WEP Warwick Evans Publishing

International Journal of Computer Science and Information Technology

ISSN: 3005-9682 (Print), ISSN: 3005-7140 (Online) | Volume 4, Number 2, Year 2024
DOI: https://doi.org/10.62051/ijcsit.v4n2.09
Journal homepage: https://wepub.org/index.php/IJCSIT/index

# Design and Implementation of a Web Attack Analysis System Based on Honeypot Technology

Shuqian Wang *

Department of Information Security, Zhengzhou University, Zhengzhou, China
*Corresponding Author: 18738984836@163.com

## ABSTRACT

Nowadays, with an increasing number of Web services being deployed in university network systems, Web attacks have become a significant threat due to the inherent vulnerabilities in Web services and the diverse forms of these attacks. Traditional Web attack defenses can only react passively, resulting in a lag in protection. Therefore, to enhance the security of Web systems, this paper develops a Web attack analysis system based on honeypot technology. Firstly, the system captures Web attack information through simulated honeypots. Secondly, it extracts and integrates effective data from the attack information to analyze hackers' attack methods and intentions. Thirdly, the analysis results are visualized through charts, providing data support for system maintenance. Finally, the feasibility of the system is verified through experiments.

## KEYWORDS

Honeypot Technology; Network Security; Web Honeypot; Web Attack

## 1. INTRODUCTION

With the rapid development of network technology, universities have fully embraced digitalization. However, due to weak network security awareness during the initial stages of network construction, many vulnerabilities remain, providing a foundation for hackers to launch attacks. Web attacks [1], with their unique characteristics, have become the preferred attack method for hackers, causing immeasurable losses to universities. How to safeguard campus network security has become a realistic network security issue faced by modern universities.

Honeypot technology [2] effectively compensates for the shortcomings of traditional defense methods, which can only defend passively, making it one of the hotspots for proactive defense.

In domestic research, scholar Zhen Chen [3] proposed simulating real networks using honeypot nodes and monitoring these nodes to extract attack data features. Scholar Yanyan Gao [4] suggested using a hybrid honeypot, where low-interaction honeypots serve as gateways for some high-interaction honeypots, forwarding attack traffic to the latter and using an improved ACSFCM algorithm for clustering analysis of the collected data. Scholars such as Weijie Mao [5] proposed deploying honeypots using Docker technology. Feng Jiang [6] proposed capturing encrypted traffic using the Sebek system and achieving automatic alerts through the Swatch tool. Multiple scholars have also suggested using redirection technology to connect specific ports or IP addresses to virtual honeypots, ensuring the normal operation of the system.

In foreign research, Scholars such as Artail [7] proposed using hybrid honeypots to enhance IDS capabilities. Yang and Mi [8] introduced the use of unsupervised clustering algorithms and genetic clustering algorithms to extract attack features. Scholars such as Buda [9] proposed addressing Web

application security issues by constructing Web application honeypots. Scholars such as Vasilomanolakis [10] presented TraCINg and TraCINg monitors for acquiring alarm data.

Combining the above research, this paper proposes a Web attack [11] analysis system based on honeypot technology. By analyzing attack behaviors in honeypots, it can directly feedback to the real system, defend against attacks before they occur, lock attackers, and improve the campus network's resistance to Web attacks.

## 2. SYSTEM DESIGN

The system primarily achieves three functions while ensuring its own security: recording attack behaviors through honeypots, extracting and integrating effective information from attack records [12], and visualizing analysis results. The corresponding system architecture is shown in the figure below.

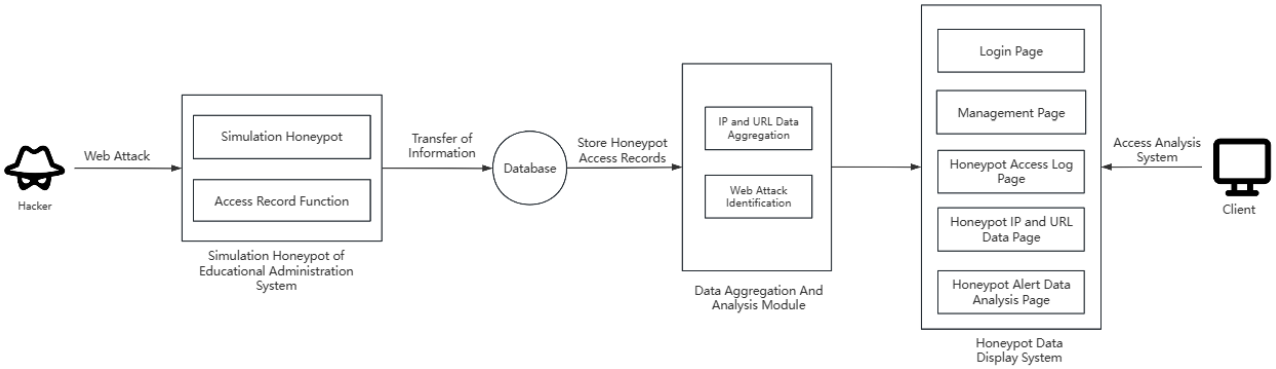


**Figure 1.** System Architecture Diagram

### 2.1. Design of Simulation Honeypot for the Educational Administration System

The simulation honeypot for the educational administration system is specifically designed for hackers. It first replicates the real educational administration system and adds fake sensitive information. Then, the honeypot is deployed under a subdomain of the educational administration system and exposed to hackers to attract attacks. Finally, all attack behaviors occurring in the honeypot are recorded in a database, forming attack logs.

During the setup process, it is crucial to ensure that hackers cannot compromise the honeypot and use it as a stepping stone to attack the real educational administration system. Normal users should not have access to the honeypot address to prevent accidental access. Any access to the honeypot is considered potentially malicious.

### 2.2. Design of Data Aggregation and Analysis Module

This module deeply analyzes and aggregates valuable data from the attack logs generated by the honeypot, realizing clustering and recognition of Web attacks, providing data support for the honeypot data display system.

### 2.3. Design of Honeypot Data Display System

The honeypot data display system visualizes the analyzed data through five pages:

Login Page: For access control, ensuring that only authorized personnel can view and analyze sensitive information.

Management Page: For navigating to other pages.

Honeypot Access Log Page: Displaying all attack records in the log database.

IP and URL Statistics Page: Integrating and displaying the TOP 10 IPs and URLs in charts.

Alert Page: Exhibiting a pie chart of the TOP 10 attack types, a curve chart of alert counts over time, and detailed data for each attack type.

### 2.4. Design of Database

Set the database file path to "../honeypot_log.db", create the database file referenced by DATABASE_FILE, and define the structure of a logs table. Then, using the with sqlite3. Connect (DATABASE_FILE) as conn statement, establish a connection to the SQLite database. Once the connection is open, call the conn.execute () function to create the logs table. Additionally, leverage the builtin logging library in Python to generate attack logs.

**Table 1.** Definition of the logs Table Structure (with id as the Primary Key)

| Name | Data Type | Description |
|---|---|---|
| id | INTEGER | Unique identifier for each record |
| username | TEXT | Stores the username submitted by the user |
| password | TEXT | Stores the password submitted by the user |
| headers | TEXT | Stores HTTP request header information in JSON string format |
| method | TEXT | Stores the HTTP request method (e.g., GET or POST) |
| URL | TEXT | Stores the full URL of the request |
| remote_addr | TEXT | Stores the IP address of the attacker |
| time | INTEGER | Stores the timestamp of the request |

## 3. SYSTEM IMPLEMENTATION

### 3.1. Implementation of Simulation Honeypot for Educational Administration System

#### 3.1.1. Construction of Simulation Honeypot

Firstly, render a page identical to a real educational administration system and utilize the jQuery library for image carousel to achieve dynamic effects consistent with a genuine educational administration system.

Next, call the checkLogin () function to validate the username and password. If the validation fails or either field is empty, display a login failure message. However, even if the validation succeeds, to maintain a high level of deception, the system will still display a login failure message.

Subsequently, differentiate and handle access requests to this page. Access via the GET method will return the login page, while access via the POST method will return a login failure page, ensuring that hackers are never able to successfully log in and thereby preventing them from obtaining further valid information.

Finally, customize 404 and 500 error handlers. When hackers attempt to access nonexistent pages, the 404 error handler is triggered, and when internal server errors occur, the 500 error handler is activated. Both of these will cause the system to execute a redirect operation back to the login page, ensuring that hackers cannot obtain more information about the honeypot by accessing other pages.


**Figure 2.** Simulation Honeypot Page

### 3.1.2. Implementation of Access Record Function

To define a before_request () function using the @app. before_request decorator that executes before every request, the function can be structured as follows. This function will accomplish the tasks as outlined in your requirements:

(1) Retrieve the current date and time using datetime. now ().

(2) Obtain the client's IP address (client_ip) and the full URL of the request.

(3) Determine whether or not to proceed with logging (though this step is somewhat abstract without a specific condition, it's assumed you have a way to decide).

(4) Retrieve the username and password from the request form using request. form. Get () and convert the request headers into a dictionary format. Record the HTTP method (GET or POST), the full URL, the IP address, and the timestamp, packaging them into a dictionary request_data.

(5) Call the log_request_to_db () function, passing the request_data dictionary as an argument to log the data into a database.

## 3.2. Implementation of Data Aggregation and Analysis Module

### 3.2.1. Implementation of IP and URL Data Aggregation

Define the handler_remote_addr () function for IP handling and the handler_url () function for URL handling, to achieve the statistics of the TOP 10 attacking IP addresses and accessed URLs.

### 3.2.2. Implementation of Web Attack Identification

Firstly, define multiple attack recognition functions using a set of regular expressions for various types of attacks. Then, in the att_ck_check() function, iterate through each attack recognition function and invoke the re.search() function to match suspicious content within the input data, enabling the classification of attack types. Simultaneously, store the attack data in corresponding lists.

**Table 2.** Regular Expression Table

| Attack Recognition Function | Regular Expression Sets | Function Description |
|---|---|---|
| sqli_check | select\|and<br>\bunion\b.*\bselect\b<br>'(?:\s*or\s*\|--<br>\|'\|=\|\s*(select\|insert\|update\|delete\|drop\|exec)\s*)<br>\bor\b\s+\d+\s*=\s*\d+<br>\bAND\b\s+[^\s]+\s*=\s*[^\s]+<br>\b(select\|insert\|update\|delete\|drop\|exec\|create\|alter\|cast<br>\|convert)\b<br>\b(benchmark\|sleep\|load_file\|outfile)\b | Detect SQL Injection Attack |
| xss_check | <\s*script[^>]*>.*?<\s*/\s*script\s*><br>on\w+\s*=\s*[\'"].*?[\'"]<br>javascript\s*:\s*[^\'"]*<br><\s*(iframe\|frame)[^>]*><br>\b(alert\|eval\|prompt\|confirm\|expression)\b\s*\(.*?\)<br><!--.*?-->\|<!\[CDATA\[.*?\]\]><br><\s*img[^>]*src\s*=\s*[\'"][^\'"]*[\'"]<br><\s*style[^>]*>.*?<\s*/\s*style\s*>\|style\s*=\s*[\'"].*?[\'"] | Detect XSS Attacks |
| cmd_inject | \|<br>;<br>&&\|\|\|<br>>\|<<br>`\|\$\(.*?\)\|\$\{.*?\}<br>\b(touch\|echo\|ping\|nc\|perl\|python\|sh\|bash\|cmd)\b<br>\b(rm\|ls\|cat\|wget\|curl\|chmod\|chown)\b | Detect Command Injection Attacks |
| file_inclusion_check | (file) | Detect File Inclusion Attacks |
| open_redirect_check | redirect | Detect Redirect Attacks |
| xxe_check | <!DOCTYPE\s+[^>]+>.*<!ENTITY\s+[^>]+><br><!ENTITY | Detect XML External Entity Attacks (XXE) |
| code_inject_check | (python\|ruby\|perl\|php\|java\|c#\|c\+\+)<br>(eval\|exec\|compile\|run) | Detect Code Injection Attacks |
| ssti_check | \{\{.*?\}\}<br>\{%.*?%\} | Detect SSTI Template Injection Attacks |
| java_deserialization_check | java\.io\.Serializable<br>java\.io\.ObjectInputStream<br>ObjectInputStream.*readObject | Detect Java Deserialization Attacks |

## 3.3. Implementation of Honeypot Data Display System

### 3.3.1. Implementation of Login Page

Using the Flask framework, define several key routes and view functions to handle operations such as the homepage, login, and logout.

Define the root route, where when a user accesses the root path "/", the index() view function redirects the request to the "login" path, forcing hackers to log in first.

Next, utilize the login () view function to handle the "/login" path by partitioning its behavior. Accessing the system through a GET request returns the login page. Accessing through a POST request prompts the function to perform login validation. If the validation passes, the system redirects to the admin page; if it fails, the system rerenders the login page.

Finally, define the "/logout" path and the corresponding logout() view function. When a user accesses this path, the system redirects back to the login page.

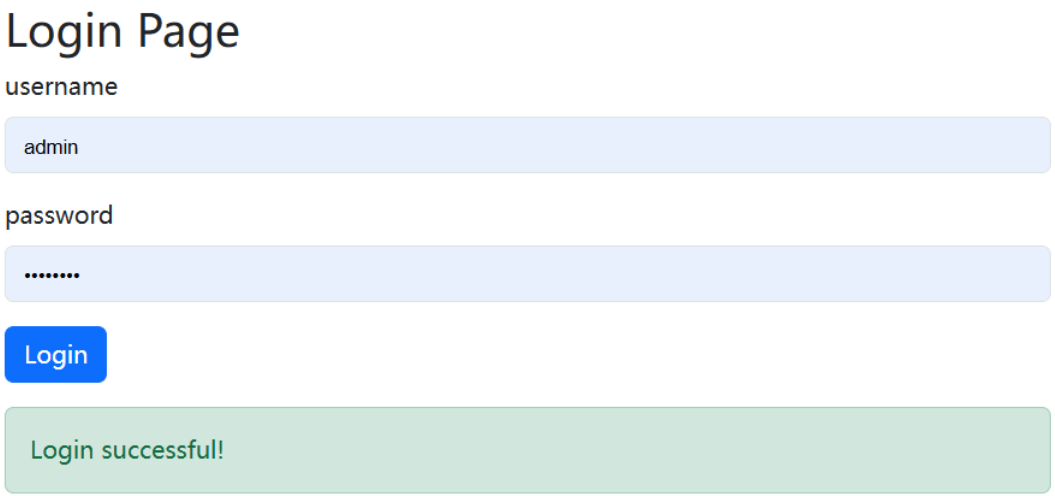


**Figure 3.** Login Page

### 3.3.2. Implementation of Management Page

This page achieves navigation to other pages through the use of redirect links in the form of "<a href="{{ url_for() }}" class="btn btnprimary">Page Name</a>", where url_for() is a function that generates URLs for the specified endpoint, allowing for dynamic page transitions within the admin page.

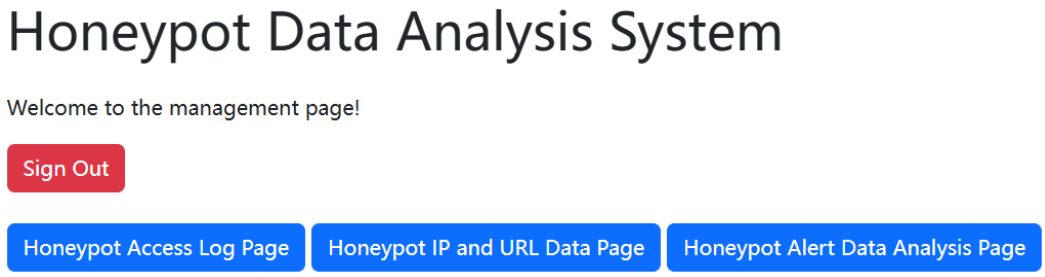


**Figure 4.** Administration Page

### 3.3.3. Implementation of Honeypot Access Log Page

First, define the "/logs" route to handle the display of log information. This route calls the get_db_connection () function to establish a connection with the database, executes a database query to retrieve log records ordered by id in descending order, retrieves all results using the fetchall () function, and finally calls the render_temPlate () function to render the page as depicted in Figure 5.

### 3.3.4. Implementation of Honeypot IP and URL Data Page

Firstly, define the "/ip_url" route and bind it to the handler function ip_url (). Then, create an Analysis object analysis_obj, passing the database path "../honeypot_log.db" as a parameter to this object. After that, call the analysis_obj. get analysis () function to extract IP and URL analysis data from the

database. Finally, invoke the render_template () function to render the page, as shown in Figure 6 below.

### 3.3.5. Implementation of Honeypot Alert Data Analysis Page

Firstly, define the "/attacks" route and bind it to the handler function attacks (). Then, create an Analysis object analysis_obj and call the analysis_obj. get_analysis () function to retrieve attack analysis data from the database, including attack content, request method, URL, remote address, and timestamp. Finally, invoke the render_template () function to render the page, as shown in Figure 7 below.

## 4. SYSTEM TEST

Test whether the system can achieve the expected results by simulating hacker attacks on the system.

**Table 3.** Attack Test Case Table

| Test ID | Test Functionality | Test Description | Expected Result | Passed |
|---|---|---|---|---|
| 01 | SQL Injection Detection | Input “and 1=1” into username or password field | Can analyze | Yes |
| 02 | XSS Attack Detection | Input “<script>alert("xss")</script>” into username or password field | Can analyze | Yes |
| 03 | Command Injection Detection | Input “cat /etc/passwd\|grep root” into username or password field | Can analyze | Yes |
| 04 | Inclusion Attack Detection | Input “file:///etc/passwd” into username or password field | Can analyze | Yes |
| 05 | Redirect Attack Detection | Input “/redirect?url=http://www.baidu.com” into navigation bar | Can analyze | Yes |
| 06 | XEE Attack Detection | Input ””<!ENTITY % file SYSTEM “file:///path/to/file'>” into username or password field | Can analyze | Yes |
| 07 | Code Injection Detection | Input “python exec("echo hello")” into username or password field | Can analyze | Yes |
| 08 | SSTI Template Injection Detection | Input “{{7*7}}” into username or password field | Can analyze | Yes |
| 09 | Java Deserialization Vulnerability Detection | Input “ObjectInputStream.readObject()” into username or password field | Can analyze | Yes |

## 4.1. Performance Analysis

Honeypot Access Log Page

Return to Management Page

Log Data

| ID | payload | Request Method | URL | IP | Time |
|---|---|---|---|---|---|
| 83 | | GET | http://127.0.0.1:5000/login.action | 127.0.0.1 | 2024-08-15 01:41:02 |
| 82 | | GET | http://127.0.0.1:5000/login.action | 127.0.0.1 | 2024-08-15 01:41:02 |
| 81 | | GET | http://127.0.0.1:5000/login.action | 127.0.0.1 | 2024-08-15 01:36:50 |
| 80 | | GET | http://127.0.0.1:5000/login.action | 127.0.0.1 | 2024-08-15 01:36:49 |
| 79 | | GET | http://127.0.0.1:5000/login.action | 127.0.0.1 | 2024-08-15 01:25:01 |
| 78 | ObjectInputStream.readObject() | POST | http://127.0.0.1:5000/login.action | 127.0.0.1 | 2024-08-15 01:25:01 |
| 77 | | GET | http://127.0.0.1:5000/login.action | 127.0.0.1 | 2024-08-15 01:24:56 |
| 76 | ObjectInputStream.readObject() | POST | http://127.0.0.1:5000/login.action | 127.0.0.1 | 2024-08-15 01:24:56 |
| 75 | | GET | http://127.0.0.1:5000/login.action | 127.0.0.1 | 2024-08-15 01:24:42 |
| 74 | {{7}} | POST | http://127.0.0.1:5000/login.action | 127.0.0.1 | 2024-08-15 01:24:42 |

1 2 3 4 5 6 7 8 9 next page

**Figure 5.** Honeypot Access Log Page

Through the analysis of log data, it is proven that our system is capable of capturing attack behaviors, and by parsing the log data, key information such as payload, request method, target URL, remote address, and attack time can be extracted.

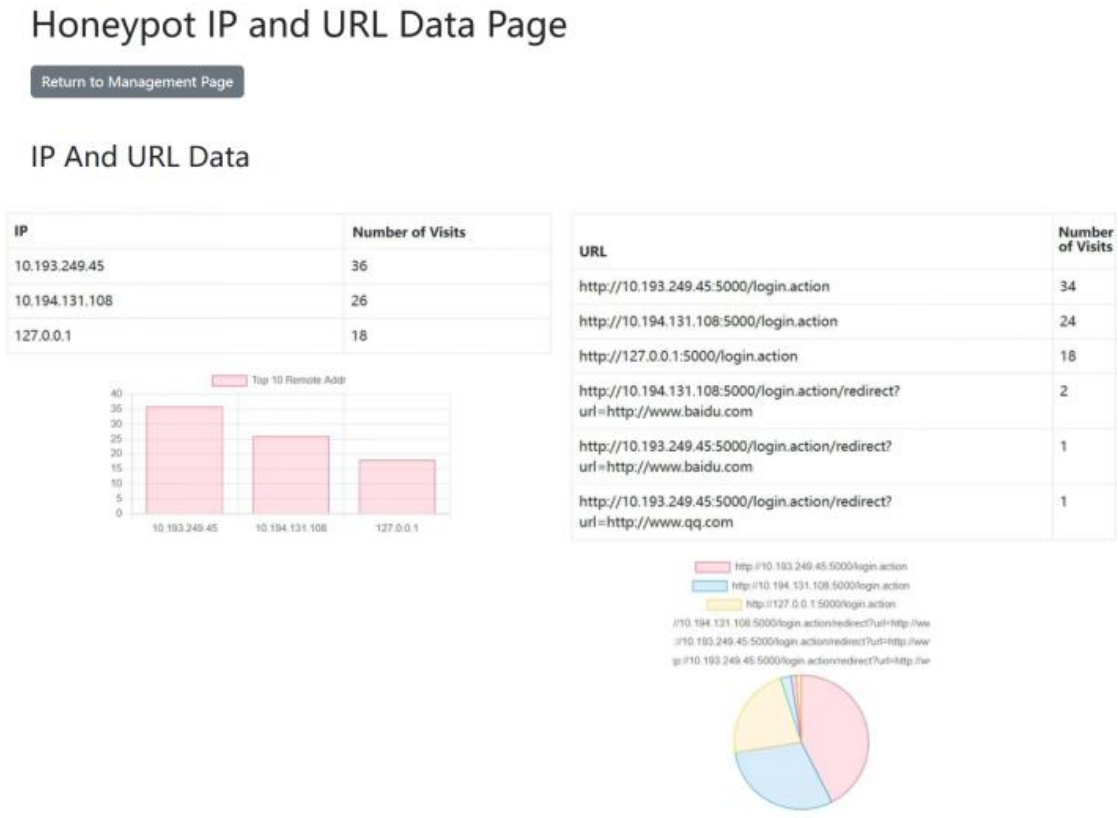


**Figure 6.** Honeypot Access Log Page

Through the analysis of IP and URL data, it is proven that our system can aggregate and classify data, and by logging IP addresses, security personnel can block high-frequency IP addresses and trace them. By logging URLs, security personnel can check if the corresponding URLs exist in the actual educational administration system and pay attention to whether these URLs contain 0-day or 1-day vulnerability information, thereby enhancing the security of the real system.

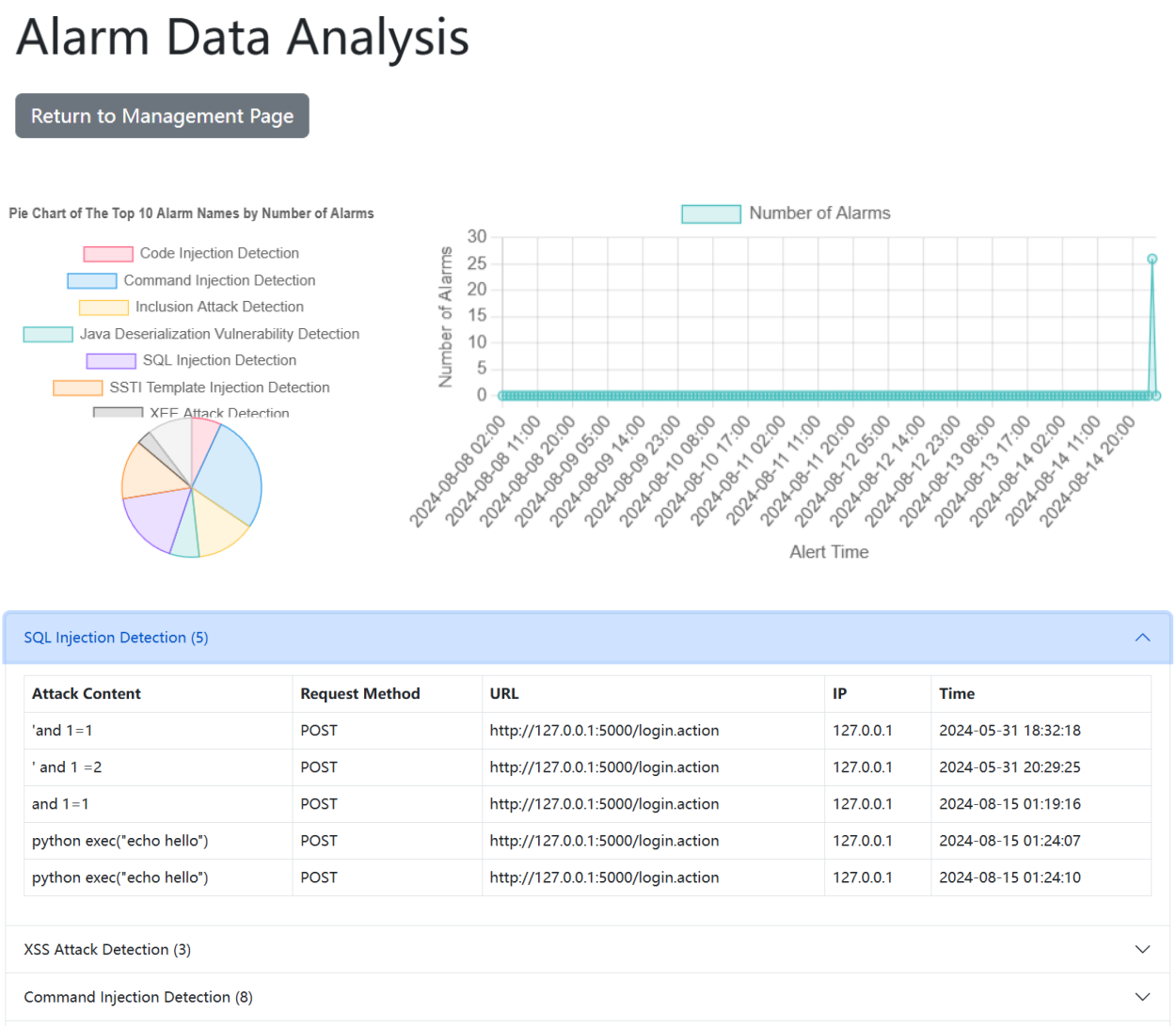


**Figure 7.** Alert Data Analysis Page

Through the analysis of alert data, it is proven that our system can identify and record different types of attacks. By correlating time points with the number of attacks to form a graph, it alerts security personnel to enhance monitoring of the educational administration system during periods of frequent attacks and speculate on the hackers' intentions, such as conducting numerous attacks due to interest in a specific resource within the system.

Through the detailed analysis of data from each attack type, we can provide guidance for network security maintenance and prevent hackers from attacking the actual educational administration system.

## 5. CONCLUSION

This paper primarily implements a simulation honeypot for an educational administration system, a data aggregation and analysis system, and a honeypot data display platform. By analyzing the current research status at home and abroad and the significance of developing this system, the main functions of the system were determined. Through testing, it was proven that the system achieved the expected

functions, providing a direction for the security maintenance of the educational administration system. As can be seen from the research in this paper, honeypot technology plays a significant role in network attack defense and deserves further research.

## REFERENCES


[1] Dan Wang, Wenbing Zhao, Zhiming Ding. Review of key technologies for common injection security vulnerability detection in web applications [j]. Journal of Beijing University of technology, 2016, 42 (12): 62-72

[2] Longjiang Wang. Research and implementation of campus network security system based on honeynet technology [d]. Anhui University, 2011

[3] Zhe Chen. Design and implementation of network attack early warning system in Colleges and Universities Based on Honeynet [j]. information and computer (theoretical Edition), 2023, 35 (19): 101-103

[4] Yanyan Gao. Research on Intrusion Detection Technology Based on hybrid honeypot [d]. Kunming University of technology, 2021

[5] Weijie Mao. Research and implementation of Intranet threat capture system based on Honeypot [d]. Jiangsu University of science and technology, 2021

[6] Feng Jiang. The implementation path of constructing campus network active defense system based on honeypot technology [j]. electronic technology and software engineering, 2017, (20): 197-198

[7] Wahid Rajeh.Hadoop Distributed File System Security Challenges and Examination of Unauthorized Access Issue [J]. Journal Information Security, 2022, 13(2):23-42

[8] Jermsittiparsert, Kittisak: Phuong, Nguyen Hoang: Hieu, Min VU. Editorial: Special. Issue on Language and Educational Administration [J]. World Journal of English Language, 2022, 12(3): 1-2

[9] Dokev N, Valeva V. SYSTEM FOR EDUCATIONAL PROCESSADMINSTRATION IN COMPUTER ROOMS AND LABS, 2022

[10] Sanjeev Kumar, Anil Kumar. Image-based malware detection based on convolution neural network with autoencoder in Industrial Internet of Things using Software Defined Networking Honeypot [J]. Engineering Applications of Artificial IntelLigence, 2024, 133:108374

[11] Xuezhang Liu, Qingjia Huang, Jing Xie, Xiaoqi Jia, Baoxu Liu. Research on Honeypot Technology for Web Security Protection [J]. Journal of Secrecy Science and Technology, 2021, (02): 28-33

[12] Wei Wan, Xin Shi, Jinxia Wei, Chang Li, Chun Long. Web Attack Detection Method Based on Stacking Fusion Model [J]. Journal of Information Security, 2024, 9(1): 84-94