Attack Chains Construction Method Based on Vulnerabilities Combination

Jing Zhao[1], Hao Sun[1], and Yang Cheng[2]
(Corresponding author: Jing Zhao)
College of Software, Dalian University of Technology[1]
321 TuqiangStreet, Dalian 116620, China
Email: zhaoj9988@dlut.edu.cn
College of Computer Science and Technology, Harbin Engineering University[2]
145 NantongStreet, Harbin 150001, China
(Received May 23, 2021; Revised and Accepted Mar. 28, 2022; First Online Apr. 7, 2022)

Abstract

In computer networks, vulnerabilities exist in all aspects of system design and operation management, and vulnerabilities cannot be eliminated. Therefore, the attacker will use a series of vulnerabilities in the target system to achieve the purpose of the attack. Generally, a multi-stage vulnerabilities combination attack form has a higher success rate and destructiveness. To accurately reflect the security risks brought by the multi-stage vulnerabilities combination attack to the target system, this paper proposes a method of constructing attack chains based on the vulnerabilities combination. First, perform generalized clustering of the vulnerabilities and then use the idea of combined testing to combine the vulnerabilities and build attack chains. At the same time, this paper also proposes using answer set programming to remove redundant and meaningless combinations of vulnerabilities further. Finally, in the experimental environment designed in this paper, the attack experiment is carried out according to the attack chain. The experiment verifies the effectiveness of the attack chain construction method based on the vulnerabilities combination and improves the efficiency of network security analysis.

Keywords: Answer Set Programming (ASP); Attack Chain; Clustering; Vulnerability Combination

1 Introduction

With the continuous development of information technology, the penetration rate of the Internet is getting higher and higher, almost involving most countries and regions in the world. But while enjoying the convenience of the Internet, people often ignore the dangers behind it. With the continuous expansion of the network scale and the increasing complexity of the network structure, there are more and more security issues in the network. Due to the lack of security awareness and protection methods, cyber espionage activities, privacy, and security issues are becoming more frequent, complex, continuous, and difficult to intercept [23]. Among them, targeted network intrusions are also called Advanced Persistent Threats (APT). Attackers will comprehensively consider the vulnerabilities in the network system before APT attacks, and then use a combination of multi-stage vulnerabilities to slowly sneak into the network system to achieve their goals, usually stealing network resources or even network extortion [1, 12]. Attacks launched using combined vulnerabilities are often more subtle and difficult to detect, and are more likely to cause persistent and serious threats to the network. For example, the Russian threat organization APT28 used two 0day vulnerabilities to invade a multinational government agency. The two 0day vulnerabilities are CVE-2015-3043 of Adobe Flash and CVE-2015-1701 of Microsoft Windows. When a user visits a malicious website, Flash triggers the CVE-2015-3043 vulnerability, executes the shellcode to download the payload, and finally triggers CVE-2015-1701 to steal the system token. Egyptian security researcher Yasser discovered three high-risk vulnerabilities on the PayPal website, namely CSRF (cross-site request forgery) vulnerabilities, authentication token bypass vulnerabilities, and security authentication reset vulnerabilities. Yasser used the combination of these three vulnerabilities to reset the answers to the PayPal user's security verification questions, and finally took control of the victim's PayPal account.

Multi-stage vulnerabilities combined attacks can attack targets with high efficiency and high concealment while being strategic and intelligent to avoid detection. Even with the most advanced protection strategies, it is still difficult to avoid this new type of attack [2,20]. Therefore, the vulnerability must be analyzed from the overall network system. Proposing an effective security analysis strategy for the target network system is particularly critical to realize the predictability and visualization of net-

work attacks. At present, many scholars have proposed different analysis methods. For example, [16, 17, 21] proposed a method that can analyze the attack path of the attacker from the perspective of the bottom layer of the operating system. This method believes that the attacker will use some commands to achieve the goal when entering the network system, and the invocation process of these commands will be recorded in the log by the system's monitoring tool or command audit tool, to analyze the causality of the attack. However, this method needs to collect a large number of log records for a specific network, and it is not an easy task to extract effective attack information from a large amount of data. And because of the diversity of logs, it is difficult to analyze the dependency of attacks [14]. As early as 1998, Phillips *et al.* proposed the concept of constructing an attack graph to analyze the vulnerability of the overall network system [22]. Analyze the possible attack path from the attack graph. However, because the attack graph enumerates all possible attack paths in the network system, the complexity and computational cost of constructing the attack graph will increase as the system scales up.

To solve the difficulty of analyzing attack information from log records, we hope to analyze the attack path from the vulnerabilities in the system. In this way, it is possible to analyze the correlation in the attack process more intuitively. Given the high complexity and high cost of the attack graph, we found that during the construction of the attack graph, the exploitation of vulnerabilities in certain attack stages is similar. The similarities mentioned here will be explained in detail later. To reduce the risk of being discovered, the attacker usually chooses a simple and efficient attack path to achieve the goal. Similar vulnerabilities bring similar results to attackers, so attackers usually do not reuse similar vulnerabilities. The attack chain construction technology based on vulnerability clustering-combination can effectively solve the problem of attack graphs. It is worth mentioning that vulnerabilities are widespread in network systems and will always exist [24, 25]. Even if the maintenance personnel make timely patches and updates when they detect vulnerabilities in the system, they cannot guarantee that the vulnerabilities no longer exist and whether new vulnerabilities are introduced. Moreover, in some cases, it is very expensive to repair vulnerabilities, especially in the field of industrial control. It is difficult to repair vulnerabilities. Therefore, it is necessary to combine the vulnerabilities in the known network system, construct the attack chain, and analyze the security of the network system.

The main contributions of this paper are as follows:

1) According to the characteristics of vulnerabilities, this paper designs a formal description method.

2) According to the similarity of vulnerabilities, this paper designs a method of vulnerabilities clustering, which divides vulnerabilities into limited categories.

3) This paper designs two methods to combine vulnerabilities. First, based on the idea of t-way combination, we use the IPOG algorithm to perform a 2-way combination of vulnerabilities. Second, we consider using an answer set programming program to combine vulnerabilities to remove redundant and meaningless combinations of vulnerabilities.

4) Finally, we designed an experimental environment. Then use the attack chain to attack the target system. The experiment proves the effectiveness of our method.

The rest of this paper is organized as follows. In the second section, we reviewed more related work. In the third section, the description method of the vulnerability and the architecture of the attack chain are introduced. Then, in the fourth section, we will introduce the algorithm process of vulnerability clustering and combination in detail. In the fifth section, we conducted an experimental analysis. Finally, summarize the paper.

2 Related Works

Common network security defense methods are passive. For example, virus detection, intrusion detection, firewall, etc. are all passive defense methods. Most of these security defense methods can only be deployed and defended on one node, so the vulnerability of the system cannot be considered from the overall network system. The combined use of vulnerabilities can often bypass these defense methods. Therefore, with increasing vulnerabilities and diversification of attack methods, traditional security defenses cannot meet the needs of network security. Many outstanding scholars are studying the method of vulnerability combination. From the combination of vulnerabilities to analyze the vulnerability of the system, attack graphs are the main research method.

The construction of attack graphs is usually divided into two categories: the first is to represent the network status from the perspective of known vulnerabilities as a whole and enumerate state transitions through model checking [26]. The second is to combine and code individual vulnerabilities by identifying individual causality [9]. The first form suffers from the state explosion problem due to the increase in the number of vulnerabilities, so the second form is more and more popular because of its better scalability.

Attack graphs are also used for offline and online network security analysis. In offline situations, it can be used to determine the best location of firewalls and intrusion detection/defense systems without intervening in the current operation of the target network [6], calculate network security evaluation indicators [13], and perform network security risk analysis [3].

However, the attack graph is mainly constructed from the perspective of system attributes and status. Although it can reflect the combination and utilization of vulnerabilities in the network to a certain extent, it is still not

intuitive enough. To analyze the attacker's behavior pattern and combination of system vulnerabilities, this paper proposes a method to efficiently construct an attack chain based on the current vulnerabilities of the system. At present, academia and industry have two general models for the concept and classification of cyberattacks, namely "Cyber Kill Chain" and "ATT&CK" cyber-attack techniques and tactics. Cyber Kill Chain is an attack model proposed by Lockheed Martin, which is essentially targeted, staged attack. The attacker's attack process is usually planned, and each step has a clear goal. Lockheed Martin divides the attacker's planned steps into seven stages, namely reconnaissance, weaponization, delivery, exploitation, installation, command and control, and actions on objective [27]. The Cyber Kill Chain model reflects that the attacker's penetration process is a combination of various attack methods and a combination of vulnerabilities. MITRE launched the ATT&CK (Adversarial Tactics, Techniques, and Common Knowledge) model in 2013 [18]. The ATT&CK model describes and categorizes the technologies and tactics used in the attack process based on network attack events that occur in the real world. ATT&CK not only classified and summarized the tactics and techniques that attackers may use, but also collected some information about penetration testing teams and hacker organizations, including the techniques and tactics they used, attack weapons, and other information. At present, the ATT&CK model is still being continuously updated, and with the continuous development of the actual attack methods, the techniques and tactics are continuously refined. Using the general classification of attacker behavior in ATT&CK can help security teams such as cyber incident response teams, security operations centers, red and blue teams, threat hunters, and IT departments to better test their detection and response mechanisms against cyber attacks. This paper will build attack chains with the help of the seven stages of the Cyber Kill Chain and the techniques and tactics in the ATT&CK model.

3 The Architecture of The Attack Chain

3.1 Formal Description of Vulnerability

Since CVE and CNNVD describe vulnerabilities in the form of text, this paper uses an automated framework proposed by Joshi *et al.* [7], which can extract entities, concepts, and relationships related to network security from text sources describing vulnerabilities. To enable vulnerabilities to be easily clustered and combined, and to express the prerequisites and consequences of exploiting vulnerabilities, we designed a description method for vulnerabilities and atomic attacks, and the form is as in Equation (1).

$$V_t = (e_t^{Pre}, \lambda, \delta, \sigma, e_t^{Post}) \tag{1}$$

V_t means vulnerability. The vulnerability V_t requires the implementation of λ tactics under e_t^{Pre} conditions, use δ tools to attack σ targets, and the follow-up result can be e_t^{Post}, as in Equation (2).

$$e_t^{Pre} \times \{\lambda, \delta, \sigma\} \rightarrow e_t^{Post} \tag{2}$$

Among them, each attribute value has the following characteristics:

$\lambda \in Tech$, $Tech$ is the collection of network attack techniques and tactics in the ATT&CK framework, which is used to launch attacks by exploiting the vulnerability.

$\delta \in Tool$, $Tool$ represents the set of penetration testing tools or network attack weapon library that an attacker may use.

$\sigma \in Target$, $Target$ represents a collection in the system that can be used as an attack target. It can be a host in the target network, or the operating system, software, or running service on the host.

$e_t^{Pre} = \{c_1, c_2, ..., c_n\}$, $e_t^{Post} = \{c_1, c_2, ..., c_n\}$, $c \in \Sigma$ and Σ represent the collection of target system assets and all available resources or permissions of the system, such as user rights, administrator permissions, system data, etc.

Such a description can describe not only vulnerabilities but also atomic attacks, such as detection and scanning mentioned above. These attacks are indispensable in the process of constructing the attack chain.

3.2 Attack Chain Construction Based on Vulnerability Combination

As shown in Figure 1, this paper is divided into three processes when constructing the attack chain, which are the construction of a vulnerability database, the clustering of vulnerabilities, and the combination of vulnerabilities to form the attack chain.

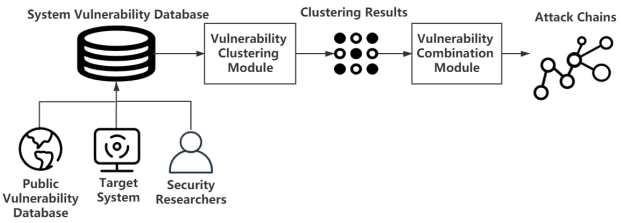

Figure 1: Attack chain construction process

Vulnerability database construction. For a given network system, security analysts use vulnerability scanning tool Nmap to perform host detection, port scanning and vulnerability detection on network systems. Security analysts will count the information of all nodes in the entire network system. The nodes here include PCs, servers, switches, and routers. Security analysts mainly count the operating system version, installed software version, which services are running, and what vulnerabilities exist on the node. Security analysts can easily find the corresponding vulnerability and atomic attack information

from the vulnerability database disclosed by CVE and CNNVD. Then use an automated framework proposed by Joshi *et al.* to extract entities, concepts, and relationships related to network security in the text source of the vulnerability. According to the formal description method of the above vulnerabilities, all vulnerabilities and atomic attacks are formalized and stored in the vulnerability database in the form of entries. At the same time, the $Tech$, $Tool$, $Target$, and Σ generalization hierarchies are constructed through the formal description of vulnerabilities and atomic attacks, and these generalization hierarchies will be used in subsequent cluster analysis.

Vulnerability clustering. This paper will cluster vulnerabilities based on the clustering algorithm of generalized hierarchical structure. For the target network system, if the scale is large, the vulnerability database of the target system will be very large and complex. Assuming that the size of the vulnerability is N, it is known that there are a variety of algorithms that can find the path from one point to the rest. Among them, the most classic way to find the single source shortest path is Dijkstra, and its time complexity is $O(N^2)$. To traverse all attack paths, the time complexity will reach $O(N^3)$. Although this time scale can be solved in polynomial time, the expansion of the scale has reached the cubic level. So when faced with large-scale vulnerabilities, it is very inefficient to use this method to solve all attack chains. Moreover, a large number of offensive and defensive practices have shown that there are certain similarities between many vulnerabilities. Attackers use similar vulnerabilities to achieve the same effect. To reduce the possibility of exposure in network systems, attackers usually do not reuse similar vulnerabilities. Here we may as well assume that there are M groups of vulnerabilities after clustering, and the vulnerabilities between the same groups are not reachable. Then when the single-source shortest path calculation is performed, the time complexity will become $O(M^2)$. Similarly, when traversing all attack paths, the time complexity is $O(M^3)$. Although the order of magnitude is still cubic, the size of M will not increase significantly as the size of the vulnerability increases. It can even be considered to a certain extent that the size of M is fixed, then $O(M^3)$ will be reduced to a constant level. Therefore, the clustering of vulnerabilities is very necessary, but the similarity of vulnerabilities is difficult to define. We will introduce in detail how to cluster vulnerabilities in the fourth section of the vulnerability clustering algorithm.

Vulnerability combination. For vulnerabilities after clustering, we consider using the idea of combined testing [5] to generate a sequence of vulnerability combinations as test cases for further analysis. The number of combinations of vulnerabilities happens to limit the length of the attack path, so we don't have to worry about those ultra-long attack paths. In each group of combinations, there may be vulnerabilities that are related to each other. In the t-way coverage combination test, 2-way coverage can achieve pairwise combinations of all parameters, and the overall number of test cases is much smaller than the combination of all parameters. At the same time, 2-way coverage can ensure that the generated combination contains an attack chain with a length of at least 2. Then we use answer set programming to further filter and merge to get a collection of attack chains. Therefore, the method of combining vulnerabilities and constructing an attack chain using the idea of combined testing can achieve the optimization of the attack chain construction process to a certain extent.

4 Vulnerability Clustering and Combination

4.1 Vulnerability Clustering Algorithm

Julisch *et al.* proposed an algorithm for clustering alarms [8]. The motivation for proposing this algorithm stems from the observation that the alarms of a given root cause are usually similar, but a large number of alarms affects the efficiency of the operation and maintenance personnel to maintain the system. Therefore, the alarms can be clustered, and the alarms with the same root cause can be summarized into a generalized structure that can cover the content of the alarm, and finally, an alarm summary with only a few generalized structures can be formed. We noticed that the vulnerability description method used in this paper is similar to the alarm structure input by the clustering algorithm. Julisch uses a row of attributes with multiple values to represent an alarm. Similar to the way we use five-tuples for vulnerability descriptions, we also use a row of attributes with multiple values. But the difference is that in our vulnerability description method, e_t^{Pre} and e_t^{Post} are not attributes of a single value. And when the attribute value has multiple parent nodes, the original algorithm cannot generalize the vulnerability attribute. Meanwhile, some vulnerabilities may be too unique to be clustered into a group with other vulnerabilities. If a uniform minimum coverage index is adopted, it may lead to over generalization and make clustering meaningless.

Given these three points, we have made the following improvements:

1) When generalizing e_t^{Pre} and e_t^{Post}, each attribute value of them is replaced by a parent node. If any node is generalized as the descendant node of other nodes, these attributes are deleted and only the attribute value with the best generality is retained. After generalizing different vulnerabilities, comparing whether a or B is equal requires comparing whether each value in the set is equal;

2) When a node has more than one parent node to select for generalization, one parent node is randomly selected for generalization;

3) Set the parameter minimum coverage ratio p, which means that all classes don't need to reach the minimum coverage to finish clustering. As long as the

Algorithm 1 Vulnerability Clustering Algorithm

Input: L: A list of vulnerabilities;
Input: G_0: Generalization hierarchy of Σ;
Input: G_1: Generalization hierarchy of $Tech$;
Input: G_2: Generalization hierarchy of $Tool$;
Input: G_3: Generalization hierarchy of $Target$;
Input: min_size: Minimum coverage;
Input: p: Minimum coverage ratio;
Output: A solustion for$(L, G_0, G_1, G_2, G_3, min_size, p)$;

$T := L$;
for all vulnerabilities v in T **do**
  $v[count] := 1$;
end for
while count$(v[count] > min_size)$/**sizeof**$(T) < p$ **do**
  Use heuristics to select an attribute A_i, $i \in \{0, 1, 2, 3\}$;
  for all vulnerabilities v **in** T **do**
    $v[A_i]$ **:= a random father of** $v[A_i]$ **in** G_i
    while identical v, v' **exist do**
      Set $v[count]$ **:=** $v[count] + v'[count]$ **and delete** v' **from** T;
    end while
  end for
end while
return all generalized vulnerabilities $v \in T$ **with** p **of** $v[count] > min\_size$;

number of vulnerabilities in some classes reaches the index, clustering can be stopped.

Our improved algorithm is shown in Algorithm 1.The heuristic functions used in the algorithm are as in Equations (3) and (4).

$$f_i(a) = \mathbf{sum}\{v\,[count]\,|A_i = a\} \quad (3)$$

$$F_i = \mathbf{max}\{f_i(a)|a \in \mathbf{Dom}(A_i)\} \quad (4)$$

Since the value of the same attribute A_i is different, the corresponding number of vulnerabilities is different. The function of f_i is to count the number of vulnerabilities whose attribute A_i corresponds to different values among all vulnerabilities. F_i is to count the largest number of vulnerabilities in different attribute values. The heuristic function guides which attribute should be selected for generalization in each iteration. Through this heuristic function, as many vulnerabilities as possible can be classified into the same category at each step. During the execution of the algorithm, we hope to save the classification results of vulnerabilities with a collection *coverlist*. The actual operation steps of the algorithm are as follows:

1) Because the generalization hierarchy G_i may not be a tree structure, a node may have multiple parent nodes, so the algorithm randomly selects a parent node for generalization.

2) Each vulnerability object will save a collection *coverlist*. When the *coverlist* is initialized, there is only a reference to the vulnerability itself. The meaning is that each vulnerability is in its category at the beginning, and other categories will be added during clustering.

3) Use heuristic function to select one of e_t^{Pre},λ, δ, σ, e_t^{Post} as A_i, and the A_i value of all vulnerabilities in T is replaced by the parent value of A_i in its generalized hierarchy G_i. If A_i is e_t^{Pre} or e_t^{Post} , each attribute value in the selected set is generalized, and each step determines whether the attribute in the set has an inclusion relationship. If so, the two attributes are merged.

4) Scan all vulnerabilities at this time. If the attributes of any vulnerability are identical, add the *coverlist* of the second vulnerability to the *coverlist* of the first vulnerability, and delete the second vulnerability from the vulnerability formal description table.

5) Continue the operation of Steps (3) (4). Because some vulnerabilities may vary greatly, they will not be covered unless they are generalized to the root node. Excessive generalization is meaningless, so when the percentage of vulnerabilities whose *coverlist* size is greater than min_size in all vulnerabilities reaches p, clustering stops.

6) Output the remaining vulnerabilities in Step (5). The *coverlist* of each vulnerability is all the vulnerabilities of this class.

4.2 Vulnerability Combination Algorithm

4.2.1 Vulnerability Combination Based on ACTS

This paper chooses to use ACTS [19] tool to generate vulnerability combinations. ACTS is a test case generation tool for constructing t-way coverage combination, which is widely used in system combination testing [4]. Due to its good performance, this paper chooses to use it to generate a 2-way coverage combination of vulnerabilities. ACTS supports the use of multiple algorithms to generate t-way coverage combinations, these algorithms include IPOG, IPOG-D, IPOG-F, and IPOG-F2, etc. [10, 11].

The strategy of the IPOG algorithm for constructing the t-way combination is expansion-based. In each iteration, horizontal expansion and vertical expansion are performed until the generated test cases can cover all t-way combinations. The strategy framework is described as follows: First, the categories are sorted non-increasingly according to the number of parameters in each category. Then select the first t classes to form all the combinations of the parameters in these classes, and expand the combination level to the t+1 parameter. If the horizontal expansion cannot guarantee to cover the t-way combination of the first t+1 parameters, then the vertical expansion is performed until it is satisfied. And so on, until all the t-way combinations of parameters can be covered.

Among them, horizontal expansion refers to the expansion of each existing combination by adding a value for the new parameter. Vertical growth refers to adding new combinations to the test set generated by horizontal expansion when necessary.

Algorithm 2 describes the test generation algorithm that implements this strategy, named IPOG-Test. The input of the algorithm is two parameters: an integer t that specifies the coverage strength, and a parameter set PS that contains the input parameters and their values. The output of the algorithm is the t-way test set of the parameters in PS. Assume that the number of parameters in the set PS is greater than or equal to t. Figure 2 shows the application of the IPOG-Test algorithm in an example 3-way test system. This example system consists of four parameters, P1, P2, P3, P4, where P1, P2, P3 have two values 0 and 1, and P4 has three values 0, 1, and 2. The algorithm first generates all the combinations between the three parameters P1, P2, and P3, that is, 2^3 combinations. Then, expand horizontally based on the original 8 combinations, and introduce the three parameters of P4 into the combination. However, after introducing the three parameters of P4, the combination of P4 and any three parameters before P1, P2, and P3 cannot be covered. Therefore, vertical expansion is required until the final combination can cover any combination of three parameters among these four parameters.

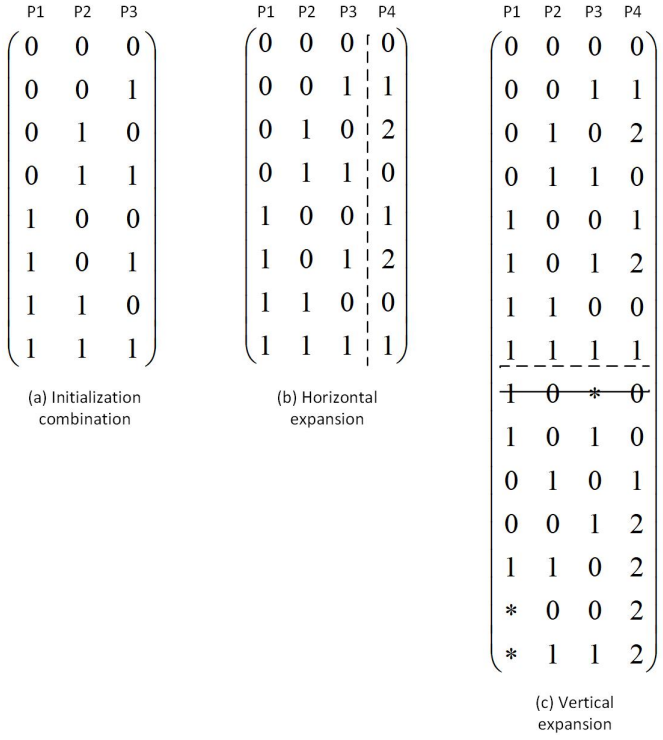

Figure 2: Schematic diagram of IPOG algorithm

Algorithm 2 IPOG Algorithm

Input: t: the strength of Combnation
Input: PS: the parameter set
Output: TS:Numerical test cases set

```
initialize test set TS to be an empty set
sort the parameters in set PS in a non-increasing order of their domain sizes, and denote them as P1, P2,...,Pk
add into TS a test for each combination of values of the first t parameters
for t+1 ≤ i ≤ k do
    let π be the set of all t-way combinations of values involving parameter P_i and any group of t−1 parameters among the first i−1 parameters
    // horizontal extension for parameter P_i
    for each test (o=v_1, v_2, ..., v_{i−1}) in TS do
        choose a value v_i of P_i and replace o with o'={v_1, v_2, ..., v_{i−1}, v_i} so that o' covers the most number of k-way of values in π
        remove from π the combinations of values covered by o'
    end for
    // vertical extension for parameter P_i
    for each combination σ in set π do
        if there exists o in TS such that it can be changed to cover σ then
            change test o to cover σ
        else
            add a new test to cover σ
        end if
    end for
end for
return TS
```

4.2.2 Vulnerability Combination Based on Answer Set Programming

Although the number of combinations generated by 2-way coverage is much smaller than the number of combinations of all vulnerabilities, it is still possible to generate many redundant and meaningless combinations of vulnerabilities. Since there are constraints between vulnerabilities and vulnerabilities, that is, some vulnerabilities are the prerequisites for other vulnerabilities, these constraints can be found first to assist in solving the vulnerability combinations, and this method can further reduce the number of generated combinations. Considering this feature, we can use answer set programming to solve the vulnerability combination.

Answer set programming [15] is a declarative programming method and language, mainly used to solve complex search problems, and can solve combined problems well. The main focus of programming with answer sets is the model of the problem, not the solution of the problem, which is very different from languages such as Java and python. When the problem is modeled using the syntax of answer set programming, the solution of the problem can

be obtained by running the answer set solver. For answer set programming, the University of Potsdam has developed an assembly Potassco (Potsdam Answer Set Solving Collection). Among them, clingo can be used to run ASP code. Therefore, for a problem that builds an attack chain based on a combination of vulnerabilities, we can use the answer set programming method to describe the vulnerability combination problem as an answer set programming logic program, and then use clingo to solve it. Finally, get the vulnerability combination. The main special symbols of answer set programming are shown in Table 1.

Table 1: ASP Special Symbol

Symbol	Function
%	Represents the start of code comments.
.	Represents the end of statement.
:-	Represents a constraint.
,	Represents "AND".
;	Represents "OR".
#show	Represents the output.

According to the vulnerability constraints, we can design programs to complete the modeling of automatic implementation problems. It mainly consists of four parts:

1) All possibilities of vulnerability combination. Assuming that four classes are obtained by clustering, four-parameter predicates need to be constructed. The values of parameters in each class are separated by ";".

2) Vulnerability constraints. The basic requirement of the attack chain is that the result of the previous exploit can be the precondition of the next atomic attack.

3) Constraints of problem modeling. These constraints are mainly used to eliminate the vulnerability combinations that do not contain the constraints between vulnerabilities in (1), and a group of combinations needs to contain more than one directed edge. When designing an answer set programming program, we can set the number of constraints at least contained in each group.

4) The final output is based on the result of the vulnerability combination attack chain construction.

5 Experiments and analysis

5.1 Experimental Environment Design

We designed an experimental environment, as shown in the Figure 3. The experimental environment is mainly composed of the attacker and target system. Attackers

Table 2: ASP special symbol

Machine	IP
Attacker	10.10.10.128
Web Server (OWASPBWA)	10.10.10.129
Back-end Server (Win2k3 Metasploitable)	10.10.10.130
Gateway	10.10.10.254 192.168.10.254
Intranet Client (WinXP Metasploitable)	192.168.10.128

are mainly computers with pre-installed penetration testing tools, such as Kali Linux or BackTrace. The target system is mainly composed of 4 machines, which are website server (OWASPBWA), back-end server (Win2k3 Metasploitable), gateway server (Linux Metasploitable), and intranet client (WinXP Metasploitable).

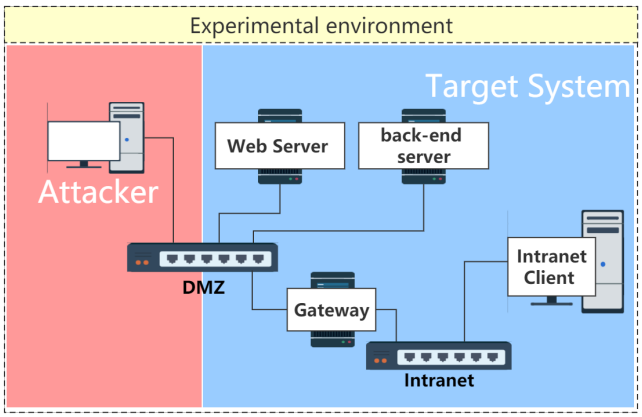

Figure 3: Experimental environment

OWASP BWA brings together a large number of training experimental environments and real web applications with known security vulnerabilities. There are various pre-set web applications with vulnerabilities, which are divided according to the security level, and the defect code programs at each security level are given. Win2k3 Metasploitable, Linux Metasploitable, WinXP Metasploitable, etc. are a series of virtual target machine images, these virtual machines contain a large number of unfixed security vulnerabilities.

Set two network segments through VMware's virtual network setting function, and distinguish the internal network and external network of the target system through the gateway. Intranet clients can access the internal network through the gateway, but the attacker cannot directly access intranet clients. The IP address of each machine is set as Table 2.

The virtual machine of the target system is specially selected by us. These virtual machines have some classic

vulnerabilities. According to our collation and continuous experiments, we mainly used 37 vulnerabilities in our experiment.

5.2 Experimental Data

The input required for the attack chain construction technology based on the vulnerability combination is the vulnerability list, the technical and tactical generalization hierarchy, the attack weapon generalization hierarchy, the target system asset generalization hierarchy, and the attack target generalization hierarchy. The construction of these inputs has been introduced in Section 3. To facilitate the operation of the program, the data is stored in XML format. Due to space limitations, this section mainly shows the vulnerability list and technical and tactical generalization hierarchy.

List of vulnerabilities/atomic attacks. The XML document of the attack list stores all the vulnerabilities/atomic attacks of the system. After scanning the target system with tools such as Nmap, W3AF, and Metasploit, there is no hierarchical division of atomic attacks generated by combining manual operations, and they are unified under the root node. The assets used by each atomic attack are stored in the "Conditions" node, the techniques and tactics used are stored in the "tech" node, the attack tools are stored in the "tool" node, the targets are stored in the "target" node, and the results generated are stored in the "results" node. For example, CVE-2009-1979 on the back-end server, its manifestation is shown in the Figure 4.

```
<bug id="11" type="Back-end_CVE-2009-1979">
  <Conditions>10.10.10.130:1521_oracle-tns</Conditions>
  <Conditions>10.10.10.130_Oracle_10.2.0.1.0</Conditions>
  <tech>Thread_Local_Storage</tech>
  <tool>msf_exploit_oracle_tns_auth_sesskey</tool>
  <target>10.10.10.130_Oracle_10.2.0.1.0</target>
  <Results>10.10.10.130_admin</Results>
</bug>
<bug id="12" type="Back-end_CVE-2011-0406">
...
</bug>
<bug id="13" type="Back-end_MS08-067">
...
</bug>
```

Figure 4: Vulnerability list

Technical and tactical generalization hierarchy. We designed a crawler script to crawl the data in https://attack.mitre.org/beta/ and build a generalized hierarchy of attack techniques and tactics based on the hierarchical structure of the ATT&CK matrix. Part of the manifestation of this hierarchical structure is shown in the Figure 5.

5.3 Vulnerability Clustering Experiment

This experiment uses the vulnerability clustering algorithm described in Section 4.1 to cluster vulnerabilities. According to different parameter settings, different clustering results can be obtained. The main parameters to

```
<Level_1>
  att&ck
  <Level_2>
    Initial_Access
    <Level_3>Drive-by_Compromise</Level_3>
    <Level_3>Exploit_Public-Facing_Application</Level_3>
    <Level_3>External_Remote_Services</Level_3>
    <Level_3>Hardware_Additions</Level_3>
    <Level_3>
      Phishing
      <Level_4>Spearphishing_Attachment</Level_4>
      <Level_4>Spearphishing_Link</Level_4>
      <Level_4>Spearphishing_via_Service</Level_4>
    </Level_3>
    <Level_3>Replication_Through_Removable_Media</Level_3>
    <Level_3>
    ...
```

Figure 5: ATT&CK generalization hierarchy

be set are *min_size* and p, which are the minimum coverage and the minimum coverage ratio. When *min_size* is set to 1, it means that there is only one vulnerability in each class. At this time, clustering is meaningless, so the value of *min_size* must be greater than 1. We tested multiple sets of use cases through dichotomy, and finally determined *min_size* and p. Take two of them as examples.

In the first group of clustering experiments, the parameters we adopted were $min\_size = 2$, $p = 1$. The experiment finally divided vulnerabilities into 2 classes. The vulnerabilities of each class are shown in Table 3, and the numbers represent different vulnerabilities. It can be seen that at this time, it has been over-generalized, and only divided into two groups is useless for vulnerability combinations.

Table 3: Clustering Result 1

Class	Vulnerabilities
1	1,8,2,9,4,5,3,6,24,7,21
2	10,11,35,17,18,34,12,14,16, 25,26,28,30,32,13,15,27,31, 29,33,19,20,36,37,22,23

In the second group of clustering experiments, the parameters we adopted were min_size=2, p=0.25. The experiment finally divided atomic attacks into 11 categories. The specific atomic attacks of each category are shown in Table 4. In this set of experiments, the number of categories 1, 4, 5, 6, and 10 exceeded *min_size*. However, there is only one vulnerability in the 2, 3, 7, 8, 9, and 11 categories. Among them, the fourth category contains 13 vulnerabilities. By analyzing the list of vulnerabilities, it can be known that the main target of these vulnerabilities is 10.10.10.130 and 10.10.10.254, which can achieve control of the target, but use different software vulnerabilities, different services have been attacked, different techniques and tactics have been used, and the use conditions and consequences are also different. The remaining vulnerabilities in categories 1, 5, 6, and 10 also achieved the goal of clustering "similar" vulnerabilities. This set of

Table 4: Clustering Result 2

Class	Vulnerabilities
1	1,8,2,9,4,5,3,6,24
2	7
3	10
4	11,35,17,18,34,12,14, 16,25,26,28,30,32
5	13,15,27,31
6	19,20,36
7	21
8	22
9	23
10	29,33
11	37

clustering results is relatively good, and the experiment in the vulnerability combination stage will adopt this set of clustering results.

5.4 Vulnerability Combination Experiment

We have designed two methods for vulnerability combination, which are mainly based on the combination test tool ACTS's vulnerability combination and the further screening of the vulnerability combination based on answer set programming.

This experiment uses the vulnerability combination method described in Section 4.2 and invokes the interfaces of the ACTS tool, and selects the vulnerabilities from each class of the clustering results for combination. Since 2-way coverage has been able to find pairwise combinations of vulnerabilities between different classes, it can find all effective attack paths with a length of at least 2. Therefore, 2-way coverage is used for vulnerability combination. Part of the result is shown in Figure 6.

```
# ACTS Test Suite Generation: Tue May 26 02:48:36 PDT 2020
#  * represents don't care value

# Degree of interaction coverage: 2
# Number of parameters: 11
# Maximum number of values per parameter: 13
# Number of configurations: 2808
# ************************************************
0,1,2,3,4,5,6,7,8,9,10
1,7,10,11,13,19,21,22,23,29,37
1,7,10,11,13,19,21,22,23,33,37
1,7,10,11,13,20,21,22,23,29,37
1,7,10,11,13,20,21,22,23,33,37
```

Figure 6: Combination result

Each row of the result represents a combination of vulnerabilities. A total of 2808 2-way combinations were generated. From these combinations, attack chains can be obtained, and each attack chain appears differently. However, the vulnerabilities combined in this way have overlapping parts and can be further merged. According to the vulnerability combination method based on answer set programming mentioned in Section 4.2.2, we designed the ASP program and used clingo to solve it. In this way, the relationship between the 2-way combinations of vulnerabilities is no longer considered, but only whether the number of constraints in a set of combinations meets the requirements. We set each group needs to contain at least 4 constraints. The results are shown in Table 5. In Table 5, F represents the frequency of each attack chain. Each number in the attack chain represents a vulnerability. It can be seen from Table 5 that some attack chains overlap, so attack chains can be further merged.

Table 5: Attack Chain Statistics 2

Index	F	Attack Chain	Index	F	Attack Chain
1	1102	7→21	30	312	6→7→21
2	1102	7→22	31	312	6→7→22
3	1102	7→23	32	312	6→7→23
4	312	6→7	33	156	29→6→7→22
5	216	10→17	34	156	29→6→7→23
6	216	35→37	35	156	29→6→7→21
7	156	29→6	36	78	31→6→7→22
8	108	33→17	37	78	27→6→7→22
9	78	5→27	38	78	31→6→7→21
10	78	31→6	39	78	27→6→7→23
11	78	27→6	40	78	31→6→7→23
12	78	4→13	41	78	27→6→7→21
13	78	4→15	42	72	10→17→19
14	72	17→19	43	72	10→17→20
15	72	17→20	44	36	33→17→19
16	72	18→19	45	36	33→17→20
17	72	35→36	46	24	24→35→37
18	72	18→20	47	24	32→6→7→23
19	24	30→6	48	24	28→6→7→21
20	24	4→12	49	24	30→6→7→23
21	24	24→35	50	24	32→6→7→22
22	24	32→6	51	24	30→6→7→22
23	24	8→18	52	24	28→6→7→22
24	24	8→25	53	24	32→6→7→21
25	24	8→26	54	24	30→6→7→21
26	24	4→11	55	24	28→6→7→23
27	24	4→14	56	8	8→18→20
28	24	28→6	57	8	24→35→36
29	8	8→18→19			

It can be seen that the vulnerability combination method based on answer set programming is the same as the attack chain generated by the ACTS-based vulnerability combination method, but the total number generated by the vulnerability combination method based on answer set programming is small, which reduces the sample space while achieving the same effect.

To further analyze the attack chains that exist on the experimental target system, the generated attack chains need to be merged. The final result is shown in Figure 7. Attack chain merge result graph, each node represents a vulnerability. We will use the attack chain to test in the experimental environment.

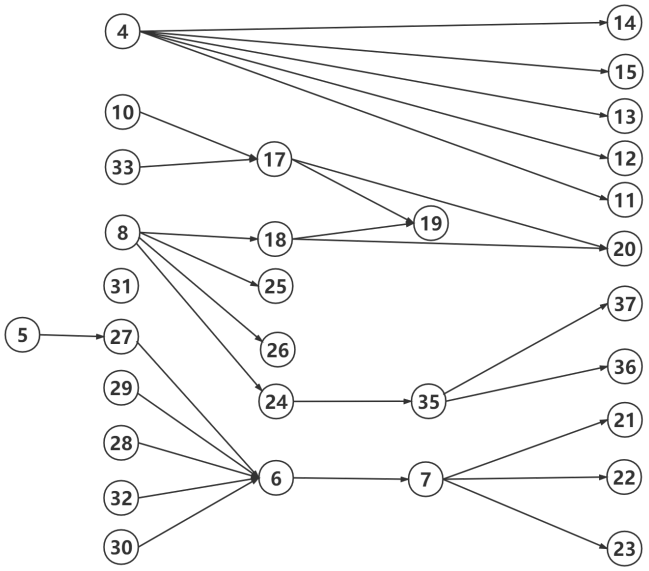

Figure 7: Attack chains

5.5 Attack Chain Implementation Experiment

In response to the 37 vulnerabilities of this experimental platform, we have compiled a manual for their utilization methods. The attack can be easily realized with the help of the manual. This section does not enumerate all the processes of the attack chain. We choose the longest attack chain {5,27,6,7,21} for illustration, and the most difficult attack chain is used to illustrate the effectiveness of the attack process. The attack process is as follows:

1 Vulnerability No.5: Obtain the services running on each port of the gateway server. Use Nmap to use the "-sT -PN -spoof-mac 0" option to detect 10.10.10.254, then we can get the result of the port service of the gateway.

2 Vulnerability No.27: Exploit CVE-2007-2447 and obtain the root of the gateway server. According to the results of the previous step, it can be seen that the gateway has opened port 139 to the outside world. Usually, samba runs on this port. Some versions of the samba service have the CVE-2007-2447 vulnerability. Use Metasploit's payload to attack the gateway. The gateway does have a vulnerability, so the control of the gateway is obtained.

3 Vulnerability No.6: Obtain the IP of the host that has communicated with the gateway. Since the attacker cannot directly scan the intranet, we can find the IP of the intranet client through the "arp" command of the gateway.

4 Vulnerability No.7: Scan the port of 192.168.10.128. After obtaining the IP of the intranet client, we can use Nmap to scan the client.

5 Vulnerability No.21: Exploit MS08-067(CVE-2008-4250) and get control of the intranet client. After scanning the client in the previous step, it is known that the internal network client machine opens port 445. This port is an SMB channel. The SMB channel may have the MS08-067 vulnerability. Then we can use Metasploit's payload to attack the intranet client. It is found that the attack is successful and the client control is obtained. The attack result is shown in the Figure 8.

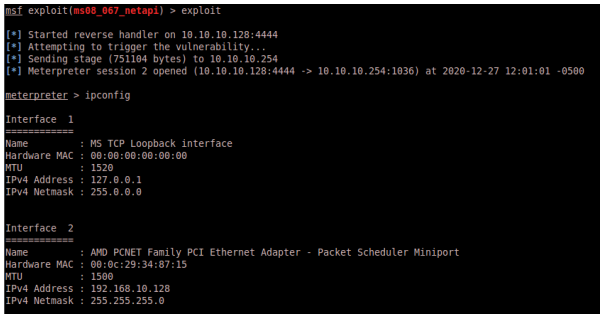

Figure 8: Result: Penetration into the intranet

Through the experimental process, the attack chain realized the penetration of the target system from the outside to the inside, and finally gained control of the intranet client. This attack chain realizes step-by-step penetration from the external network into the internal network and gains the control authority of the internal network machine, which has a strong threat. We carry out experiments according to the attack chain screened in Figure **??**, and all of them can achieve the purpose of the attack. To verify the effectiveness of our attack chain method for the target network construction, we analyze the different vulnerable attack paths in the target network as a whole, which provides a basis for the subsequent network defense process.

6 Conclusion

To achieve high efficiency and visualization of target network security analysis, this paper proposes an attack chain construction method based on the combination of vulnerabilities based on the research status of vulnerabilities and attack models. This paper uses the 5-tuple field to formally describe the vulnerabilities, using this method to build a vulnerability library for the target network. At the same time, we build a generalization hierarchy for each attribute on the 5-tuple, which makes the vulnerabilities can be clustered. Then we combined the generalized hierarchy of each attribute in the vulnerability, and we proposed a clustering algorithm to group "similar" vulnerabilities into one category, so the vulnerabilities are grouped into a limited number of categories.

We use the combination test tool ACTS for 2-way vulnerabilities combination, which guarantees a way to find

all the two-way combinations of vulnerabilities. Further use answer set programming to solve vulnerabilities combinations to remove redundant and meaningless combinations. Finally, an experimental simulation environment was built, and the attack chain constructed by the vulnerability combination was used to conduct attack experiments. The experiment shows the effectiveness of the attack chain construction method and provides a basis for the subsequent network system defense. Due to the limitation of the experimental environment, the target network designed in this paper is not complex enough. In the future, we will build a larger network attack and defense environment, and introduce industrial control systems, rail transit systems, and other simulation scenarios that are closely related to the construction of network security and national defense. In the complex network environment, the validity and practicability of the attack chain construction technology based on the combination of vulnerabilities are further verified.

References

[1] A. Alshamrani, S. Myneni, A. Chowdhary, and D. Huang, "A survey on advanced persistent threats: Techniques, solutions, challenges, and research opportunities," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1851–1877, 2019.

[2] F. J. Aparicio-Navarro, K. G. Kyriakopoulos, I. Ghafir, S. Lambotharan, and J. A. Chambers, "Multi-stage attack detection using contextual information," in *MILCOM 2018-2018 IEEE Military Communications Conference (MILCOM)*. IEEE, 2018, pp. 1–9.

[3] K. Beckers, M. Heisel, L. Krautsevich, F. Martinelli, R. Meis, and A. Yautsiukhin, "Determining the probability of smart grid attacks by combining attack tree and attack graph analysis," in *International Workshop on Smart Grid Security*. Springer, 2014, pp. 30–47.

[4] M. N. Borazjany, L. Yu, Y. Lei, R. Kacker, and R. Kuhn, "Combinatorial testing of acts: A case study," in *2012 IEEE Fifth International Conference on Software Testing, Verification and Validation*. IEEE, 2012, pp. 591–600.

[5] M. Grindal, J. Offutt, and S. F. Andler, "Combination testing strategies: a survey," *Software Testing, Verification and Reliability*, vol. 15, no. 3, pp. 167–199, 2005.

[6] S. Jajodia and S. Noel, "Topological vulnerability analysis," in *Cyber situational awareness*. Springer, 2010, pp. 139–154.

[7] A. Joshi, R. Lal, T. Finin, and A. Joshi, "Extracting cybersecurity related linked data from text," in *2013 IEEE Seventh International Conference on Semantic Computing*. IEEE, 2013, pp. 252–259.

[8] K. Julisch, "Clustering intrusion detection alarms to support root cause analysis," *ACM transactions on information and system security (TISSEC)*, vol. 6, no. 4, pp. 443–471, 2003.

[9] M. Khouzani, Z. Liu, and P. Malacaria, "Scalable min-max multi-objective cyber-security optimisation over probabilistic attack graphs," *European Journal of Operational Research*, vol. 278, no. 3, pp. 894–903, 2019.

[10] Y. Lei, R. Kacker, D. R. Kuhn, V. Okun, and J. Lawrence, "IPOG: A general strategy for t-way software testing," in *14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'07)*. IEEE, 2007, pp. 549–556.

[11] Y. Lei, R. Kacker, D. R. Kuhn, V. Okun, J. Lawrence, "IPOG-IPOG-D: Efficient test generation for multi-way combinatorial testing," *Software Testing, Verification and Reliability*, vol. 18, no. 3, pp. 125–148, 2008.

[12] A. Lemay, J. Calvet, F. Menet, and J. M. Fernandez, "Survey of publicly available reports on advanced persistent threat actors," *Computers & Security*, vol. 72, pp. 26–59, 2018.

[13] E. Lemay, W. Unkenholz, D. Parks, C. Muehrcke, K. Keefe, and W. H. Sanders, "Adversary-driven state-based system security evaluation," in *Proceedings of the 6th International Workshop on Security Measurements and Metrics*, 2010, pp. 1–9.

[14] T. Li, J. Ma, Q. Pei, Y. Shen, C. Lin, S. Ma, and M. S. Obaidat, "Aclog: attack chain construction based on log correlation," in *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2019, pp. 1–6.

[15] V. Lifschitz, *Answer set programming*. Springer Berlin, 2019.

[16] Y. Liu, M. Zhang, D. Li, K. Jee, Z. Li, Z. Wu, J. Rhee, and P. Mittal, "Towards a timely causality analysis for enterprise security." in *NDSS*, 2018.

[17] S. M. Milajerdi, R. Gjomemo, B. Eshete, R. Sekar, and V. Venkatakrishnan, "Holmes: real-time apt detection through correlation of suspicious information flows," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 1137–1152.

[18] MITRE ATT&CK, *ATT&CK*, Feb. 19, 2022. (https://attack.mitre.org/)

[19] NIST, *Website for the NIST Automated Combintorial Testing (ACTs) Project*, Feb. 19, 2022. (https://www.nist.gov/programs-projects/automated-combinatorial-testing-software-acts)

[20] J. P. P. M. Orvalho and R. M. S. Silva, "Flexible approach to multi-stage network attack recognition," *International Journal of Computer Science and Information Security (IJCSIS)*, vol. 17, no. 8, 2019.

[21] T. Pasquier, X. Han, T. Moyer, A. Bates, O. Hermant, D. Eyers, J. Bacon, and M. Seltzer, "Runtime analysis of whole-system provenance," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 1601–1616.

[22] C. Phillips and L. P. Swiler, "A graph-based system for network-vulnerability analysis," in *Proceedings of the 1998 workshop on New security paradigms*, 1998, pp. 71–79.

[23] S. Smarter and S. Malware, "Security threat report 2014."

[24] P. S. Shinde and S. B. Ardhapurkar, "Cyber security analysis using vulnerability assessment and penetration testing," in *2016 World Conference on Futuristic Trends in Research and Innovation for Social Welfare (Startup Conclave)*. IEEE, 2016, pp. 1–5.

[25] V. Subrahmanian, M. Ovelgonne, T. Dumitras, and B. A. Prakash, "The global cyber-vulnerability report," 2015.

[26] L. P. Swiler, C. Phillips, D. Ellis, and S. Chakerian, "Computer-attack graph generation tool," in *Proceedings DARPA Information Survivability Conference and Exposition II. DISCEX'01*, vol. 2. IEEE, 2001, pp. 307–321.

[27] T. Yadav and A. M. Rao, "Technical aspects of cyber kill chain," in *International Symposium on Security in Computing and Communication*. Springer, 2015, pp. 438–452.

Biography

Jing Zhao received the Ph.D. degree in computer science and technology from the Harbin Institute of Technology of China in 2006. In 2010, she was with the Department of Electrical and Computer Engineering, Duke University, Durham, North Carolina, working as a postdoctoral researcher under the supervision of Dr. Kishor Trivedi. From 2006 to 2018, she was a professor at the School of Computer Science and Technology, Harbin Engineering University, China. She is currently a professor at the School of Software Technology, Dalian University of Technology, China. Her research interests include cyber security and Internet of vehicles security.

Hao Sun is currently a master of Software Engineering, Dalian University of Technology. He received a bachelor's degree from Harbin Engineering University in 2019. His research interests include network security and Internet of Things security.

Yang Cheng received a master's degree in computer science and technology from Harbin Engineering University in 2021. He received a bachelor's degree from Harbin Engineering University in 2018. His research direction is mainly network security.