An attack graph-based approach to network vulnerability analysis

Panpan Yang [a], Jing Zhao, Chengli Wang

Unit 92020 of the PLA, Tsingtao, China

[a] helloyangpan@163.com

Abstract. As the global threat situation intensifies, it is difficult for traditional security strategies to effectively respond to complex attack methods. In this paper, we propose a value-aware and memorized attack graph generation method which successfully compresses the strategy space and effectively reduces the algorithm complexity. Through a virtual network experiment, we validate the feasibility of the proposed methods above.

Keywords: Network vulnerability assessment; attack graph; network security.

1. Introduction

Along with the escalation of the epidemic and global threats, security breaches, data leaks, ransomware and other cyber security threats have become increasingly prominent, and the situation of organised and purposeful cyber attacks has become more pronounced [1]. Attackers exploit the dependencies between vulnerabilities and design to choose different attack paths to achieve intrusion into specific services or hosts. Considering that, in practice defenders cannot deploy defence resources for all attack paths due to limited resources [2], traditional means of network defence are no longer effective against coordinated intelligent attack patterns. Therefore, it has become a current research hotspot to identify multi-stage cross-host attack paths in the network by combining the backward and forward correlation conditions of vulnerability in the network topology and considering the host connection relationship comprehensively. In this paper, from the perspective of the attacker, we simulate specific execution situations, exploit the dependency relationships between the vulnerabilities of various software and services in the network, generate a map of attack strategies made by the attacker from a psychological perspective, identify the local optimal set of attacks in the target network, and provide a theoretical basis for the comprehensive prediction of possible attacks, so as to reasonably construct network security defences and maximise the security protection effect.

2. Current status of research

Attack graph is a graph theory method to determine network security by studying the interrelationship between nodes in a network, which is widely used in network penetration testing, network security defense, and network attack simulation and emulation. The earliest concept of an attack graph was proposed by Swiler[3] , and their proposed attack graph is a representative of the early manual construction of attack graphs. Extending to the automatic generation of attack graph research stage, researchers first obtain the configuration information of the target network, then establish the attack pattern, consider the attack environment and attacker capabilities, combine the above three parts of information, and use the attack graph generation algorithm to generate the target network attack graph. Different attack graph generation algorithms generate different attack graphs, and the search methods generally use a hybrid search strategy, combining monotonicity assumptions and selection restrictions to reduce the attack graph size and enhance the readability of the attack graph. The Ingols algorithm[4] has the best performance as the complexity grows polynomially with the number of hosts, with the disadvantage that it is only applicable to remote buffer overflow attack patterns and does not support other attack methods. This paper focuses on how to improve the efficiency of attack graph generation while controlling the size of the attack graph, and proposes a value-aware search strategy to ensure that the direction of each search is approximately optimal and

improve the search efficiency. The generated attack graphs are between single- and multi-objective attack graphs, with good scalability and ease of use.

3. A new attack graph generation method

The traditional attack graph generation strategy is to use forward search or inverse search to generate the network attack graph, which shows exponential growth in time complexity as the number of nodes increases. In this paper, we adopt the idea of combining BFS and inverse search, propose a host asset-based computing model to induce the concept of relay nodes, and adopt a value-aware search strategy based on relay nodes as the centre to perform memory search by path-tracing method to ensure that each search direction is near-optimal and improve the search performance.

3.1 Mainframe asset calculation model

The asset value reflects the importance of the device and the likelihood of an attacker attacking it. Each device is rated according to its position in the network system and the possible consequences of successful penetration by an attacker, using an expert scoring method to set a quantitative asset value or each device, calculated as shown in Equation (1).

$$\text{value} = \frac{h_{v_i}}{\sum_{i=1}^{n} h_{v_i}} \tag{1}$$

Where h_{v_i} represents the value of the asset on the ith host (where i takes 1 to the number of all devices in the network, n) and it is calculated as

$$h_{v_i} = \lambda_1 v_{info} + \lambda_2 v_{sw} + \lambda_3 v_{hw} \quad (\lambda_1 + \lambda_2 + \lambda_3 = 1) \tag{2}$$

v_{info} refers to the value of information assets, which is a quantitative value between 0 and 10. Information assets include every piece of information in the organisation, including databases, customer information, data files, operational and support processes, archived information and continuity plans. v_{sw} refers to software assets, which include systems and applications that the organisation has purchased or developed itself. v_{hw} refers to the value of physical assets, which include computer hardware, storage media and printers in the organisation. The information asset value, the software asset value and the physical asset value of the host hi are represented by a triplet at. Considering these three scales together, a cluster decision method is used to give a reasonable weighting.

3.2 Relay node calculation method

Name: Key Host Quantification Algorithm
Input: network host set HS; network topology map G; access control rule ACL; initial network security attribute set P.
Output: KeyHost set of key target hosts.
Process.
Initialize KeyHost = $\emptyset$
for each $h_i \in HOST$ {
$d_i = \frac{count(connection(h_i))}{2}$;
$\gamma_\iota = caculate(vulnerability(h_i))$;
$h_{vi} = v_{info} * v_{sw} * v_{hw}$;
$v_i = \frac{h_{vi}}{\sum_{i=1}^{n} h_{vi}}$;
}
for (i=0;i<n;i++) {
$h_i\_value = \lambda_1 d_i + \lambda_2 n_i + \lambda_3 v_i$;
}
H = Sort (h_value);
KeyHost = KeyHost ∪ H_η;
end

Figure 1. Key host quantization algorithm

3.3 Attack graph generation algorithm

Based on the attack graph search strategy, a locally optimal attack graph generation algorithm is designed and implemented. A description of the algorithm is shown in Figure 2.

Name: Local Optimal Attack Graph Generation Algorithm
Input: initial attribute set P; attack rule set AR; network topology information TP; attack control rule ACL; Target node set KeyHost; Time control initial value TTL; Probability maximum ω_{max};
Output: local optimal attack subgraph SAG(Property,Action, Edge, Control, Type, timeValid).
Process.

```
A = ∅ , E = ∅;
for each h_i ∈ KeyHost (1 < i ≤ k) {
    Goal←Privilege(h_i ,root);
    Control←Get_ACL(h_i );
    Build host_queue;
    host←Get_host(host_queue);
    R←Build attackrule_queue;
    AR = Instantiate(γ, v, p);
    CaculateValue(AR);
    N_G ← N_G ∪ oal ;
    for each n_i ∈ N_G{
      while(TTL) {
      if(ω_i <ω_max ) {
            T_G ← T_G ∪ n_i ;
            T_A ← a|(a ∈ AR, n_i ∈ post(a)) ;
      }
  if (T_A = ∅ ) return TTL;
  else
     for each a ∈ T_A    {
        N_p ← pre(a) ∩ ¬P ∩ ¬P_d //T_p
if((N_p ≠ ∅)&(N_p ∩ T_G = ∅)) {
            for each n_j ∈ N_p   {
                Goal_waif( P,n_j , TTL+1.ω_max ) ;
            Type=success;
            timeValid=TTL;
              }
          if(N_p ≠ ∅ ) continue;
          else {
            N_G ← N_G ∪ (P ∩ pre(a) );
              P_d ← P_d ∪ goal ;
            N_G ← N_G − goal;
            E ← E ∪ {< a, goal > |a ∈ T_A, goal ∈ oal } ;
            E ← E ∪ {< p, a > |p ∈ pre(a)} ;
            P ← P ∪ {P} ∪ P_d ;
            Dot(sub_AG(P,A,E,Control,type,timevalid);
           }
      }
    return (type=failed);
 }
```

Figure 2. Local optimal attack graph generation algorithm

The algorithm uses a search strategy that combines breadth-first and parallel inverse. The time control value for issuing requests is TTL, and since each search request is executed in parallel, the maximum value of TTL is taken as the maximum search time value. Setting the number of hosts to N, the maximum number of query requests issued is N, and the total search time is N× max(TTL), where TTL is a linearly growing discrete value. Therefore, the time complexity of this algorithm is $O(N(\max(TTL)))$

4. Simulation Analysis

The experimental network proposed in this paper is divided into three zones: the extranet, the DMZ zone and the intranet. The DMZ zone is an isolation zone, which serves as a buffer area between the intranet and the extranet, where utilities are deployed.

The vulnerability profile of each host in the experimental network, and the quantitative information on the attacks based on the vulnerability calculation model, are shown in Table 1.

Table 1. Quantitative table of targeted network vulnerability attacks

Mainframe	Exploitable Vulnerability	Risk of attack	Quantitative value of attacks	Successful use of Baseline probability	Consequences of penetration
WS	CVE-2015-1635	570	0.3	0.69	root permissions
FS	CVE-2011-4800	160	0.9	0.81	Trusted relationships
MS	CVE-2008-4250	216	0.7	0.45	root access
MS	CVE-2006-5815	200	0.7	0.61	root access
DB	CVE-2009-4484	135	0.7	0.86	root access
User1	CVE-2012-0458	81	0.5	0.72	user permissions
C1	CVE-2012-0158	167	0.7	0.56	user permissions

4.1 Attack graph generation and visualisation

The attack graph obtained using the attack graph generation method proposed in this paper for the remote attack scenario is shown in Figure 3.

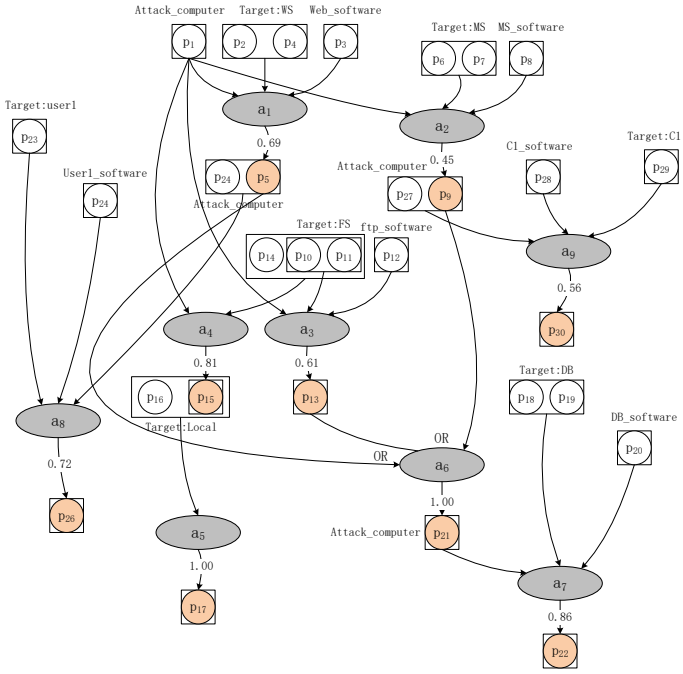

Figure 3. Remote attack graph generation results

The second local attack scenario is simulated by selecting an internal user who has access to the company device (Client1) and has extensive knowledge of the internal facilities (e.g. internal user account names). Based on the host asset computation model, the attacker's goal is to obtain sensitive data Data1, and the generated attack graph is shown in Figure 4. To show the complete attack process,

the atomic attack nodes and important attribute semantics are integrated into a single node representation.

The MaxTries shown in the attack map for the insider attacker in Figure 4 are the maximum number of attempts made when considering that the experimental web server has a security control device.

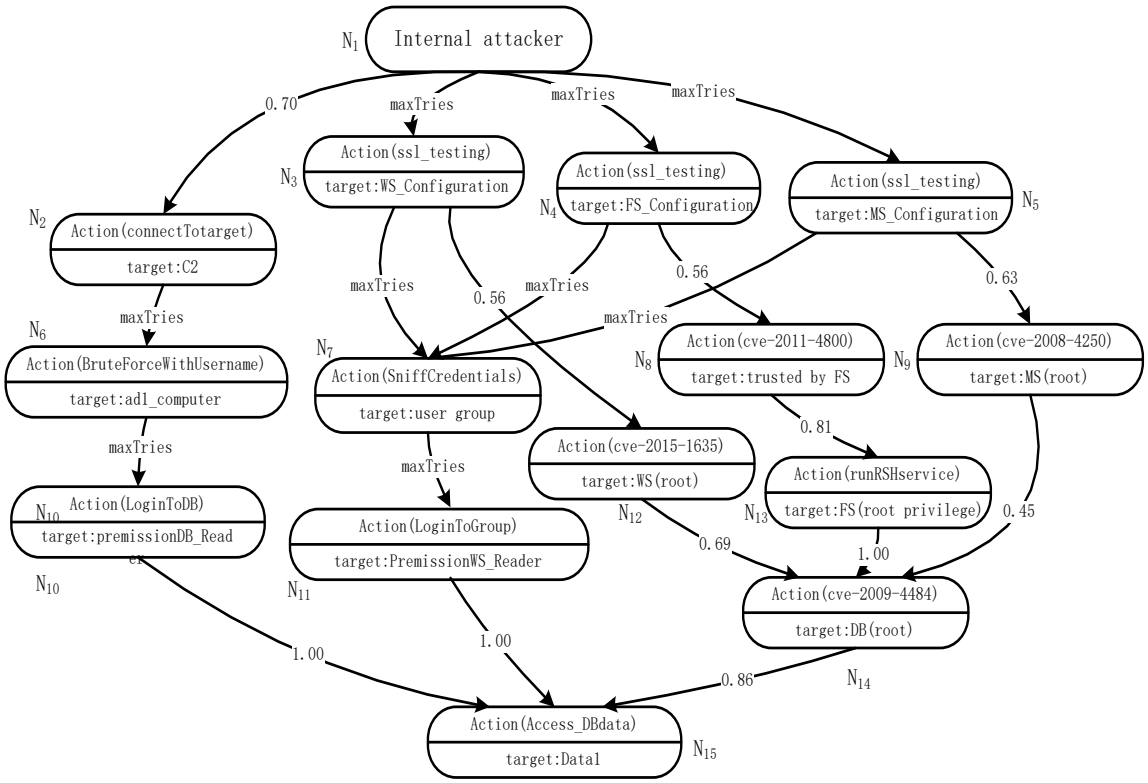

Figure 4. Internal attack map generation results

4.2 Experiment results analysis

For external attackers, it is clear from the attack diagram that whichever attack path is taken, they will need to first break through the defences of the trust area and successfully use the vulnerability information of the servers deployed within it to break through the intranet and access the data of the core servers. Therefore, with good protection of the three servers, the experimental network can be ensured against a successful attack by an external attacker. When the attacker is located internally, the attacker first establishes a connection with the internal client host1 through the internal information in his possession to carry out a brute-force cracking attack; then, through SSL probing, he gradually obtains the server configuration information in the trust area; using the vulnerability points that exist in the configuration, the attacker starts to gain trust in the internal network database server, logs into the Whichever attack path is successful, it results in the loss of the core server DB. The experiments show that the attack graph can automatically simulate the attacker's behaviour and visualise the state of network security, allowing security administrators to sense threats and formulate targeted measures in a timely manner to prevent security breaches from occurring.

5. Conclusion

This paper takes into account the heterogeneity of attackers, sets up two scenarios of remote and local attacks according to different attack locations, simulates and analyses discrete event combinations of complex multi-step attacks, and conducts an example analysis of the proposed attack graph generation algorithm. The experimental results show that the attack graph-based network security assessment method can identify critical vulnerabilities caused by dynamically changing interactions in the system, comprehensively reflect the security status of the target network, and

provide a theoretical basis for administrators to formulate security hardening measures. However, it can be seen from the test scenario that the scale of the network attack graph is already very complex when the target network is considered with only 9 hosts and 7 vulnerabilities. Therefore, from a theoretical point of view, considering that the generation of attack graphs is tightly related to the scalability limitation of large network environments, the dynamic generation of attack paths through step-by-step simulation of attacks, rather than the premise of building a complete attack graph, is an entry point for the study of attack graphs. Finally, the design of an optimization layer that adds automatic detection of security system configurations to the evaluation framework is also a research area worth trying.

References

[1] National Computer Network Emergency Technology Processing Coordination Center. (2020) . Report on China Internet network security in 2021. Beijing: People's Post and Telecommunications Publishing House.

[2] LYE K W, WING J M. (2005). Game strategies in network security. International Journal of Information Security, 4, 71-86.

[3] Phillips, C. and Swiler, L. (1998) A graph-based system for network-vulnerability analysis. Proceedings of the Workshop on New Security Paradigms, Charlottesville, 22-26 September 1998, 71-79.

[4] Ingols K, Lippmann R, Piwowarski K (2006). Practical attack graph generation for network defense. In: Proceedings annual computer security applications conference, ACSAC, pp 121–130. https://doi.org/ 10.1109/ACSAC.2006.39.