Searching security policy with acyclic directed graphs

Xiaorong Cheng[1]*, Sizu Hou[2]

[1]*Department of Computer Science, North China Electric Power University, Baoding 071000, China*

[2]*Department of Electronic & Communication Engineering, North China Electric Power University, Baoding 071000, China*

Received 1 October 2014, www.cmnt.lv

Abstract

To improve the efficiency of policy searching, a method based on the use of a weighted directed graph is studied. Regarding security states as vertices and trigger conditions as edges, the security policy knowledge base can be described as an acyclic weighted directed graph. Firstly, the graph is divided into some areas with just has an initial state node and a termination state node. Secondly, weights for each edge are set according to trigger condition frequencies, and then the optimal path from the initial state node to the termination state node is found using the A* algorithm. Finally, all state nodes are reordered on the basis of their optimal path to build an adjacency matrix and conduct depth-first traversal to search policies. Experiments showed that this method improved policy search efficiency.

Keywords: security policy search, acyclic directed graph, A* algorithm, adjacency matrix

1 Introduction

With the expansion of computer networks and increases in their complexity, information systems face serious security threats. Reasonable and effective security policy can ensure that network devices work together, so that an information system can deal with a series of security incidents. The security policy is event-driven and enforced by changing the configuration of security devices. Based on the model of a state machine, security devices or their related functions can be regarded as security states, and the security policy can be described as changes in these security states under special trigger conditions. Combined with the frequency of these trigger conditions, the set of security policies can be created as an acyclic weighted directed graph. On the basis, this paper discusses improvements to the efficiency of search policy rules on a directed graph.

The complexity of searching a security policy depends on how its description. A security policy can be expressed as a Datalog program [1], by using the Datalog semantic query algorithm to implement searched thereon. Based on first-order logic, a security policy can also be expressed by well-founded semantics [2], by introducing the SLG algorithm [3, 4] to realise policy searching. In this research, a direct graph was used to express the policy knowledge base instead of creating a logical system through well-founded semantics. The method of searching policies by traversing the graph was simpler and more efficient.

2 Definition of the security policy and system operation principle

A security policy is a set of rules, which control the security management of people and resources [5]. A security policy expresses multiple requirements related to system security while not all these requirements can be characterised by security models [6]. This paper describes a security policy with $P=<D,C,R>$, where D represents the security domain. C represents the trigger condition which consists of a security incident and a system incident (only security incidents are shown in the acyclic directed graph). R represents the rule set (each rule represents a configuration action described by $r=<i,o,s_{0,}l>$). i represents the former state. o represents the output state (state transition can be described as $i(C)\rightarrow o$). s_o represents the initial state of the security domain, and l represents whether the output state of policy is a final state or not. All state transition rules in the system constitute the policy knowledge base, which can be described as an acyclic directed graph containing one initial state node and some termination state nodes: it is shown in Figure 1, which shows part of the rules as based on policy within the policy knowledge base.

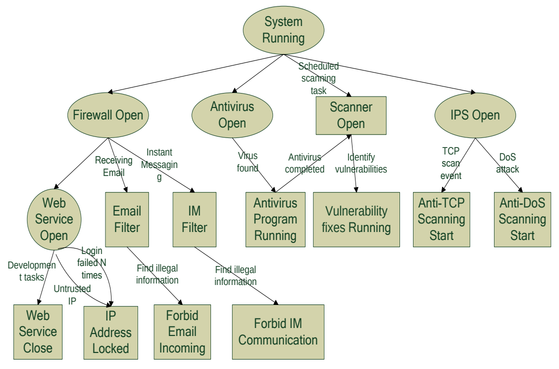


FIGURE 1 Part of the rules in the knowledge base described as a directed graph

*__Corresponding author__ e-mail: cheng3100@sohu.com

There is a current state set in the system, which is used to save the active states, and a state will be deleted from the current state set when it changes under the conditions imposed by the security incident. The system operation principle is shown in Figure 2. When the system receives notification of a security incident, it searched the former state and determined whether the former state belonged to the current state set. If the current state set contained the former state, then system distributed the security policy to related devices and enforced it, otherwise no action was taken. When policy enforcement was complete, the output state of the policy was merged into the current state set. Sometimes the trigger condition would be a system incident such as "system running" provided that the trigger condition continued in the system, then its output state would always remain in the current state set.

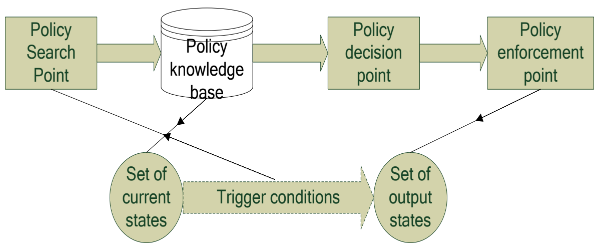


FIGURE 2 System operation principle

As shown in Figure 2, policy searching plays an important role in system operation: it is the premise of policy enforcement. In the direct graph, depth-first traversal or breadth-first traversal can be used when searching policies: both methods required an adjacency matrix. A path from the initial state node to the termination state node forms a workflow, considering the association among states in the same workflow; depth-first traversal was a better choice. When building the adjacency matrix, the sequence of nodes determined the nodal access order in depth-first traversal. To give priority to nodes, which stand for high frequency security incidents, the weights of the graph and the method of arranging the nodes were combined: this was feasible when building the adjacency matrix as proposed.

3 Method of searching security policy based on weighted direct graph

3.1 INSTANCE OF WEIGHTED DIRECTED GRAPH

There is no loop in the directed graph, which is created according to the knowledge base. The graph only has an initial state node (the in-degree of the node is zero) and some termination state node (the out-degree of the node is zero). With a small-scale knowledge base (e.g. Figure 3), V1 represents the initial state node, V10, V11, and V12 are termination state nodes, c1 to c16 are trigger conditions, the frequency of trigger conditions is described by an integer from 1 to 10, where 10 stands for the highest occurrence probability and 1 the lowest.

Using an adjacency matrix to describe this directed graph, each item in the matrix was represented by the name of its trigger condition and event frequency. In accordance with the sequence from V1 to V12, the weighted direct graph was transformed into a matrix as follows:

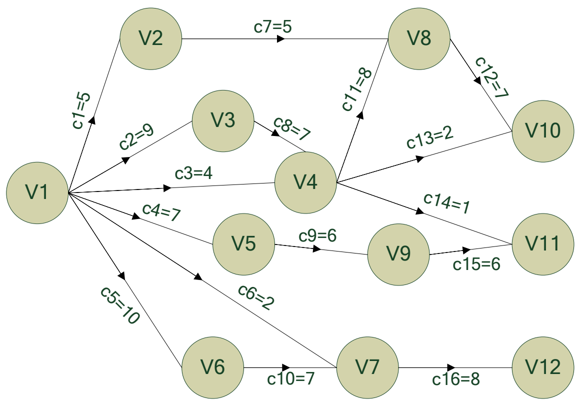


FIGURE 3 A small-scale policy knowledge base represented by a weighted direct graph

$$A=\begin{bmatrix}
null & c1,5 & c2,9 & c3,4 & c4,7 & c5,10 & c6,2 & null & null & null & null & null \\
null & null & null & null & null & null & null & c7,5 & null & null & null & null \\
null & null & null & c8,7 & null & null & null & null & null & null & null & null \\
null & null & null & null & null & null & null & c11,8 & null & null & c13,2 & c14,1 \\
null & null & null & null & null & null & null & null & c9,6 & null & null & null \\
null & null & null & null & null & null & c10,7 & null & null & null & null & null \\
null & null & null & null & null & null & null & null & null & null & null & c16,8 \\
null & null & null & null & null & null & null & null & null & c12,7 & null & null \\
null & null & null & null & null & null & null & null & null & null & c15,6 & null \\
null & null & null & null & null & null & null & null & null & null & null & null \\
null & null & null & null & null & null & null & null & null & null & null & null \\
null & null & null & null & null & null & null & null & null & null & null & null
\end{bmatrix}$$

If the frequency of the trigger condition was not taken into account when building the adjacency matrix, then depth-first traversal would not be conducive to a timely response to high frequency events, especially when the scale of the directed graph were large. As shown in Figure 3, the frequency of c16 was relatively high in all trigger conditions while almost all state nodes needed to be traversed to find the former state (V7) of c16. So the sequence of nodes when building the adjacency matrix was important when improving the efficiency of a policy search. Early detection, early implementation methods of arranging the nodes entailed two key steps: division of the graph into sub-areas, each containing one initial state node and one termination state node and discovery of the optimal path for each area using the A* algorithm. On the basis of each optimal path, all state nodes were reordered.

3.2 DIVIDE THE DIRECT GRAPH

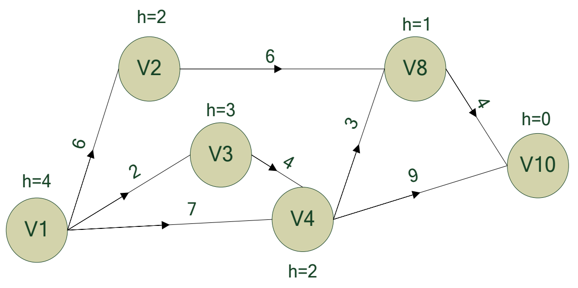
Starting with one of the termination state nodes, the previous state of that node was successively searched until reversion to its initial state node. All state nodes were visited from the termination state to initial state node to form a division of the directed graph. The number of termination state nodes on a graph was the number of areas after sub-division. This was realised by a recursion method.

For a direct graph with n nodes and e edges, the time complexity of the algorithm was $\Omega(n+e)$. By executing the algorithm, all relevant state nodes from the initial state node to the termination state nodes can be found. The graph was sub-divided as shown in Figure 3 into three areas, and the node sequence was output in reverse order as follows:

{{V1,V2,V3,V4,V8,V10},{V1,V3,V4,V5,V9,V11},{V1, V6,V7,V12}}

3.3 FIND THE OPTIMAL PATH OF EACH AREA AND REORDER ALL STATE NODES

The optimal path was that most likely to have been chosen from the initial node to the termination node corresponding to a series of higher frequency security events. Using the A* algorithm to find the optimal path required alteration of the optimal path to the minimum dissipation path. In the weighted direct graph, value of arcs represented the frequency of security events, the higher the frequency, the smaller the dissipation. The frequency values ranged from 1 to 10, so the conversion formula from frequency to dissipation value was set to $g_i = 10 - c_i + 1$ where g_i represents the dissipation value and c_i represents the frequency. Meanwhile, to reduce the number of nodes in this extended search, a heuristic function was required. The value of this heuristic function h was set to encompass the maximum number of steps to a given termination node. One of the sub-areas of the graph shown in Figure 3 was chosen, marking the heuristic function value of each node and the dissipation value of each arc, as shown in Figure 4.



FIUGRE 4 One graph sub-division showing its heuristic function value and dissipation value

The A* algorithm [7] was used to find the optimal path from initial node V1 to termination node V10: Table 1 shows such the search table thus created.

TABLE 1 A* algorithm search table

OPEN`	CLOSED
Initialisation(V1(4))	()
1 (V2(8) V3(5) V4(9))	(V1(4))
2 (V2(8) V4(8))	(V1(4) V3(5))
3 (V4(8) V8(13))	(V1(4) V2(8) V3(5))
4 (V8(10) V10(15))	(V1(4) V2(8) V3(5) V4(8))
5 (V10(13))	(V1(4) V2(8) V3(5) V4(8) V8(10))
End of search	

According to the search result, the optimal path was V1 →V3 → V4 →V8 → V10, and the dissipation of the path was 13. Similarly, for the division from V1 to V11, the optimal path was V1 → V5 → V9 → V11, the dissipation value was 14, and for the division from V1 to V12, the optimal path was V1 → V6 → V7 → V12 and the dissipation value was 8. According to the dissipation value of each optimal path, the division of nodes was arranged in ascending order:

{{V1,V6,V7,V12}, {V1,V3,V4,V8,V2,V10}, and {V1,V5,V9,V3,V4,V11}}.

The first node was removed from each set to form the final node arrangement, if the node was already in the final arrangement, then the original node was overwritten to give the termination nodes for each division. After nodal arrangement the set was:

{V1,V6,V3,V5,V7,V4,V9,V8,V2,V12,V10,V11}.

According to this nodal arrangement, the adjacency matrix was rebuilt as follows:

$$A' = \begin{bmatrix} null & c5,10 & c2,9 & c4,7 & c6,2 & c3,4 & null & null & c1,5 & null & null & null \\ null & null & null & null & c10,7 & null & null & null & null & null & null & null \\ null & null & null & null & null & c8,7 & null & null & null & null & null & null \\ null & null & null & null & null & null & c9,6 & null & null & null & null & null \\ null & null & null & null & null & null & null & null & null & c16,8 & null & null \\ null & null & null & null & null & null & null & c11,8 & null & null & c13,2 & c14,1 \\ null & null & null & null & null & null & null & null & null & null & null & c15,6 \\ null & null & null & null & null & null & null & null & null & null & c12,7 & null \\ null & null & null & null & null & null & null & c7,5 & null & null & null & null \\ null & null & null & null & null & null & null & null & null & null & null & null \\ null & null & null & null & null & null & null & null & null & null & null & null \\ null & null & null & null & null & null & null & null & null & null & null & null \end{bmatrix}$$

A depth-first traversal was run using the adjacency matrix above: nodes representing high frequency events were always searched first so the method could improve the overall efficiency of a policy search.

4 Experimental verification

An experiment was designed to verify the advantages of this policy search method. The experiment was run with the Java programming language on a computer with a Core 2 Duo processor and 2 GB of random access memory. Some 26 sample of security incidents were generated according to frequency. The 26 trigger conditions were searched in the original adjacency matrix of the directed graph and in the adjacency matrix after nodal reordering. The time required under both cases was recorded (see Figure 5). It was seen from the result of this experiment that, with the expansion of the direct graph, the advantages of the proposed policy searching method were obvious.

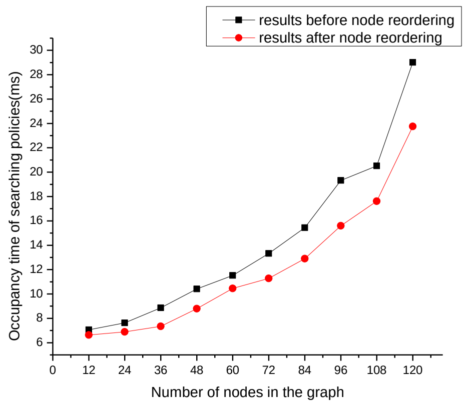


FIGURE 5 Comparison of the times required for searching the security policy: the case for nodal reordering

5 Conclusions

This research into methods of improving the efficiency of searches in a security policy based on direct graph depth-first traversal was successful. By means of a state machine, the policy knowledge base could be described as an acyclic weighted direct graph with a single initial state node and some termination state nodes.

Combined with the reverse topological sorting algorithm and the A* algorithm, and according to the frequency of each trigger condition, all state nodes were reordered before creating the adjacency matrix for a given graph. As a result, nodes associated with high frequency security events could be searched first. The experimental result showed that the proposed method could improve search efficiency within a security policy.

References

[1] Bonatti P A, Shahmehri N, Duma C, Olmedilla D, Nejdl W, Baldoni M, Baroglio C, Martelli A, Patti V, Coraggio P, Antoniou G, Peer J, Fuchs N E 2004 Rule-Based policy specification: State of the art and future work *Technical Report IST506779 Project Deliverable D1 Working Group I2*

[2] Bao Y, Yin L, Fang B, Guo L 2012 Approach of security policy expression and verification based on well founded semantic *Journal of Software* 2012 **23**(4) 912-27

[3] Chen W D, Warren D S 1996 Tabled evaluation with delaying for general logic programs *Journal of ACM* **43**(1) 20-74

[4] Chen W D, Warren D S 1993 Query evaluation under the well-founded semantics *Proc 12th ACM SIGACT-SIGMOD-SIGART Symp. On Principles of Database Systems Washington*

[5] Tang C, Yu S 2009 Verifying Network Security policy based on features *Journal of Computer Research and Development* **46**(11) 1854-61

[6] Wen H, Zhou Y, Qing S 2005 A formal commercial security policy model based on framework *Chinese Journal of Electronics* 2005 **33**(2) 222-6

[7] Ma S, Zhu X 2004 Artificial Intelligence *Tsinghua University Press the first edition* 34-46

Authors

Xiaorong Cheng, born on April 25, 1963, Handan, Hebei province, China

Current position, grades: North China Electric Power University, professor.
University study: Master of Engineering of North China Electric Power University, Power Systems and Automation, 1994.
Research activities: computer network technology, network information security.
Professional Activities and Memberships: Network Security Research, GIS-based Ethernet management software, Panjin Power Administration Cable Management Software, Concentrate on the practice teaching reform and research.

Sizu Hou, born on May 23, 1962, Yuncheng, Shanxi province, China

Current position, grades: North China Electric Power University, professor.
University study: Master of Engineering of North Jiaotong University, specializing in communication and electronic systems, 1988.
Research activities: power systems communication technology, information and communication engineering.
Professional Activities and Memberships: Broadband Power Line Communication Technology Research, Communications network monitoring and management system, Integrated Network Management System.