler technology has become an important tool in the field of network security due to its automated information collection capability. This study synthesizes 10 relevant literatures to explore the application of crawler technology in network security. It first outlines the main types of crawlers and their technical principles, with a focus on analyzing Python-based frameworks. Then, it examines its specific applications in vulnerability detection, such as XSS cross-site scripting vulnerability detection, automated SQL injection detection, and malicious crawler identification. In addition, combined with some researches, it discusses the challenges faced by crawler technology, such as anti-crawler mechanisms, compliance, and privacy protection. Finally, it looks forward to integrating machine learning into crawler strategy optimization and constructing an intelligent security detection framework, aiming to provide references for network security research and practice. It also provides future research directions for researchers in related fields.

Keywords: Web crawler technology, Network security, Vulnerability detection, Anti-crawler mechanism.

1. Introduction

In the digital age, the popularization of Web applications and the explosive growth of data have made network security threats increasingly complex, such as XSS cross-site scripting vulnerabilities, SQL injection, malicious crawler attacks, etc., which all seriously threaten information systems. Traditional manual detection methods are inefficient and difficult to meet the needs of large-scale, real-time security monitoring, making automated technologies represented by web crawlers a key tool in the field of network security.

Web crawlers provide a method for automatically traversing the network and extracting various types of information. They have achieved remarkable results in vulnerability mining and malicious behavior detection. These tools are important because they can enhance our network defense capabilities.

Several studies have drawn some conclusions: Hu Hongling explored the application of Python web crawlers in information security monitoring, involving aspects such as network public opinion management and threat intelligence aggregation, and proposed preventive measures such as anti-crawler mechanisms and data encryption [1]. Liu Feifei focused on XSS cross-site scripting vulnerability detection, studying how to discover targets with XSS vulnerabilities through web

DOI: 10.54254/2755-2721/2025.LD28446

crawler methods such as scanning URLs and pages [2]. They found that the disadvantage is that once some attack codes are inserted into the target URL or content, the problems hidden by these codes cannot be directly detected. Therefore, they proposed the idea of adding an attack code generator to detect and fix these problems. Chai developed a dynamic and efficient web crawler for XSS vulnerability detection, which uses a parallel algorithm to process URLs and an improved Bloom filter to achieve better detection results [3]. Xiao Qiuping used crawler and fuzz testing technologies to build an automated SQL injection detection system, which can not only locate injection points but also evaluate security levels [4]. Eswaran et al. proposed an enhanced network intrusion detection system (NIDS) that can identify malicious crawlers based on learned characteristics of malicious crawlers and conduct correlation analysis of security events [5].

This paper integrates and sorts out all the above studies. Firstly, it sorts out the current application methods of web crawlers; secondly, it expounds on their specific applications in vulnerability detection, malicious crawler identification, etc.; then, it analyzes the technical problems that web crawlers may face and proposes some feasible solutions. These contents are not only beneficial to the research field but also have reference value for practical work, aiming to contribute to optimizing the application of web crawlers in the field of network security and the standardized development of related fields.

2. Overview of web crawler technology

2.1. Technical definition and core principles

A web crawler, also known as a web spider, is a program or script that traverses websites and crawls web data in a user-like manner according to certain rules [6]. Its working principle is to simulate user requests to the server using the HTTP/HTTPS protocol, starting from the initial URL (seed URL), and recursively crawl new links appearing on the page until the requirements are met (covering the target website or reaching the required crawling depth).

From a technical perspective, the core components of a crawler include the following:

URL queue: Stores URLs to be crawled and already crawled to avoid repeated visits (e.g., using Bloom Filter algorithm or MD5 hash for deduplication); Page downloader: Sends HTTP requests to obtain web page source code and supports processing dynamic content (such as JavaScript-rendered pages, which requires tools like Selenium) [7]; Parser: Extracts valid information (such as links, forms, text) from web page source code. Common tools include Python's BeautifulSoup (for parsing HTML) and Scrapy framework (integrating crawling, parsing, and storage functions); Data storage module: Stores the crawled content in a database (for example, using MongoDB to store unstructured data and MySQL to store structured data) for later analysis [8].;

For Python, with the increasing richness of third-party libraries such as Requests and Scrapy, the development of crawlers has become increasingly simple. The Requests library can be used to quickly send HTTP requests, and the Scrapy framework greatly improves efficiency through its 4 modular designs ("engine-scheduler-downloader-crawler") to complete concurrent crawling and distributed deployment [9].

2.2. Main types and characteristics of web crawler technologies

Based on different crawling targets and strategies, web crawlers can be divided into the following main types:

DOI: 10.54254/2755-2721/2025.LD28446

2.2.1. General crawler

Crawls information across the entire network to build indexes for search engines (such as Google crawler and Baidu spider), belonging to the category of generalized crawlers. It is characterized by a wide coverage but inability to achieve precise target positioning. How to balance crawling depth and resource consumption has always been a research hotspot. Due to the huge amount of information, distributed methods need to be adopted for data collection. The crawling speed of crawlers is restricted according to the Robots protocol [10].

2.2.2. Focused crawler

A crawler with a strong purpose for a specific topic or website. It mainly uses relevance algorithms to filter out links unrelated to the target task, thereby achieving precise data crawling. For example, network security-focused crawlers can specifically crawl web pages with keywords such as "vulnerability" and "attack", or focus on parts of the website that may have vulnerabilities, such as forms and URL parameters. Its advantages are low resource consumption and high accuracy, making it suitable for XSS vulnerability detection and SQL injection scanning.

2.2.3. Incremental crawler

Only crawls the updated part of data in previously crawled pages instead of re-crawling all data, which can greatly reduce bandwidth and storage consumption. The key is to adopt methods to identify whether a page has changed, and common methods here are comparing the page's hash value or the last modification time of the page. This method is more used to regularly detect the update of vulnerability patches and the spread trajectory of malicious code on certain sites [10].

2.2.4. Distributed crawler

Taking distributed crawlers as an example, multiple computers participate in crawling work, and multiple nodes work together to complete different crawling tasks, which can achieve high crawling efficiency. For example, the dynamic parallel crawler (DPS) proposed by Chai achieves a crawling rate of hundreds of pages per second by optimizing URL scheduling and DNS resolution, which is suitable for large-scale vulnerability detection of large websites [3]. However, this mode needs to consider issues such as data consistency between nodes and coordinated operation of tasks.

2.2.5. Deep web crawler

A crawler for "deep web pages" that require form submission and login to access. It crawls web pages by simulating automatic form filling and login with cookies. At the same time, in network security, such crawlers can be used to detect the risk of unauthorized access to pages with permission restrictions. However, when crawling deep websites, problems such as verification and session timeout may be encountered [7].

Some types of crawlers focus on network security. For example, general crawlers are used to capture global threat intelligence, focused crawlers and distributed crawlers are used for targeted vulnerability detection, and incremental crawlers are used for continuous security situation monitoring.

DOI: 10.54254/2755-2721/2025.LD28446

3. Applications of web crawler technology in the field of network security

3.1. Vulnerability detection and protection

Web crawler technology can effectively identify common Web security issues such as XSS vulnerabilities and SQL injection vulnerabilities by automatically traversing the target system, imitating attacker behaviors, and analyzing the response packets to extract potential risk points of the detected system.

According to Liu Feifei's research on XSS cross-site scripting vulnerabilities, crawlers can construct malicious scripts (e.g., <script>alert(1)</script>) by extracting information from user input points (such as forms and URL parameters) on the page, and then detect whether the script is executed in the returned results of the page. If it is executed, it indicates that the vulnerability exists. Chai improved this process by using his designed dynamic and efficient crawler, which adopts parallel crawler technology to scan multiple pages simultaneously and uses an improved Bloom Filter algorithm for deduplication to avoid frequent repeated requests. The crawler crawls one page each time and records the detection results, which improves the detection efficiency and can meet the needs of batch scanning large websites to a certain extent.

According to Xiao Qiuping's research on SQL injection detection, an automated detection system can be developed by combining crawler and Fuzzing technologies. The crawler obtains dynamic URLs (links with parameters) of the target website, then replaces some parameters or form fields with various injection test cases (such as "syntax errors") generated by the Fuzzing module, and sends the replaced URLs or modified forms to the target website. At this time, the system judges whether there is an injection point by reading the error information returned by the database (e.g., "syntax error" reported by MySQL database) or changes in the page, and then gives a risk assessment level. The system supports scanning various injection types such as error-based injection, boolean-based blind injection, and time-based blind injection.

3.2. Malicious behavior identification and monitoring

Web crawlers can be used to actively scan for vulnerability points and can also detect malicious behaviors in the network by analyzing access behaviors, having good application value in malicious crawler detection and security event correlation analysis, among other aspects.

The enhanced network intrusion detection system proposed by Eswaran et al. uses crawler technology to capture data from access log files, and extracts features such as access frequency, request interval, User-Agent field, and compliance with the robot’s protocol from the access logs. It then uses SVM models or decision tree models to judge the data to distinguish between normal crawlers and malicious crawlers. For example, malicious crawlers usually make a large number of irregular requests in a short period of time or continuously use a fixed User-Agent. If the relevant monitoring function of the system is triggered, anomalies can be quickly detected and corresponding early warnings can be issued, with an error rate of basically no more than 3%. In addition, through correlation analysis, the correlation between this abnormal behavior and possible security issues such as data leakage and DDoS attacks can be found, helping to trace the source.

In addition, crawlers can also be used to track the spread trajectory of malicious codes. By crawling dark web forums or malicious software sample libraries targeting specific objectives, they can obtain new malicious code features, such as hash values or behavior descriptions, and compare them with local threat intelligence databases to identify possible sources of infection and infected targets.

DOI: 10.54254/2755-2721/2025.LD28446

3.3. Information security monitoring

In terms of information security monitoring, web crawler technology is used to collect and analyze online data for public opinion management and monitoring of sensitive information leakage.

Hu Hongling believes that Python-based web crawlers can regularly crawl public online sources, such as social platforms, news websites, and forums. They can find the data to be crawled through keyword matching, such as combining enterprise names with keywords like "data leakage" and "vulnerability", and use natural language processing technology to analyze the scope of information diffusion and emotional tendencies, helping enterprises discover problems in a timely manner and take crisis management measures. In addition, crawlers can be used to monitor whether important information is leaked from websites. Crawlers will obtain sensitive content in the web page source code, then use regular expressions to identify it—for example, unt phone numbers and ID cards can be found by directly obtaining page text and matching with regular expressions such as the ID card format ^\d {17}[\dXx]$, and then inform the management for corresponding handling.

Given the dynamic network environment, it is more appropriate for crawlers to use incremental crawlers to only crawl the latest changes of previously crawled pages. For example, when crawling the "security announcements" of target websites, if new vulnerability warnings or patch contents are found, they can be directly pushed to the security team.

4. Technical challenges and solutions

4.1. Response to anti-crawler mechanisms

As crawler technology is increasingly applied in the field of network security, some websites have adopted anti-crawler technologies to ensure the security of their own data and the stability of the system. Common anti-crawler methods mainly include dynamic page rendering, captcha verification, IP blocking, and access frequency limitation.

To effectively crawl content from dynamic pages, tools like Selenium can be used to simulate browser behavior, run corresponding page scripts to obtain complete page information, which solves the defect that traditional crawlers can only obtain static HTML [9]. For detecting whether there are XSS vulnerabilities in dynamic forms on the page, Selenium can test the effectiveness of XSS form detection by simulating user input in the XSS forms [2].

For general captchas (graphic captchas, sliding verification), OCR (Tesseract) can be used to directly read simple graphic captchas; for complex verification, interfaces of coding platforms can be called for recognition. In addition, some rules can be manually set and the execution time can be extended (for example: 3-10 seconds) to make requests have a certain randomness, making human-like operations more similar to machine behavior.

For issues of IP blocking and access frequency limitation, an IP proxy pool can be established to avoid the risk of a single IP being blocked due to excessive access frequency. Additionally, methods such as randomly changing the User-Agent field (to simulate different browsers or devices) and disabling Cookie tracking can be used to make crawlers less likely to be detected. For example, Xiao Qiuping used a proxy pool and random requests to bypass the frequency limitation of the target site and detected more than 1,000 URLs for the SQL injection detection system.

DOI: 10.54254/2755-2721/2025.LD28446

4.2. Compliance and privacy protection

If illegal crawling technology is applied to network security detection, it will violate laws or users' privacy, so attention must be paid to compliance issues [1].

Legally, it is necessary to abide by the provisions of the Cybersecurity Law, the Data Security Law, and the target website's robots.txt protocol, and refrain from crawling restricted information (such as user privacy information, background management pages, etc.). Generally, search engine crawlers will actively read the robots protocol, impose certain restrictions on crawling, and not exceed their authority [6].

Regarding privacy protection, the principle of "data minimization" should be adhered to, collecting only data related to security detection (such as vulnerability URL, form parameters, etc.). For sensitive data crawled (such as user data collected during testing), encryption methods (AES algorithm) should be adopted, or the data should be directly destroyed and prohibited from being leaked to the outside. From the perspective of privacy, Eswaran et al. de-identified the data during data analysis to ensure that users' personal information would not be leaked during malicious crawler detection.

In addition, when detecting vulnerabilities in internal systems that require login, written authorization from the other party should be obtained first, informing them of the scope and purpose of the crawled data, to avoid legal risks.

5. Conclusion

To sum up, web crawler technology is an important means to enhance network security, applied in vulnerability detection, malicious behavior identification, and information security monitoring through automated crawling. Based on relevant research results, different types of crawlers (including focused crawlers, distributed crawlers, and enhanced crawlers) have diverse applications in various fields: focused crawlers and distributed crawlers can conduct targeted scans to detect XSS cross-site scripting vulnerabilities and SQL injection vulnerabilities respectively; enhanced intrusion detection systems (IDS), such as the one developed by Eswaran et al., use crawler behavior data to determine whether the detected crawlers are malicious; and incremental crawlers continuously monitor security threats and their impacts through long-term tracking of updated information.

The widespread application of crawlers faces many problems: anti-crawling measures like dynamic page rendering and IP blocking may drive the continuous update of crawler technology, such as adding Selenium to handle dynamic pages and using proxy pools for IP rotation; in addition, it is necessary to comply with legal and privacy protection norms, such as following the robots.txt protocol or minimizing data collection.

Looking ahead, integrating machine learning into crawler systems has broad prospects. This integration will use past crawling data to train models, enabling crawlers to automatically adjust URL priorities, adapt to more complex anti-crawling measures over time, and improve the detection capability of various attacks in the evolving attack environment. Moreover, developing a cross-platform security framework that integrates vulnerability detection, malicious activity identification, and real-time monitoring is expected to provide a comprehensive solution to improve the current fragmented protection methods. Meanwhile, further optimizing the lightweight crawler architecture will make it easier to deploy in edge computing environments, providing faster real-time security detection for distributed networks.

To fully unleash the great potential of web crawlers in protecting the Internet ecosystem and promoting the formation of a more stable and secure digital environment, people should address

DOI: 10.54254/2755-2721/2025.LD28446

both technical challenges and ethical issues when adopting advanced intelligent and innovative technologies.

References

[1] Hu, H.L. (2025). Application and Prevention Strategies of Python Web Crawler Technology in Information Security Monitoring. Information and Computer (Theoretical Edition), (12), 152-154.
[2] Liu, F.F. (2025). Research on the Application of Web Crawler Technology in XSS Vulnerability Detection. Automation Information Technology, 30(3), 20-22.
[3] Chai, A. (2017). Design and implementation of dynamic and efficient web crawler for XSS vulnerability detection. Advances in Engineering, 126, 1169-1176.
[4] Xiao, Q.P., Zhao, S.Q., Zhai, J.Q. (2017). Design of Automated SQL Injection Detection System Based on Web Crawler. Computer and Network, (23), 70-72.
[5] Eswaran, S., Rani, V., Daniel, D., Ramakrishnan, J., & Selvakumar, S. (2022). An enhanced network intrusion detection system for malicious crawler detection and security event correlations in ubiquitous banking infrastructure. International Journal of Pervasive Computing and Communications, 18(1), 59-78.
[6] Desai, K., Agrawal, S., Devulapalli, V., Kathiria, P., & Patel, A. (2017). Web crawler: Review of different types of web crawler, its issues, applications and research opportunities. International Journal of Advanced Research in Computer Science, 8(3), 1199-1202.
[7] Sheng, J., Xu, C., Zhou, T. (2024). Analysis of Python Web Crawler Technology. China Information Times, (09), 210-212.
[8] Ikerionwu, C., Nnadi, L. C., Okpala, I., Onuoha, M., Amadi, E. C., & Chukwudebe, G. (2020). A focused web crawler for strengthening cyber security and building a knowledge-based domain. In Proceedings of the International Conference on Emerging Applications and Technologies for Industry 4.0 (EATI 2020), 157-162.
[9] Cui, H.H. (2023). Research on Python-based Web Crawler Technology. Information Recording Materials, 24(6), 172-174.
[10] Wan, B., Xu, C., & Koo, J. (2023). Exploring the effectiveness of web crawlers in detecting security vulnerabilities in computer software applications. International Journal of Informatics and Information Systems, 6(2), 56-65.