Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2023.038268
Article

Dis-NDVW: Distributed Network Asset Detection and Vulnerability Warning Platform

Leilei Li[1], Yansong Wang[2], Dongjie Zhu[2,*], Xiaofang Li[3], Haiwen Du[4], Yixuan Lu[2], Rongning Qu[3] and Russell Higgs[5]

[1]Artificial Intelligence Academy, Wuxi Vocational College of Science and Technology, Wuxi, 214068, China
[2]School of Computer Science and Technology, Harbin Institute of Technology, Weihai, 264209, China
[3]Department of Mathematics, Harbin Institute of Technology, Weihai, 264209, China
[4]School of Astronautics, Harbin Institute of Technology, Harbin, 150001, China
[5]School of Mathematics and Statistics, University College Dublin, Dublin, DO4 V1W8, Ireland

*Corresponding Author: Dongjie Zhu. Email: zhudongjie@hit.edu.cn

Received: 05 December 2022; Accepted: 14 April 2023; Published: 09 June 2023

Abstract: With the rapid development of Internet technology, the issues of network asset detection and vulnerability warning have become hot topics of concern in the industry. However, most existing detection tools operate in a single-node mode and cannot parallelly process large-scale tasks, which cannot meet the current needs of the industry. To address the above issues, this paper proposes a distributed network asset detection and vulnerability warning platform (Dis-NDVW) based on distributed systems and multiple detection tools. Specifically, this paper proposes a distributed message subscription and publication system based on Zookeeper and Kafka, which endows Dis-NDVW with the ability to parallelly process large-scale tasks. Meanwhile, Dis-NDVW combines the RangeAssignor, RoundRobinAssignor, and StickyAssignor algorithms to achieve load balancing of task nodes in a distributed detection cluster. In terms of a large-scale task processing strategy, this paper proposes a task partitioning method based on First-In-First-Out (FIFO) queue. This method realizes the parallel operation of task producers and task consumers by dividing pending tasks into different queues according to task types. To ensure the data reliability of the task cluster, Dis-NDVW provides a redundant storage strategy for master-slave partition replicas. In terms of distributed storage, Dis-NDVW utilizes a distributed elastic storage service based on ElasticSearch to achieve distributed storage and efficient retrieval of big data. Experimental verification shows that Dis-NDVW can better meet the basic requirements of ultra-large-scale detection tasks.

Keywords: Distributed; network security; network asset detection; vulnerability warning

1 Introduction

With the rapid development of Internet technology, network security, as a prerequisite for the smooth development of Internet-related businesses, has always been a hot research content in computer-related fields [1–3]. For example, [1–3] discusses the hot issue of cybersecurity from three perspectives: Industry 4.0, public awareness of cybersecurity, and ethical discussions on cybersecurity. According to the data included in China's National Information Security Vulnerability Database [4], since 2014, the number of vulnerabilities discovered and included in the vulnerability database has been increasing year by year, and the increasing trend is obvious. Among them, the number of vulnerabilities included in the whole year of 2018 reached more than 20,000. Judging from the risk rating distribution of vulnerabilities shown in Fig. 1, among the vulnerabilities currently included, more than half of the vulnerabilities are moderately dangerous. High-risk vulnerabilities and ultra-risk vulnerabilities account for 21.08% and 14.85%, respectively. The number of dangerous vulnerabilities accounted for only 9.28%. According to the above data, it can be seen that the current network security problems are very serious. These loopholes involve all aspects of the Internet world, seriously endanger a country's politics, economy, military, and national life, and are not conducive to the healthy development of government or enterprise information construction [5,6].

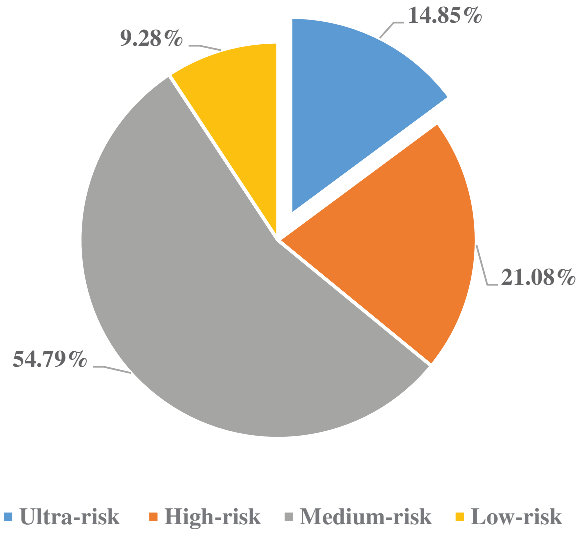


Figure 1: Vulnerability hazard level distribution map

A real-world entity (hereinafter referred to as an entity) associated with a network asset, such as a company, government agency, scientific research institute, etc., possesses or uses various network assets. In addition to regular network assets such as servers and routers, Internet of Things hardware devices such as access control with facial recognition functions [7] are becoming more and more common in the physical network asset list. The network assets of these entities are generally connected to the Internet in a wireless or wired manner, and provide entities with a variety of network service interfaces, including RESTful APIs [8], which brings multiple challenges to the security management of entity website assets. RESTful API is a style of web service architecture based on the HTTP protocol and standard HTTP methods (GET, POST, PUT, DELETE) to enable communication and interaction between distributed systems [8]. When an application or product deployed by an entity on a certain network device has certain security vulnerabilities, some criminals may use technical means to invade the entity's network asset control system. This will pose a serious threat to the data stored by the entity in the network assets or the system running on the network assets, and it may also cause immeasurable economic losses to the entity [9,10].

In response to the above problems, many network security engineers have launched a variety of network asset scanning and vulnerability detection tools. Among them, Nmap [11,12], Masscan [11], Wappalyzer [13], Whatweb [14], zgrab2 [15,16], and other scanning and detection tools for a specific problem are more typical, and there are more mature integrated scanning and detection tools such as Goby [17] and Vxscan [18]. As a relatively advanced and mature integrated scanning and detection tool in related fields, Goby has a relatively comprehensive scanning and detection function, which can better meet the needs of the above scenarios. However, when encountering large-scale physical assets that need to be scanned, scanning and detection tools such as Goby based on single-node scanning do not have parallelized large-scale task scanning and detection capabilities, and cannot provide accurate and effective scanning and detection results. Therefore, it is very necessary to build a network asset management and vulnerability detection system based on distributed clusters.

In summary, this paper has absorbed the advantages of integrated scanning and detection tools such as Goby, integrated the functions of existing scanning and detection tools such as Nmap [11,12], Masscan [11], Whatweb [14], zgrab2 [15,16], etc., and proposed a set of network asset management and vulnerability detection system based on the distributed cluster (Dis-NDVW). Specifically, it has the following advantages:

1. Highly available distributed system architecture. Dis-NDVW integrates distributed systems in task clusters and storage clusters. The availability and stability of the Dis-NDVW system are improved through multi-node redundant connection and redundant backup, and the parallelization level of Dis-NDVW processing tasks is also improved;
2. A tiered task subscription and publishing system supporting load balancing. This paper proposes a three-tier task subscription and publishing system based on Kafka and Zookeeper. The system implements a horizontal grouping of task clusters and a vertical hierarchy within task groups. At the same time, Dis-NDVW integrates three different scheduling algorithms, enabling Dis-NDVW to adaptively schedule pending tasks according to the load conditions of the working nodes in the cluster;
3. Complete system functionality. Expand the warning function of vulnerability information based on ensuring the integrity of the basic functions of the system. The system can scan and detect various network asset information such as active Internet Protocol (IP), active ports, port running applications, port flags, certificate information, website source code, vulnerability information, etc., with complete functions;

2 Related Work

This chapter will introduce the related content of network asset management and vulnerability detection tools from two aspects, namely: scanning detection tools for a specific problem and integrated scanning detection tools.

2.1 Scanning and Detection Tools for A Specific Problem

In terms of TCP port scanning and detection, the Masscan tool, which was open-sourced by Robert David Graham on GitHub in 2013, uses the form of asynchronously sending Synchronize Sequence Numbers (SYN) packets, which can scan and detect active IP and active ports for a given IP segment very efficiently. It is not possible to perform detailed scanning and detection of services running on the port [11]. Developed and maintained by http://nmap.org with Paulino Calderon, Gordon Fyodor Lyon, and Yang Luo as key members, the network scanning and sniffing toolkit Nmap (Network Mapper) is known for its accurate and detailed scanning results [11]. From the first launch of Nmap in September 1997 to the release of Nmap 7.92 on August 7, 2021, Nmap has become a mature scanning and detection tool after nearly two decades of development [12]. Nmap has excellent functions such as obtaining port services and versions, inferring the target host operating system, User Datagram Protocol/Transmission Control Protocol (UDP/TCP) port scanning, port script information discovery, ping scanning, etc., but the scanning efficiency is low and the scanning detection speed is slow [12,19]. Combining the above two detection tools can achieve good results in port scanning and detection: Masscan has high scanning efficiency but scans and detection results are not detailed, and Nmap has comprehensive scan results but low scanning efficiency.

In terms of Web site scanning and detection, WhatWeb [14,20], as an open-source website fingerprint identification software, can identify and analyze content management systems (CMS), blog platforms, statistics/analysis packages, JavaScript libraries, web servers, embedded devices. This tool can help us accurately identify and obtain the web technologies used by a given website. Zgrab2 is an application-layer network scanner written in the Go language. It can output detailed records of network handshakes (for example, all messages exchanged in Transport Layer Security (TLS) handshakes) for offline analysis [15,16]. Using zgrab2 can help us get some detailed information about the website's certificate and encryption algorithm. Although the above-mentioned scanning and detection tools can solve a certain aspect of scanning and detection requirements, they do not meet the needs of using distributed and multi-nodes to deal with large-scale scanning and detection tasks.

2.2 Integrated Scan Detection Tool

The integrated scanning and detection tool, represented by Goby proposed by Zwell et al. [17], has relatively comprehensive functional coverage, capable of scanning and detecting active IP, active ports, port services, application versions, etc., and has functions such as port service screenshots, vulnerability detection, and risk assessment. In terms of vulnerability detection, Goby has built a vulnerability framework based on the network security vulnerability community, which can filter the most valuable vulnerabilities from Common Vulnerabilities & Exposures (CVE) and other channels for daily updates. The coverage of vulnerabilities includes the most serious vulnerabilities such as Weblogic and Tomcat. Although Goby can meet most of the scanning and detection requirements and management function requirements, it still exists as a single-node scanning and detection client and cannot scale horizontally and vertically. Vxscan [18] is a comprehensive scanning detection tool based on the Python programming language. It already has the main scanning detection functions such as port scanning, website Web Application Firewall (WAF) fingerprinting, and Structured Query Language (SQL) injection. However, it still does not support multi-node expansion to meet the needs of large-scale scanning probes.

In order to meet the needs of large-scale scanning and detection, some scholars have also introduced distributed technology into the scanning and detection of network assets. Hua et al. [21] designed a distributed vulnerability detection system based on Client/Server (C/S) architecture, but only used multi-threading technology to deal with large-scale scanning and detection tasks. This will not be able to achieve better performance under stress tests. Hu [9] designed a relatively complete distributed vulnerability detection system, which has good functionality and ease of use, but the event-driven task scheduling mode based on the Twisted framework cannot meet the needs of large batches. Tian et al. [22] applied machine learning technology to distributed vulnerability scanning and detection, but the short-term message throughput of RabbitMQ, which Tian et al. chose as the intermediary for task message transmission, was lower than that of Kafka, which this paper chose. Therefore, it is slightly inferior when dealing with ultra-large-scale tasks.

The above-mentioned scanning and detection systems/tools have relatively complete scanning and detection functions, and in order to cope with various problems, they also try to introduce distributed technology to improve the system's parallel processing capability for scanning and probing tasks, but they cannot quickly give accurate and effective scanning and probing results for high-volume tasks. Therefore, it is very necessary to build a network asset management and vulnerability detection system based on distributed clusters.

3 Dis-NDVW: Distributed Network Asset Detection and Vulnerability Warning Platform

This chapter will explain in detail the design of the network asset management and vulnerability detection system based on distributed clusters this paper proposed. First, this chapter will give a general introduction to the architecture of the entire system, and then introduce the design of each functional module in the system in detail. Finally, the scheduling algorithm used in the Dis-NDVW system is described.

3.1 System Overall Architecture Design

As shown in Fig. 2, the Dis-NDVW system adopts the Browser/Server (B/S) architecture and provides system services to users in the form of web pages constructed by the Vue framework. The Web Server written in Python language acts as the server to act as the system service interface provider and publish task information. The three roles of the searcher and the searcher of the result information. At the same time, the Dis-NDVW system uses a message subscription and publishing system based on Zookeeper [23] and Kafka [24] to implement the distributed task publishing and consumption process of the system. Kafka is a high-throughput, low-latency, distributed publish-subscribe messaging system that can be used to handle large-scale, real-time data streams [24]. Zookeeper is an open-source distributed orchestration service that provides efficient configuration management, naming services, state synchronization, and distributed locking for distributed applications [23]. Kafka uses Zookeeper as its distributed orchestration service to manage cluster metadata, coordinate topic partition allocation and monitor the health of the Kafka cluster [24]. In terms of distributed storage, the Dis-NDVW system uses distributed elastic storage services based on ElasticSearch [25] to realize distributed storage and efficient retrieval of big data. The overall workflow of the Dis-NDVW system can be roughly summarized as the following steps:

Step 1: The user publishes scanning and detection tasks through the Web page;

Step 2: The web page transmits the scanning and detection task information filled in by the user to the Web Server. After receiving the information, the Web Server records it in the Database, and at the same time sends the instruction information of the publishing task to the Kafka message publishing system;

Step 3: According to the allocation of the scheduling algorithm, the task information released by the Kafka Producer will be allocated to a Consumer Group in the Kafka Consumer Group Cluster for processing (Chapter 3.2);

Step 4: Consumers of different task types in the Consumer Group will write the scan detection results into the ElasticSearch Cluster after completing the tasks according to the set index rules (Chapter 3.3);

Step 5: The web server can report the completion of tasks to users through the Web based on the records in the Database. When the user needs to query the results of the task, the Web Server queries and scans the detection results from the ElasticSearch Cluster and returns it to the Web for a display to the user;

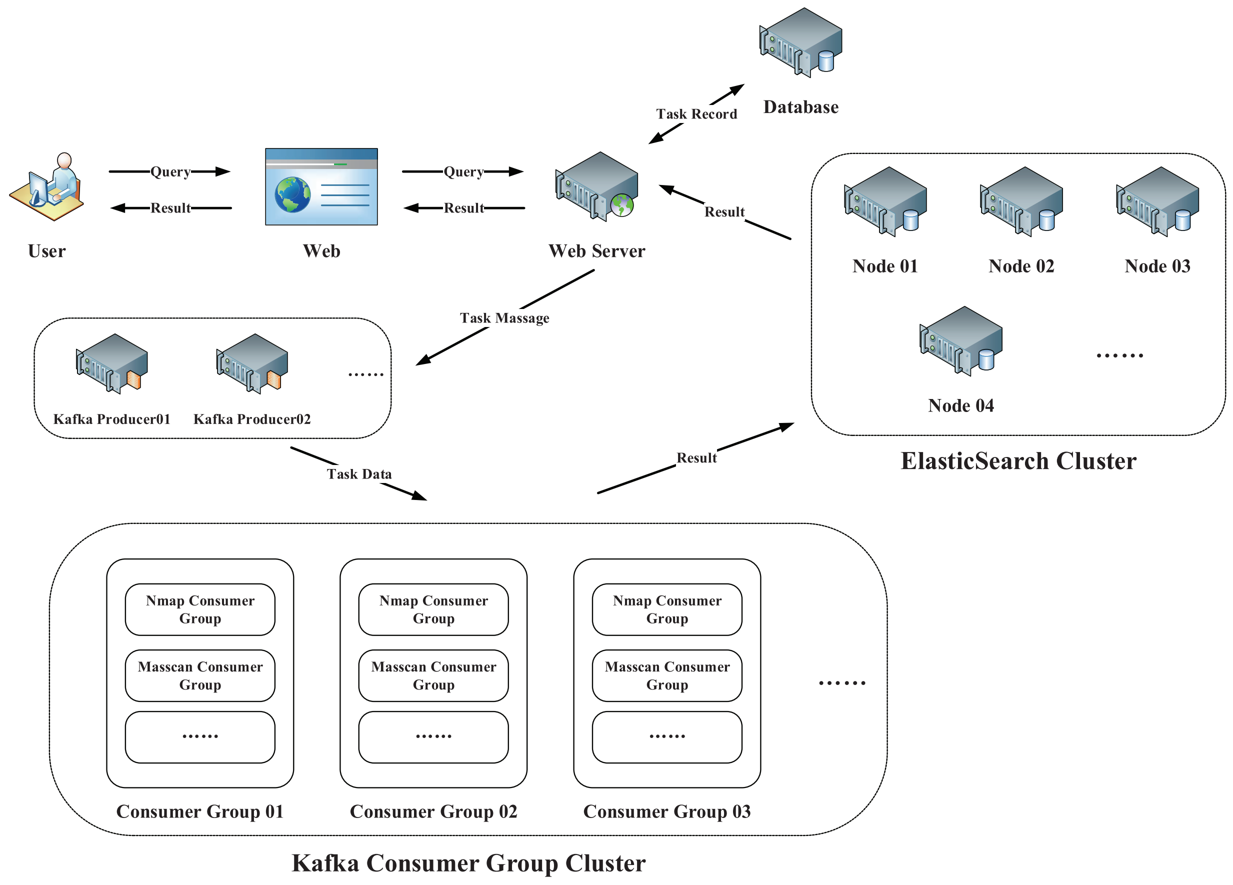


Figure 2: Dis-NDVW system overall architecture diagram

3.2 Distributed Message Subscription and Publishing

This section will give a detailed introduction to the Zookeeper & Kafka-based distributed message subscription and publishing system in the Dis-NDVW system, including overall architecture design, task flow design for IP scanning, and task flow design for domain scanning.

3.2.1 Architecture Design of Distributed Message Subscription and Publishing System

As shown in Fig. 3, the entire distributed message subscription and publishing system is roughly divided into three layers: Kafka Producer Group, Task Group Consumer, and Task Group. This paper proposes a three-tier task subscription and publishing system based on Kafka and Zookeeper (as shown in Fig. 3). The system implements the horizontal grouping of task clusters while layering them vertically within task groups. The strategy of horizontal grouping ensures that Dis-NDVW is parallelized at the task level. Specifically, different horizontal task groups can handle various large scan detection tasks simultaneously. On the other hand, the vertical hierarchical strategy within task groups ensures that Dis-NDVW can maximize parallelism between sub-tasks when executing individual scan detection tasks. The tasks undertaken by each layer of the system are described in detail at the end of this sub-section.

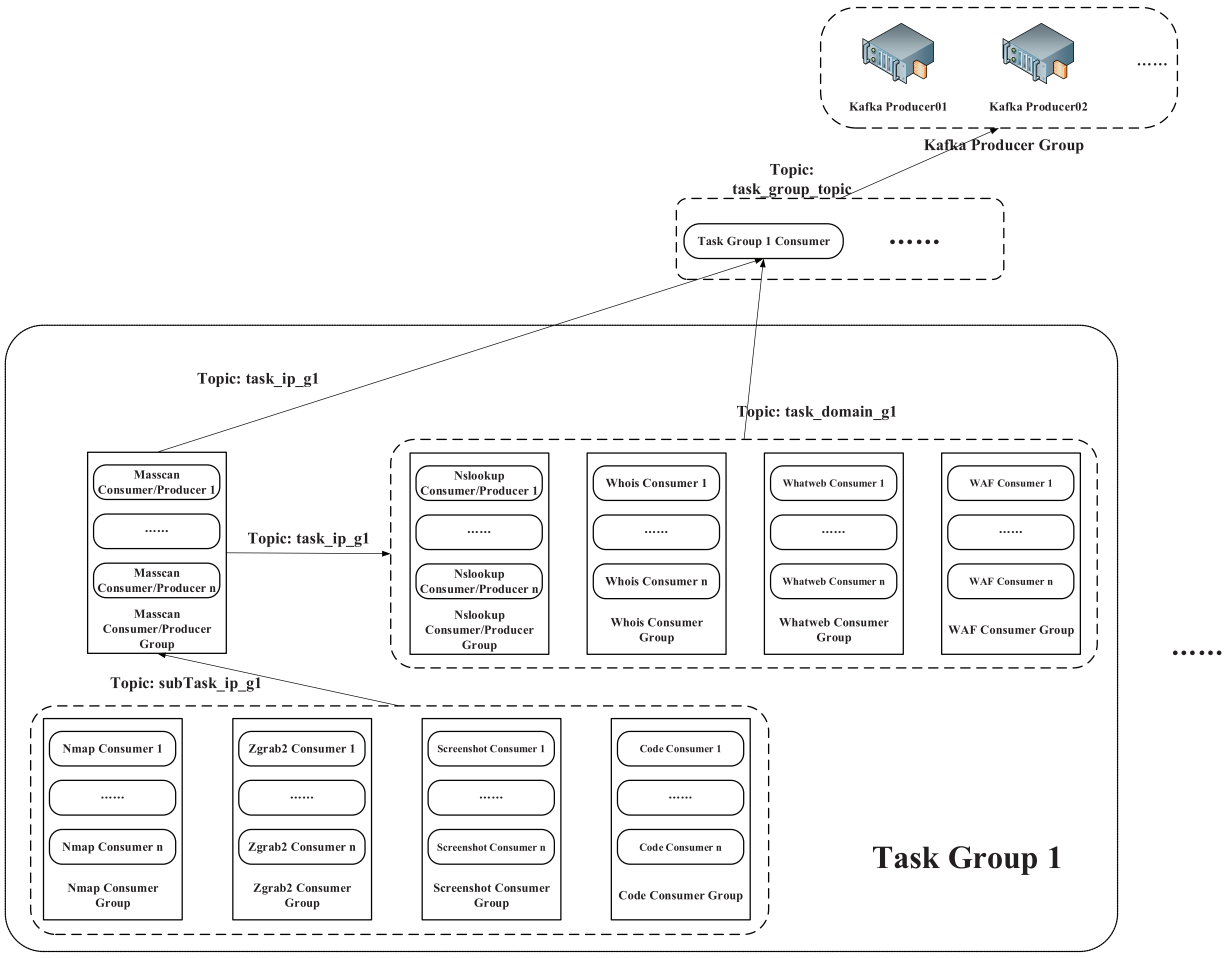


Figure 3: Distributed message subscription and publishing system architecture design diagram

The Kafka Producer Group at the top is an important middle layer that connects the Web Server and the distributed cluster. There are several Kafka Producers in the Kafka Producer Group, and each Kafka Producer is directly connected to the upper Web Server. This redundant connection method ensures the reliability and stability of the communication between the Web Server and the distributed cluster.

The Task Group Consumer group located between the Kafka Producer Group and Task Group is the top consumer of the entire Kafka consumer group. There are several consumer instances in the Task Group Consumer, and the number is the same as the number of Groups in the lower-level Task Group. When the Task Group 1 Consumer in the Task Group Consumer receives the assigned task information from the upper Kafka Producer, it immediately releases the task information to the lower Task Group under its jurisdiction, namely Task Group 1, to perform specific scanning and detection tasks.

It is important to note that any consumer in the Task Group Consumer group can publish two topic tasks. When consumers in the Task Group Consumer group receive the task information allocated from the upper Kafka Producer, they will parse the task information content. After the task type (IP Task/Domain Task) is judged, two tasks of different topic types, task_ip_g1, and task_domain_g1, will be released respectively.

The Task Group at the bottom is a collection of Kafka Consumers responsible for completing specific tasks. There are nine different types of small consumer groups in the collection, which are responsible for completing nine different types of scanning and detection tasks such as Masscan, Nslookup, Whois, and Whatweb. The number of consumer instances in each small consumer group is at least 1, the number of small consumer groups contained in each task group is 9, and the number of task groups contained in the Dis-NDVW system is at least 1. The specific task flow design in the Task Group will be introduced in detail below.

3.2.2 Task Flow Design for IP Scanning

The task flow design for IP scanning is shown in Fig. 4. After receiving the task information of the task_ip_g1 topic type, the Masscan Consumer/Producer Group in the Task Group first performs a Masscan scan to obtain the active IP status and active port status in a given IP (segment). When the consumer instances in the Masscan Consumer/Producer Group complete the Masscan scan, they will be converted to Kafka Producer, and they want to publish task messages of the subTask_ip_g1 topic type to the lower-level tasks.

The lower-level Nmap Consumer Group, Zgrab2 Consumer Group, Screenshot Consumer Group, and Code Consumer Group all subscribe to the subTask_ip_g1 topic and perform four different scanning and detection tasks in parallel after receiving the task message of the subTask_ip_g1 topic type issued by the Masscan Consumer/Producer Group. Among them, Nmap Consumer Group is responsible for Nmap scanning tasks, using the active IP and active port information scanned by Masscan to perform detailed port information detection and operating system inference; Zgrab2 Consumer Group uses the active IP information scanned by Masscan to obtain certificates and encryption algorithms; Screenshot Consumer Group is responsible for the acquisition of website screenshots, adopts the strategy of traversal attempts on the active IP and active ports scanned by Masscan, and tries to obtain screenshots for each active port of each active IP; Code Consumer Group is responsible for the acquisition of website source code. Like the Screenshot Consumer Group, the consumer instances in the Code Consumer Group also use the strategy of traversal attempts when acquiring the source code of the webpage, and trying to obtain the source code of the webpage for each active port of each active IP.

After each of the above consumer instances completes the scan detection, in addition to writing the scan detection result to ElasticSearch, it also needs to update the task progress record information in the Database.

3.2.3 Task Flow Design for Domain Scanning

The task flow design of the domain-oriented scanning task is shown in Fig. 5. First, the four small consumer groups responsible for domain name scanning tasks in the Task Group will perform four tasks: Nslookup query, Whois information query, Whatweb information query, and WAF information query after receiving task information of task_domain_g1 topic type. Among them, the IP information related to the target domain name can be obtained through the Nslookup query, so that the next five scanning and detection tasks can be performed. Whatweb information query helps us to obtain the service and version used by the target domain name website, and the WAF information query helps us to obtain the firewall information of the target domain name website. It is important to note that the above four tasks are completely parallelized.

When the Nslookup scan detection task is completed, the consumer instance in the Nslookup Consumer/Producer Group will be converted to Kafka Producer, and the task_ip_g1 topic task information will be released to the Masscan Group. The remaining task flow design is consistent with the task flow design for IP scanning, and will not be repeated here.

In summary, the task flow of domain-oriented scanning tasks is divided into three major steps: "parallel-serial-parallel": First, the four unique scanning and detection tasks of domain name scanning and detection are carried out in parallel; Secondly, perform Masscan scanning and detection tasks serially; The last four tasks related to IP scan detection, Nmap, Zgrab2, Screenshot, and Code, were performed in parallel. This task flow design achieves task parallelization to the greatest extent while conforming to the task flow logic, and achieves the goal of distributed parallelization for large-scale tasks.

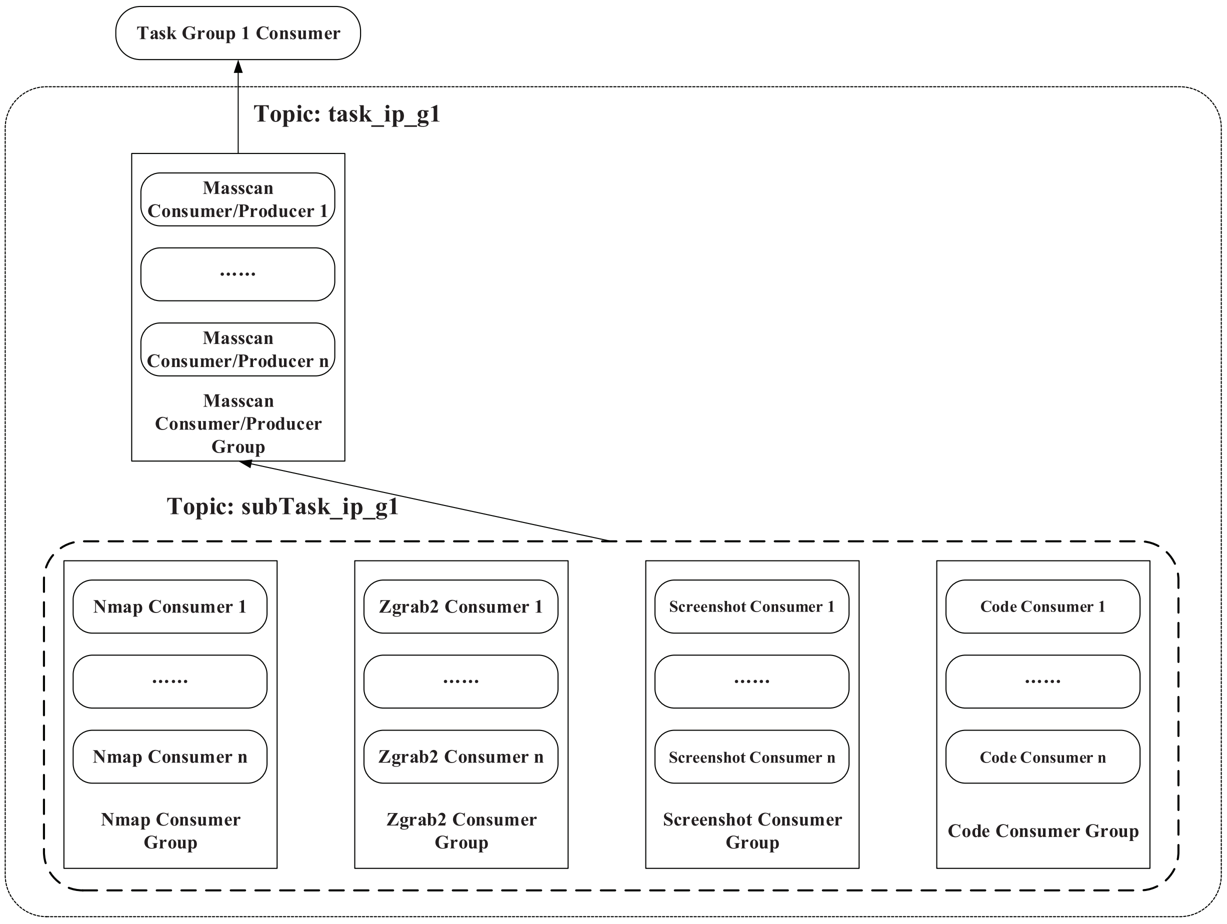


Figure 4: Task flow design for IP scanning

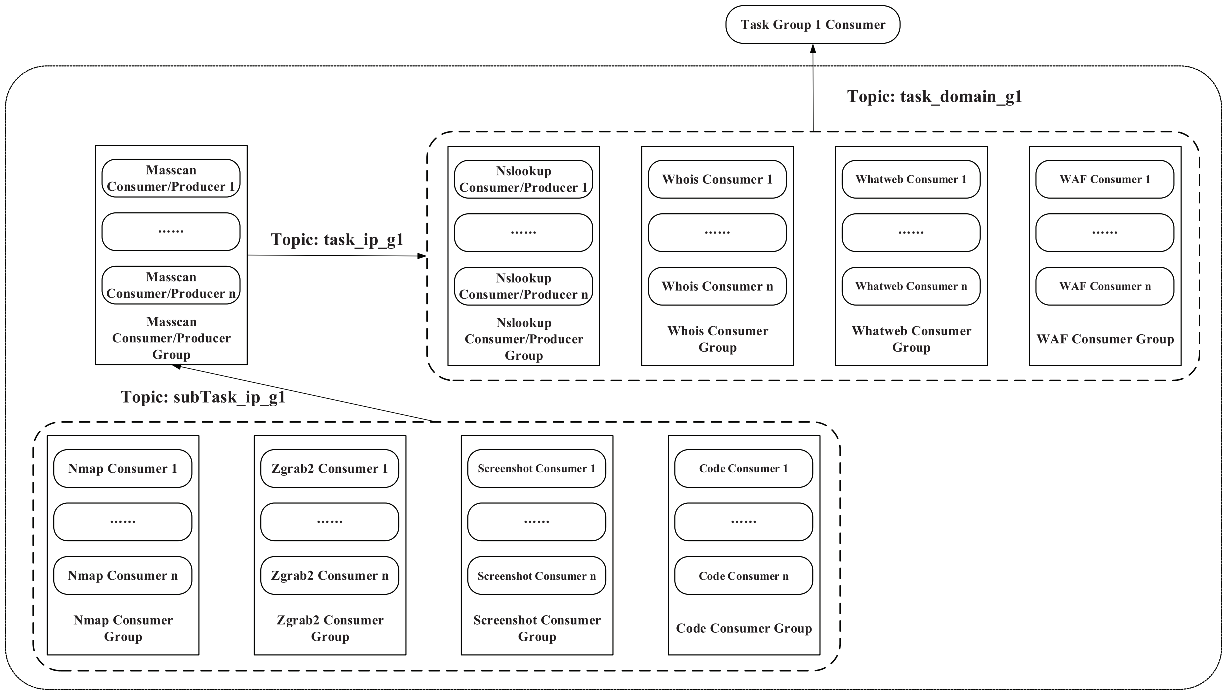


Figure 5: Task flow design for domain scanning

3.3 Distributed Elastic Access to Massive Data

As shown in Fig. 6, for the massive data generated by Dis-NDVW scanning and detection, this paper designs and proposes a distributed elastic access model for massive data. There is at least one node in the model, the specific number of nodes can be customized according to the actual number of servers, and nodes can be dynamically added according to the actual situation.

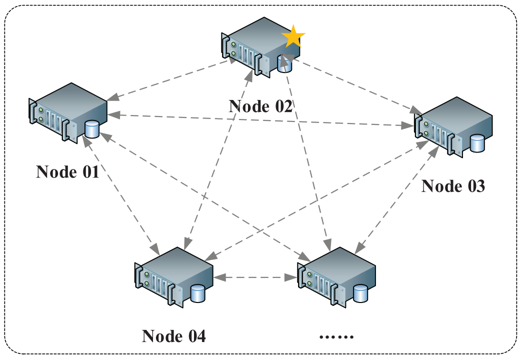


Figure 6: Distributed elastic access model for massive data

There are three different types of nodes in the cluster: client nodes, master nodes (Node 02 in Fig. 6), and data nodes. Among them, the client node mainly acts as a load balancer, and can only handle routing requests, processing searches, distributing index operations, etc.; The main responsibilities of the master node are content related to cluster operations, such as creating or deleting indexes, tracking and collecting the status of each node in the cluster, and the allocation of shards;

The main function of the data node is storage, and the storage of all document data in the cluster is completed by the data node.

In order to ensure the efficiency and availability of the distributed cluster, the index created by the user in the cluster will have multiple replica shards in addition to the primary shard. When the data in the primary shard is damaged or lost, it can be recovered from the replica shards. At the same time, the existence of the replica shards provides conditions for load balancing of cluster query operations.

3.4 Distributed Cluster Scheduling Algorithm Design

As shown in Fig. 7, this section outlines the design and algorithm definition of a distributed cluster scheduling algorithm for large-scale network asset scanning and detection tasks, including topics and partitions, task message writing and consumption, and cluster message reliability guarantee. Dis-NDVW combines the RangeAssignor, RoundRobinAssignor, and StickyAssignor algorithms to achieve load balancing of task nodes in a distributed detection cluster. The three different scheduling algorithms can adaptively schedule pending tasks according to the load situation of the working nodes in the cluster. Specifically, the RangeAssignor and RoundRobinAssignor algorithms solve the problem of allocating task information partitions for different topics in different situations: The former treats the different topics as separate message queues from each other, while the latter unifies all the partitions under all the different topics for sorting before allocating and scheduling partitions and consumer instances. The StickyAssignor algorithm is characterized by a partition reallocation policy when the number of instances in a group of consumers subscribed to a topic changes. Specifically, when the number of consumer instances in a group of consumers subscribed to a topic changes, the StickyAssignor algorithm minimizes the difference from the previous allocation while first satisfying a balanced partition allocation between different topics as far as possible. As a result, the algorithm reduces unnecessary resource consumption caused by reallocation and is more conducive to maintaining the stability of the cluster.

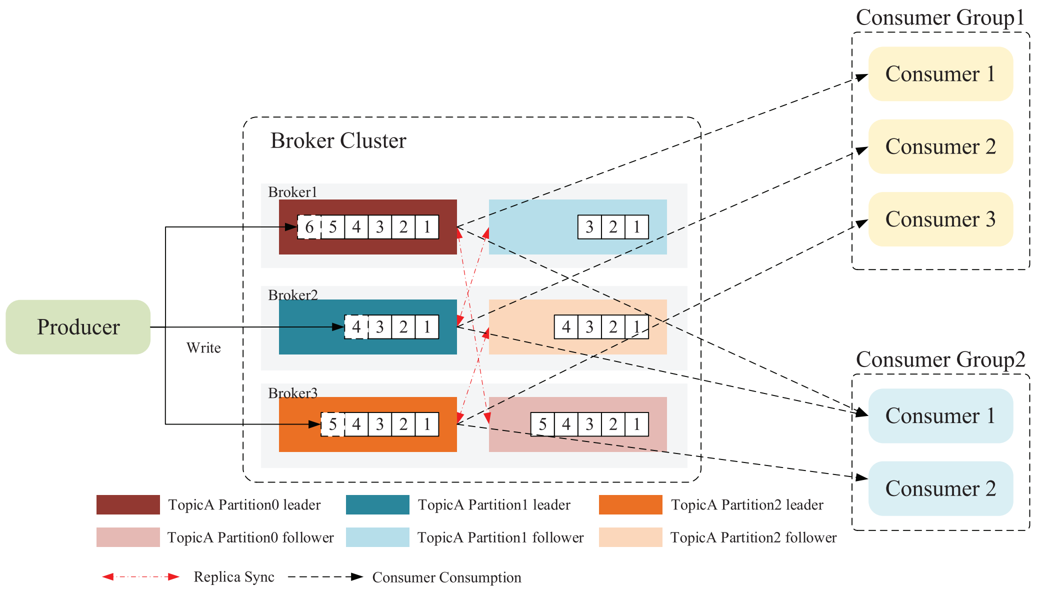


Figure 7: Schematic diagram of the distributed cluster scheduling algorithm

3.4.1 Topics and Partitions

In order to ensure that different task information is not confused and missed, this paper defines the message persistent queue between the message producer and the consumer as a task topic model. Specifically, for a certain message type/message format, there is a certain information persistent queue corresponding to it, which is called a topic (Topic). There are a certain number of topic partitions (Partition) under each topic, and these topic partitions theoretically span all physical nodes of the entire distributed message cluster (as shown in Broker Cluster in Fig. 7). As shown in the rectangles of different colors in the dotted frame of Broker Cluster in Fig. 7, the rectangles of the same color represent the master-slave copies of the same topic partition. Rectangular boxes of different colors represent three different topic partitions (Partition 0/1/2) under Topic A, and these partitions are evenly distributed among all nodes of the entire consumer cluster.

3.4.2 Task Message Writing and Consumption

As shown in Fig. 7, the pull mode in the message subscription and release mode is adopted in the distributed cluster scheduling algorithm adopted in this paper. After the producer writes a pending message to the topic it subscribes to, the consumers in the consumer cluster subscribing to the topic will consume messages from the corresponding leader partitions according to the "First-In-First-Out (FIFO)" strategy. It should be noted that since the partitions under the same topic are located on different physical nodes, the order of message consumption in the entire cluster cannot be guaranteed, but the order of message processing within the same partition can be guaranteed to comply with the "FIFO" strategy.

In addition, as shown in the dotted line of consumer group consumption on the right side of Fig. 7, in order to avoid repeated consumption of consumer data, the algorithm definition does not allow two consumers in the same consumer cluster to be responsible for processing messages in the same topic partition but allows the same consumer to be responsible for processing messages in two different topic partitions.

3.4.3 Cluster Message Reliability Guarantee

In order to ensure the reliability and stability of the task data stored in each node in the distributed cluster, a distributed cluster scheduling algorithm for large-scale network asset scanning and detection tasks is given a master-slave partition copy redundancy storage strategy. As shown in Fig. 7, there are master-slave copies of topic partitions represented by blocks of the same color in different Broker nodes. For example, Topic A Partition0 leader/follower are located in Broker1 and Broker3 respectively. It is only necessary to maintain the consistency of the stored data between the master and slave topic partitions. All data read and write operations will only interact with the leader copy in the master and slave partition copies (including asynchronous write and asynchronous read operations). The slave topic partition copy only needs to perform data interaction with the master topic partition copy to ensure the data consistency of the copy (as shown by the dotted red double-headed arrow in Fig. 7).

4 Experiments

This chapter will explain in detail the deployment and testing of Dis-NDVW.

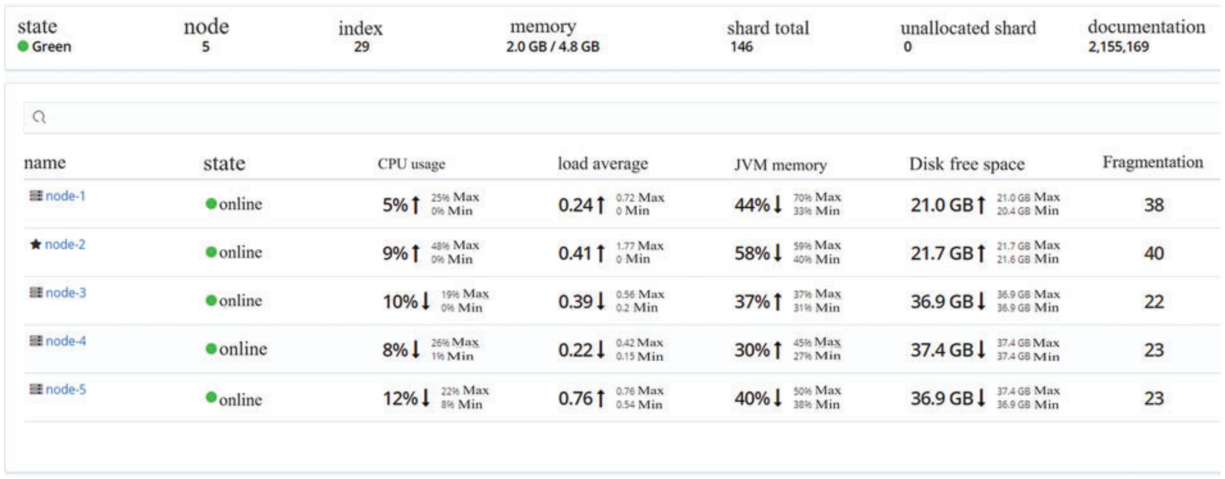
4.1 Experimental Environment

In order to test the ability of Dis-NDVW to deal with mass scanning and detection tasks, a cluster of five server nodes was built in this paper. The specific parameters are shown in Table 1.

Table 1: Machine configuration information in the experiment

Server	CPU	Memory	Disk	OS	Application
01	4 Intel Xeon Processor (Cascadelake) @2.9 GHz	8 GB	40 GB	Ubuntu 20.04 LTS	ES-Master, Kafka-Broker, Task Group 1
02	4 Intel Xeon Processor (Cascadelake) @2.9 GHz	8 GB	40 GB	Ubuntu 20.04 LTS	ES-Data, Kafka-Broker, Task Group 2
03	4 Intel Xeon Processor (Cascadelake) @2.9 GHz	8 GB	40 GB	Ubuntu 20.04 LTS	ES-Data, Kafka-Broker, Task Group 3
04	4 Intel Xeon Processor (Cascadelake) @2.9 GHz	8 GB	40 GB	Ubuntu 20.04 LTS	ES-Data, Kafka-Broker, Task Group 4
05	4 Intel Xeon Processor (Cascadelake) @2.9 GHz	8 GB	40 GB	Ubuntu 20.04 LTS	ES-Data, Kafka-Broker, Task Group 5

Among them, the real-time monitoring data of the distributed mass data elastic access module can be viewed through the Kibana visualization platform, as shown in Fig. 8.



state	node	index	memory	shard total	unallocated shard	documentation
● Green	5	29	2.0 GB / 4.8 GB	146	0	2,155,169

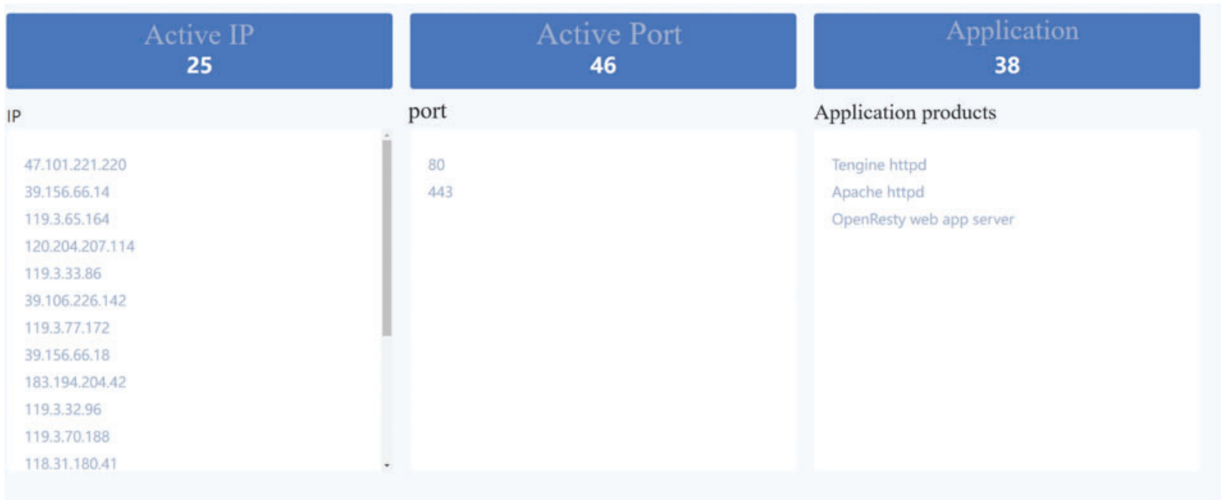
name	state	CPU usage	load average	JVM memory	Disk free space	Fragmentation
node-1	● online	5% ↑ 25% Max 0% Min	0.24 ↑ 0.72 Max 0 Min	44% ↓ 70% Max 33% Min	21.0 GB ↑ 21.0 GB Max 20.4 GB Min	38
★ node-2	● online	9% ↑ 48% Max 0% Min	0.41 ↑ 1.77 Max 0 Min	58% ↓ 59% Max 40% Min	21.7 GB ↑ 21.7 GB Max 21.6 GB Min	40
node-3	● online	10% ↓ 19% Max 0% Min	0.39 ↓ 0.56 Max 0.2 Min	37% ↑ 37% Max 31% Min	36.9 GB ↓ 36.9 GB Max 36.9 GB Min	22
node-4	● online	8% ↓ 26% Max 1% Min	0.22 ↓ 0.42 Max 0.15 Min	30% ↑ 45% Max 27% Min	37.4 GB ↓ 37.4 GB Max 37.4 GB Min	23
node-5	● online	12% ↓ 22% Max 8% Min	0.76 ↑ 0.76 Max 0.54 Min	40% ↓ 50% Max 38% Min	36.9 GB ↓ 37.4 GB Max 36.9 GB Min	23

Figure 8: Distributed massive data elastic access module real-time monitoring data

4.2 Dis-NDVW Basic Function Experiment

4.2.1 Dis-NDVW Basic Data Acquisition

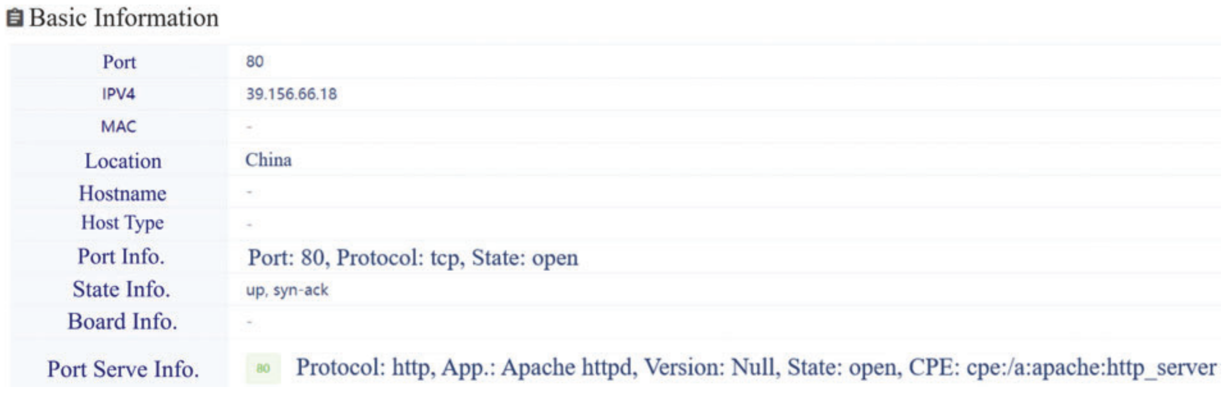
In this paper, "www.baidu.com, www.csdn.net, www.cnblogs.com, www.bilibili.com, mp.weixin.qq.com" was used as a test case for the domain scan detection task, and the overall result shown in Fig. 9 was obtained after the task was completed. As shown in Fig. 9, Dis-NDVW obtained a total of 25 active IP data, 46 active port data, and 38 different application product information in this task.



IP	port	Application products
47.101.221.220	80	Tengine httpd
39.156.66.14	443	Apache httpd
119.3.65.164		OpenResty web app server
120.204.207.114		
119.3.33.86		
39.106.226.142		
119.3.77.172		
39.156.66.18		
183.194.204.42		
119.3.32.96		
119.3.70.188		
118.31.180.41		

Figure 9: Summary of Dis-NDVW scan results

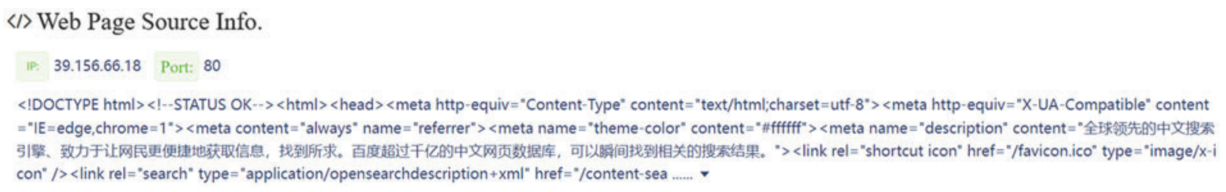
For a specific IP, Dis-NDVW can obtain various information as shown in Figs. 10 and 11, including open active ports and services, Whatweb, Whois information, web page source code, and web page screenshots. This is very necessary for sorting out network assets.



Basic Information

Port	80
IPV4	39.156.66.18
MAC	-
Location	China
Hostname	-
Host Type	-
Port Info.	Port: 80, Protocol: tcp, State: open
State Info.	up, syn-ack
Board Info.	-
Port Serve Info.	80 Protocol: http, App.: Apache httpd, Version: Null, State: open, CPE: cpe:/a:apache:http_server

Figure 10: Dis-NDVW scan detection results-basic information


</> Web Page Source Info.

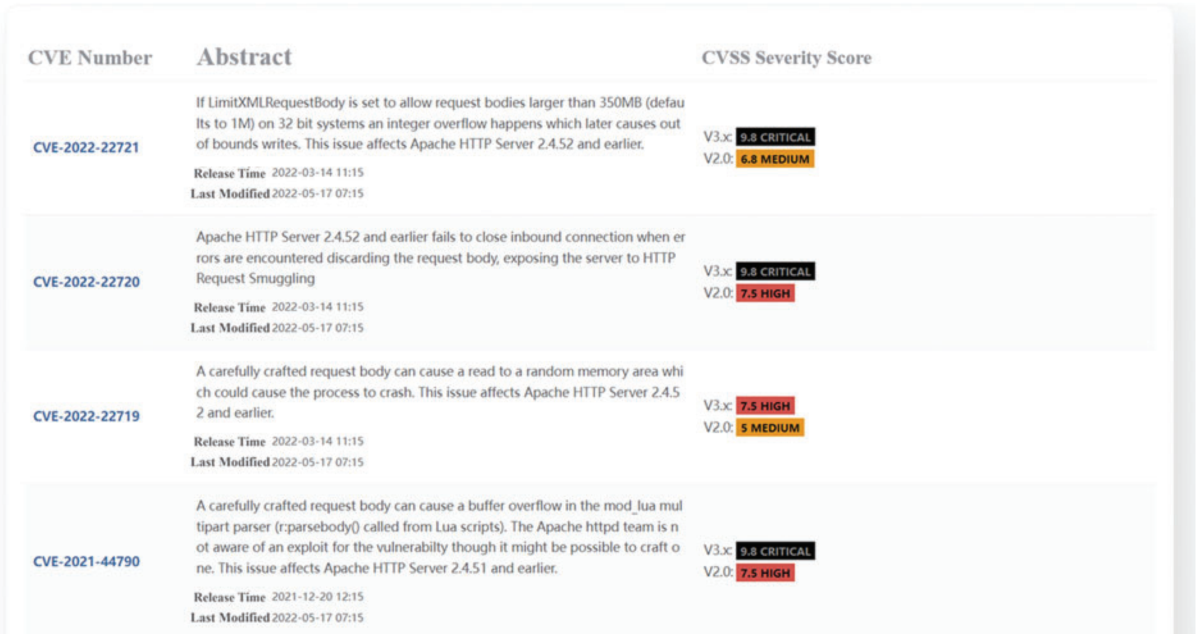
IP: 39.156.66.18 Port: 80

```
<!DOCTYPE html><!--STATUS OK--><html><head><meta http-equiv="Content-Type" content="text/html;charset=utf-8"><meta http-equiv="X-UA-Compatible" content
="IE=edge,chrome=1"><meta content="always" name="referrer"><meta name="theme-color" content="#ffffff"><meta name="description" content="全球领先的中文搜索
引擎、致力于让网民更便捷地获取信息，找到所求。百度超过千亿的中文网页数据库，可以瞬间找到相关的搜索结果。"><link rel="shortcut icon" href="/favicon.ico" type="image/x-i
con" /><link rel="search" type="application/opensearchdescription+xml" href="/content-sea ......
```

Figure 11: Dis-NDVW scan detection results-webpage source code information

4.2.2 *Dis-NDVW Vulnerability Information Detection*

As shown in Fig. 12, Dis-NDVW conducts possible vulnerability information query and retrieval based on the basic information obtained in Section 4.2.1 and displays the queried vulnerability information to users in the form of web pages. Specifically, Dis-NDVW uses the name of the product being run on the port and its version number obtained earlier as query keywords to query the relevant detailed vulnerability information from the National Vulnerability Database (NVD) interface[1]. The NVD interface is a free interface for querying vulnerability information provided by National Institute of Standards and Technology (NIST). The NVD supports querying by vulnerability-related software and its version number as keywords, and the query results will be returned in descending order of when the vulnerability was reported. We have taken the latest 5 relevant vulnerability messages to display on the Dis-NDVW page. Users can click on the CVE hyperlink to obtain detailed information about the vulnerability, such as the solution, the affected software version, and so on.



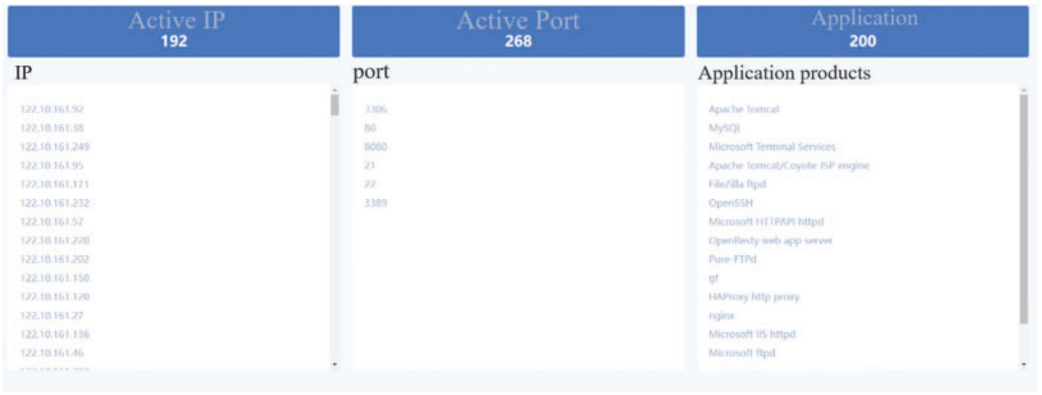
CVE Number	Abstract	CVSS Severity Score
CVE-2022-22721	If LimitXMLRequestBody is set to allow request bodies larger than 350MB (defaults to 1M) on 32 bit systems an integer overflow happens which later causes out of bounds writes. This issue affects Apache HTTP Server 2.4.52 and earlier. Release Time 2022-03-14 11:15 Last Modified 2022-05-17 07:15	V3.x: 9.8 CRITICAL V2.0: 6.8 MEDIUM
CVE-2022-22720	Apache HTTP Server 2.4.52 and earlier fails to close inbound connection when errors are encountered discarding the request body, exposing the server to HTTP Request Smuggling Release Time 2022-03-14 11:15 Last Modified 2022-05-17 07:15	V3.x: 9.8 CRITICAL V2.0: 7.5 HIGH
CVE-2022-22719	A carefully crafted request body can cause a read to a random memory area which could cause the process to crash. This issue affects Apache HTTP Server 2.4.52 and earlier. Release Time 2022-03-14 11:15 Last Modified 2022-05-17 07:15	V3.x: 7.5 HIGH V2.0: 5 MEDIUM
CVE-2021-44790	A carefully crafted request body can cause a buffer overflow in the mod_lua multipart parser (r:parsebody() called from Lua scripts). The Apache httpd team is not aware of an exploit for the vulnerabilty though it might be possible to craft one. This issue affects Apache HTTP Server 2.4.51 and earlier. Release Time 2021-12-20 12:15 Last Modified 2022-05-17 07:15	V3.x: 9.8 CRITICAL V2.0: 7.5 HIGH

Figure 12: Dis-NDVW scan detection results-vulnerability information detection

[1] https://nvd.nist.gov/vuln/data-feeds

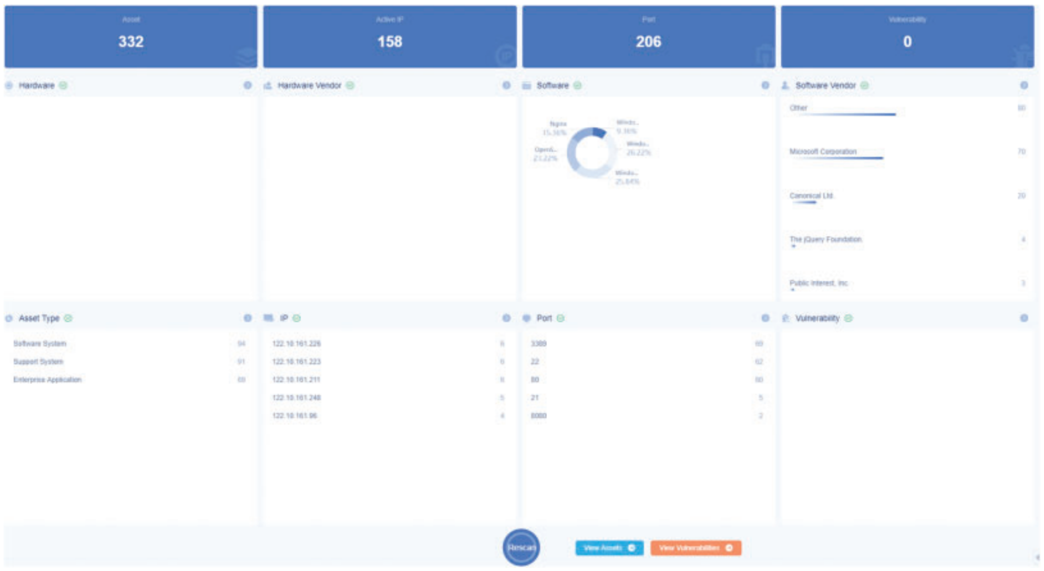
4.3 Large-Scale Scanning Task Experiment

As shown in Fig. 13, the IP segment "122.10.161.0/24" was used as an example to test Dis-NDVW's ability to cope with the large-scale scanning tasks. Using "122.10.161.0/24" as an experimental sample does not seem to meet the experimental requirements for ultra-large-scale network asset probing. However, according to our survey, a scan detection task equivalent to the size of an IP segment can meet the practical application requirements of network asset detection and management at this stage. Therefore, sample "122.10.161.0/24", which is closest to the actual application scenario, is used as the test case in this paper. Based on the results shown in Fig. 13, Dis-NDVW was able to complete the given scanning task relatively quickly, obtaining a total of 192 active IPs, 268 active ports, and 200 application products running on each port. The same experimental use case was scanned using Goby [17] and the results are shown in Fig. 14. Comparing with Fig. 13, Dis-NDVW captured more active IPs and active ports, and the scanning time was the same and the coverage of the scanning results was the same.



IP	port	Application products
122.10.161.92	3306	Apache Tomcat
122.10.161.38	80	MySQL
122.10.161.249	8080	Microsoft Terminal Services
122.10.161.95	21	Apache Tomcat/Coyote JSP engine
122.10.161.171	22	FileZilla ftpd
122.10.161.232	3389	OpenSSH
122.10.161.57		Microsoft HTTPAPI httpd
122.10.161.220		OpenResty web app server
122.10.161.202		Pure-FTPd
122.10.161.150		gf
122.10.161.120		HAProxy http proxy
122.10.161.27		nginx
122.10.161.136		Microsoft IIS httpd
122.10.161.46		Microsoft ftpd

Figure 13: Dis-NDVW large-scale scanning task experiment result



Software Vendor	
Other	80
Microsoft Corporation	70
Canonical Ltd.	20
The jQuery Foundation	4
Public Interest, Inc.	3

Asset Type	
Software System	94
Support System	91
Enterprise Application	69

IP	
122.10.161.226	6
122.10.161.223	6
122.10.161.211	6
122.10.161.248	5
122.10.161.96	4

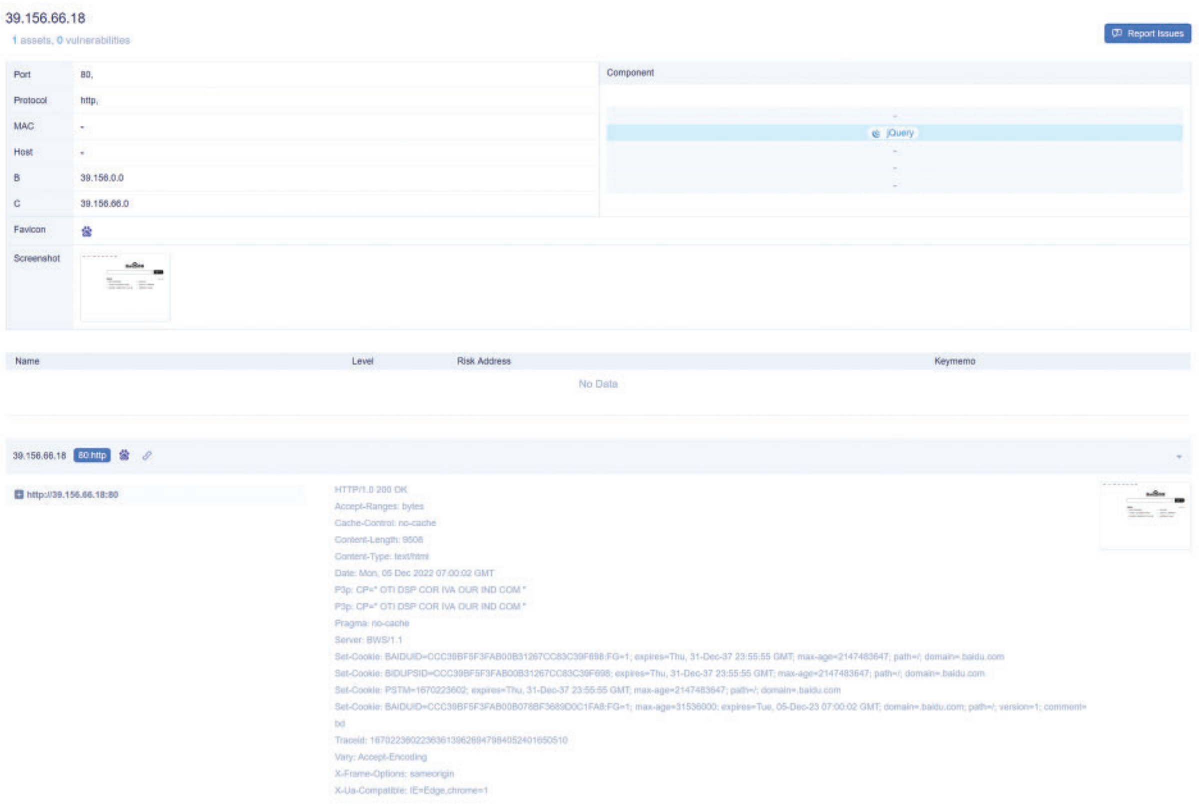
Port	
3389	69
22	62
80	60
21	5
8080	2

Figure 14: Goby large-scale scanning task experiment result

4.4 Comparison of Dis-NDVW Scan Results

4.4.1 Compare with Goby

Based on the same task instance “39.156.66.18”, this section compares and analyzes the results obtained by Dis-NDVW (as shown in Figs. 10–12) and those obtained by Goby [17] (as shown in Fig. 15). The purpose is to examine the breadth of the content covered by the scanning results of Dis-NDVW compared with the current mature integrated scanning platform Goby.



39.156.66.18
1 assets, 0 vulnerabilities

		Component
Port	80,	
Protocol	http,	-
MAC	-	jQuery
Host	-	-
B	39.156.0.0	-
C	39.156.66.0	-
Favicon		
Screenshot		

Name	Level	Risk Address	Keymemo
No Data			

39.156.66.18 80:http

http://39.156.66.18:80

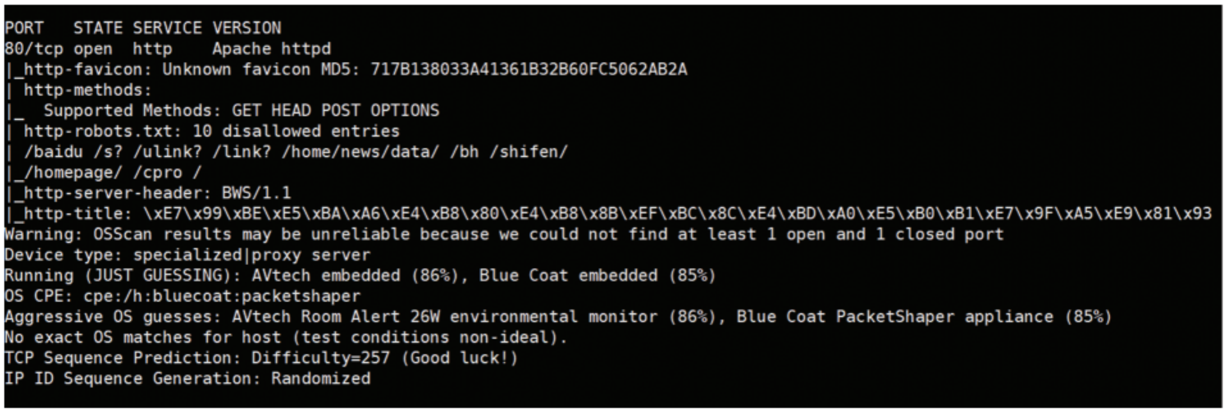
```
HTTP/1.0 200 OK
Accept-Ranges: bytes
Cache-Control: no-cache
Content-Length: 9508
Content-Type: text/html
Date: Mon, 05 Dec 2022 07:00:02 GMT
P3p: CP=" OTI DSP COR IVA OUR IND COM "
P3p: CP=" OTI DSP COR IVA OUR IND COM "
Pragma: no-cache
Server: BWS/1.1
Set-Cookie: BAIDUID=CCC39BF5F3FAB00B31267CC83C39F698:FG=1; expires=Thu, 31-Dec-37 23:55:55 GMT; max-age=2147483647; path=/; domain=.baidu.com
Set-Cookie: BIDUPSID=CCC39BF5F3FAB00B31267CC83C39F698; expires=Thu, 31-Dec-37 23:55:55 GMT; max-age=2147483647; path=/; domain=.baidu.com
Set-Cookie: PSTM=1670223602; expires=Thu, 31-Dec-37 23:55:55 GMT; max-age=2147483647; path=/; domain=.baidu.com
Set-Cookie: BAIDUID=CCC39BF5F3FAB00B078BF3689D0C1FA8:FG=1; max-age=31536000; expires=Tue, 05-Dec-23 07:00:02 GMT; domain=.baidu.com; path=/; version=1; comment=
bd
Traceid: 1670223602236361396269479840524016505l0
Vary: Accept-Encoding
X-Frame-Options: sameorigin
X-Ua-Compatible: IE=Edge,chrome=1
```

Figure 15: Example of Goby scan results

By comparing the above two scan results, we can conclude the following. (1) Dis-NDVW’s scan results completely override Goby’s scan results and are extended in terms of web page source code, vulnerability warning alerts, and IP address location acquisition. (2) Compared to Goby’s client-side model, Dis-NDVW’s B/S system architecture has better ease of use. Users can log in to Dis-NDVW via a browser to post scan detection tasks and view scan detection results from anywhere, anytime, on different devices without the need to install any additional applications.

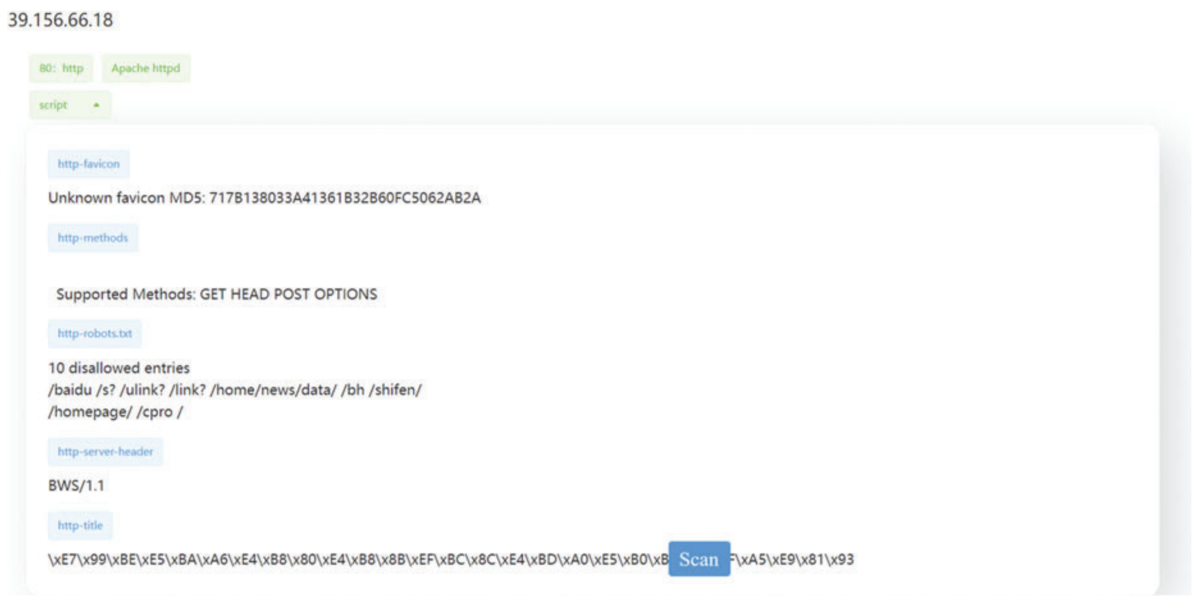
4.4.2 Compare with Nmap and Masscan

This section tests the coverage of Dis-NDVW on Nmap scanning results, and the experimental results are shown in Figs. 16 and 17. In order to ensure the fairness of the experiment, for the same experimental instance “39.156.66.18”, this paper sets the scanning parameters of Dis-NDVW and Nmap to “-v -sV -T4 -O -sC” in the same system environment for the experiment. According to the experimental results shown in Figs. 16 and 17, it can be considered that: Dis-NDVW completely covers the main content of Nmap scanning results, realizes the correct and complete function integration of Nmap, and has higher readability.



```
PORT   STATE SERVICE VERSION
80/tcp open  http    Apache httpd
|_http-favicon: Unknown favicon MD5: 717B138033A41361B32B60FC5062AB2A
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
| http-robots.txt: 10 disallowed entries
| /baidu /s? /ulink? /link? /home/news/data/ /bh /shifen/
|_/homepage/ /cpro /
|_http-server-header: BWS/1.1
|_http-title: \xE7\x99\xBE\xE5\xBA\xA6\xE4\xB8\x80\xE4\xB8\x8B\xEF\xBC\x8C\xE4\xBD\xA0\xE5\xB0\xB1\xE7\x9F\xA5\xE9\x81\x93
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: specialized|proxy server
Running (JUST GUESSING): AVtech embedded (86%), Blue Coat embedded (85%)
OS CPE: cpe:/h:bluecoat:packetshaper
Aggressive OS guesses: AVtech Room Alert 26W environmental monitor (86%), Blue Coat PacketShaper appliance (85%)
No exact OS matches for host (test conditions non-ideal).
TCP Sequence Prediction: Difficulty=257 (Good luck!)
IP ID Sequence Generation: Randomized
```

Figure 16: Nmap scan results for "39.156.66.18"



39.156.66.18

Unknown favicon MD5: 717B138033A41361B32B60FC5062AB2A

Supported Methods: GET HEAD POST OPTIONS

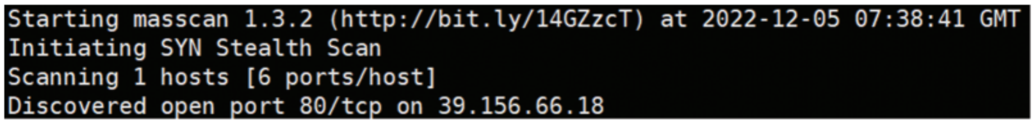
10 disallowed entries
/baidu /s? /ulink? /link? /home/news/data/ /bh /shifen/
/homepage/ /cpro /

BWS/1.1

\xE7\x99\xBE\xE5\xBA\xA6\xE4\xB8\x80\xE4\xB8\x8B\xEF\xBC\x8C\xE4\xBD\xA0\xE5\xB0\xB F\xA5\xE9\x81\x93

Figure 17: Dis-NDVW scan results for "39.156.66.18"

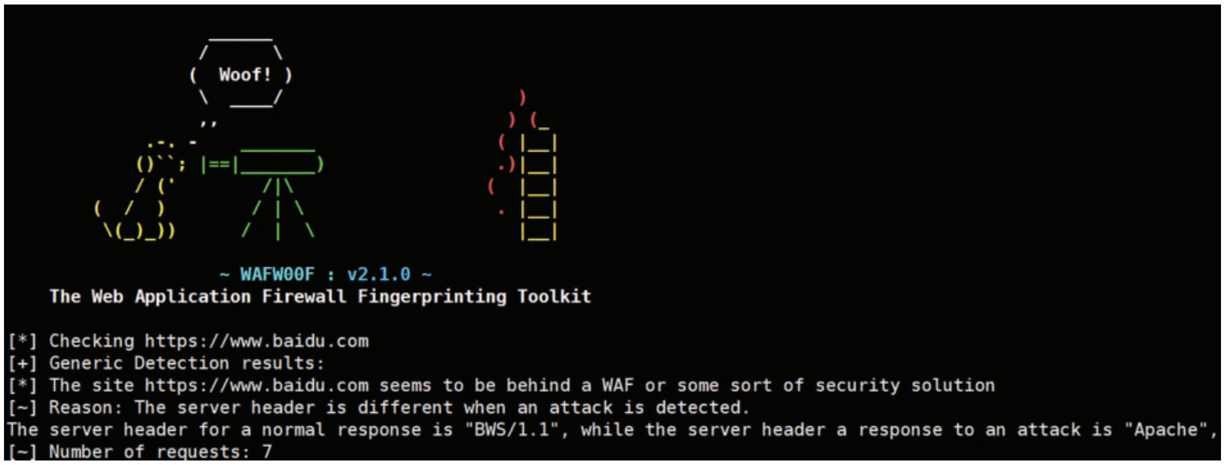
In order to test the integrity of Dis-NDVW's integration of Masscan functions, this paper compares the results of Dis-NDVW basic information scans (as shown in Fig. 10) with the results of Masscan scans (as shown in Fig. 18). According to the test results, it can be concluded that Dis-NDVW fully integrates the functions of Masscan, and users can use Masscan to detect the active IP and active port of IP (segment) in Dis-NDVW.



```
Starting masscan 1.3.2 (http://bit.ly/14GZzcT) at 2022-12-05 07:38:41 GMT
Initiating SYN Stealth Scan
Scanning 1 hosts [6 ports/host]
Discovered open port 80/tcp on 39.156.66.18
```

Figure 18: Masscan scan results for “39.156.66.18”

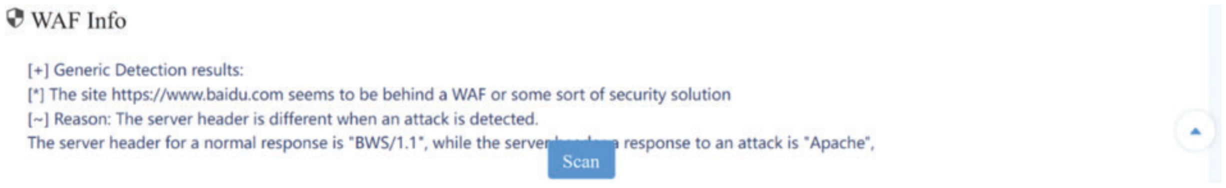
As shown in Figs. 19 and 20, we use “www.baidu.com” as a test case to test the detection capabilities of Dis-NDVW and wafw00f tools on firewalls. According to the test results, the Dis-NDVW is fully integrated with the wafw00f tool and the firewall test results are displayed in full on the system page. This comparison test confirms the capabilities of the Dis-NDVW in terms of firewall detection. This comparison test confirms the capabilities of the Dis-NDVW in terms of firewall detection.



```
                ~ WAFW00F : v2.1.0 ~
    The Web Application Firewall Fingerprinting Toolkit

[*] Checking https://www.baidu.com
[+] Generic Detection results:
[*] The site https://www.baidu.com seems to be behind a WAF or some sort of security solution
[~] Reason: The server header is different when an attack is detected.
The server header for a normal response is "BWS/1.1", while the server header a response to an attack is "Apache",
[~] Number of requests: 7
```

Figure 19: wafw00f scan results for “www.baidu.com”



```
[+] Generic Detection results:
[*] The site https://www.baidu.com seems to be behind a WAF or some sort of security solution
[~] Reason: The server header is different when an attack is detected.
The server header for a normal response is "BWS/1.1", while the server header a response to an attack is "Apache",
```

Figure 20: Dis-NDVW’s firewall probe results for “www.baidu.com”

5 Conclusion

The Dis-NDVW system this paper proposed aims at the needs of users for large-scale network asset management in practical applications. From the perspective of security guarantee for vulnerability detection, it better solves the problem that conventional network asset scanning and detection tools are difficult to complete large-scale tasks. The Dis-NDVW system combines the distributed message publishing and subscription system based on Zookeeper & Kafka and the distributed storage and high-performance retrieval system of mass data based on ElasticSearch to realize a complete set of distributed network asset management and vulnerability detection platforms. Specifically, the Dis-NDVW system has the following advantages:

1. Computing resource isolation. Computing resources between groups are isolated, that is, consumers are not shared between groups, and each Task Group can independently complete a complete task process, which improves fault tolerance;
2. Custom deployment. Users can customize the deployment of consumer groups on the server according to the actual performance of the server and scanning and detection requirements;
3. Efficiently respond to large-scale tasks. Dis-NDVW can scientifically segment the ultra-large-scale scanning and detection tasks and assign them to all available Task Groups for execution;
4. Distributed and highly available. Dis-NDVW adopts a distributed architecture design, which can be flexibly deployed on any number of servers to ensure the availability of the system;

In Section 4.2 of this paper, we tested the performance of Dis-NDVW for large-scale scan detection tasks using the closest sample to a real-world application scenario, “122.10.161.0/24”, as a test case. Further in-depth testing of Dis-NDVW’s performance for large-scale scan detection tasks will be conducted using larger test samples in future work.

Acknowledgement: The authors would like to thank the associate editor and the reviewers for their time and effort provided to review the manuscript.

Funding Statement: This work is supported by the Fundamental Research Funds for the Central Universities (Grant No. HIT. NSRIF.201714), Weihai Science and Technology Development Program (2016DX GJMS15), Weihai Scientific Research and Innovation Fund (2020) and Key Research and Development Program in Shandong Provincial (2017GGX90103).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] V. S. Rossouw and V. N. Johan, “From information security to cyber security,” *Computers & Security*, vol. 38, no. 1, pp. 97–102, 2013.
[2] Y. Nidup, “Awareness about the online security threat and ways to secure the youths,” *Journal of Cybersecurity*, vol. 3, no. 3, pp. 133, 2021.
[3] N. Z. Jhanjhi, M. Humayun and S. N. Almuayqil, “Cyber security and privacy issues in industrial Internet of Things,” *Computer Systems Science & Engineering*, vol. 37, no. 3, pp. 361–380, 2021.
[4] China National Vulnerability Database of Information Security, Data Cube, 2021. https://www.cnnvd.org.cn/home/dataCube
[5] T. Kavitha and D. Sridharan, “Security vulnerabilities in wireless sensor networks: A survey,” *Journal of Information Assurance and Security*, vol. 5, no. 1, pp. 31–44, 2010.
[6] C. Y. Jeong, S. Y. T. Lee and J. H. Lim, “Information security breaches and IT security investments: Impacts on competitors,” *Information & Management*, vol. 56, no. 5, pp. 681–695, 2019.
[7] X. Ju, “An overview of face manipulation detection,” *Journal of Cybersecurity*, vol. 2, no. 4, pp. 197, 2020.
[8] R. T. Fielding, “Architectural styles and the design of network-based software architectures,” M.S. thesis, Donald Bren School of Information and Computer Sciences, University of California, Irvine (UCI), Irvine, USA, 2000.
[9] B. Hu, “Distributed vulnerability emergency detection system,” M.S. thesis, University of Electronic Science and Technology of China, School of computer science and engineering (School of cyberspace security), Chengdu, China, 2010.
[10] E. U. Opara and O. A. Soluade, “Enterprise cyberspace threat landscape: An analysis,” *Journal of Cybersecurity*, vol. 3, no. 3, pp. 167, 2021.

[11] L. Markowsky and G. Markowsky, "Scanning for vulnerable devices in the Internet of Things," in *2015 IEEE 8th Int. Conf. on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, Warsaw, Poland, vol. 1, pp. 463–467, 2015.

[12] S. Liao, C. Zhou, Y. Zhao, Z. Zhang, C. Zhang *et al.,* "A comprehensive detection approach of Nmap: Principles, rules and experiments," in *2020 Int. Conf. on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, Chongqing, China, pp. 64–71, 2020.

[13] N. A. Rakhmawati, S. Harits, D. Hermansyah and M. A. Furqon, "A survey of web technologies used in Indonesia local governments," *SISFO*, vol. 7, no. 3, pp. 7, 2018.

[14] T. Jain and N. Jain, "Framework for web application vulnerability discovery and mitigation by customizing rules through ModSecurity," in *2019 6th Int. Conf. on Signal Processing and Integrated Networks (SPIN)*, Noida, India, pp. 643–648, 2019.

[15] Z. Li, G. Xiong and L. Guo, "Unveiling SSL/TLS MITM hosts in the wild," in *2020 IEEE 3rd Int. Conf. on Information Systems and Computer Aided Education (ICISCAE 2020)*, Dalian, China, pp. 141–145, 2020.

[16] V. Frost, D. Tian, C. Ruales, V. Prakash, P. Traynor *et al.,* "Examining DES-based cipher suite support within the TLS ecosystem," in *Proc. of the 2019 ACM Asia Conf. on Computer and Communications Security*, Auckland, New Zealand, pp. 539–546, 2019.

[17] gobies.org, "Goby," 2021. [Online]. Available: https://cn.gobies.org/

[18] GitHub, "Vxscan," 2021. [Online]. Available: https://github.com/al0ne/Vxscan

[19] M. Shah, S. Ahmed, K. Saeed, M. Junaid and H. Khan, "Penetration testing active reconnaissance phase-optimized port scanning with nmap tool," in *2019 2nd Int. Conf. on Computing, Mathematics and Engineering Technologies (iCoMET)*, Sukkur, Pakistan, pp. 1–6, 2019.

[20] D. T. Giorgio and C. N. Ngo, "Are you a favorite target for cryptojacking? A case-control study on the cryptojacking ecosystem," in *2020 IEEE European Symp. on Security and Privacy Workshops (EuroS&PW)*, Genoa, Italy, pp. 515–520, 2020.

[21] Q. Hua, L. Gao and L. Zhang, "Design and implementation of distributed vulnerability detection system," *Journal of Southeast University (Natural Science Edition)*, vol. 38, no. Sup(I), pp. 94–99, 2008.

[22] X. Tian and D. Tang, "A distributed vulnerability scanning on machine learning," in *2019 6th Int. Conf. on Information Science and Control Engineering (ICISCE)*, Shanghai, China, pp. 32–35, 2019.

[23] P. Hunt, M. Konar, F. P. Junqueira and B. Reed, "ZooKeeper: Wait-free coordination for Internet-scale systems," in *2010 USENIX Annual Technical Conf. (USENIX ATC 10)*, Boston, USA, pp. 23–25, 2010.

[24] J. Kreps, N. Narkhede and J. Rao, "Kafka: A distributed messaging system for log processing," in *Proc. of the NetDB*, Athens, Greece, vol. 11, pp. 1–7, 2011.

[25] V. A. Zamfir, M. Carabas, C. Carabas and N. Tapus, "Systems monitoring and big data analysis using the elasticsearch system," in *2019 22nd Int. Conf. on Control Systems and Computer Science (CSCS)*, Bucharest, Romania, pp. 188–193, 2019.