Concept of Intrusion Detection System with Implementation of Topological Sorting

Orest Lavriv [1], Zenoviy Kharkhalis[1], Bohdan Strykhaliuk[1]

1. Department of Telecommunicatins, Lviv Polytechnic National University, UKRAINE, Lviv, 12 S. Bandery str., email: orest.a.lavriv@lpnu.ua , zenoviy.m.kharkhalis@lpnu.ua

Abstract: **Cyberprotection topic will remain relevant as the network traffic, it's types and types of Internet-connected elements growth will continue to increase during the future decades. New data protection systems with previously unknown or unused functional anomalies detection systems have some advantage time before intruders figure out a way to bypass the defense which proves the relevance of these systems. In this paper we propose a concept of intrusion detection system with the specific implementation of topological sorting methods. We also explain the mathematical background of topological sorting and outline the possible future research directions within this area.**

Keywords: **cyberprotection, topological sorting, directed acyclic graph, algorithm.**

I. INTRODUCTION

In our recent work we outlined a large number of various threats concerning telecommunication networks [1] (see Table 1). These threats are in some way similar for all kinds of telecommunication systems varying only in the scale of influence on one hand or on the scale of intrusion detection system view angle on the other.

As we are now on the informational defense side, it is natural for us to be interested in the situation where the so called above "scale of intrusion detection system view angle" would be always wide enough to prevent and counteract to all kinds of malicious influences threatening our users' data. Thus, it is crucial for us to stay aware and consider and apply new and promising information protection and intrusion detection methods for the telecommunication systems our data traverses through.

In this paper we propose a concept of an intrusion detection systems based on processing of graph representations of content delivery network as a new and fast way to detect and counteract to anomalies in content delivery process. The ability to unambiguously interpret the changes I graph representation of content delivery network lies in the background of the concept.

TABLE 1. PAGE LAYOUT DESCRIPTION [1]

THREATS	SOURCES	ORGANIZATIONAL LEVEL	REASONS	GOALS	LEVEL OF THREAT FOR STATE
Hacking	Individuals, groups	Low, sometimes with national/international coordination	Individual, sometimes political and patriotic	Development of own skills, promotion of an attitude, values	Very low
Cybercrime	Individuals, groups	Low, sometimes with national/international coordination	Individual	Personal benefits	Medium, strenuous for business activities
Cyberterrorizm	Individuals, groups	High level of organization and coordination	Political	Serious destructions, psychological effect	High, main goal are civilian and military infrastructures
Cyberspying	States and public organizations	High level of organization and coordination	Political, economic	Acquisition of secret data	High, threats for data important for a state's security
Military use of cyberspace	States	Very high level of organization and coordination	Political, military	Military goals e.g. destruction of a specific object	Very high, connected with military operations, possible consequences: physical destructions, death of people

Nowadays, topological sorting had found its application as an important aspect in the scheduling of jobs such as instruction scheduling, determining the order of compilation tasks to perform in makefiles (a file containing a set of directives used with the make build automation tool), resolving symbol dependencies in linkers, and deciding in

978-1-5090-5045-1/17/$31.00 ©2017 IEEE

which order to load tables with foreign keys in databases. Most users of Debian-based UNIX systems have used topological sorting indirectly by invoking the apt-get command, which uses topological sorting in order to determine the best way to install or remove packages [2]. Another example of using topological sorting in research is using it to find the best ways to schedule a large amount of jobs in distributed networks [3].

II. Topological Sorting Basics

In science, topological sorting is presented in the form of specific algorithms for sorting tops directional acyclic graph vertices to obtain a sample in which the vertices will be placed in order of execution of logical operations within a process. Algorithms differ among themselves only by implementation and, therefore, mathematical complexity. This is what many scientists are struggling for in the field of topological sorting.

In mathematic topological sorting is the problem of finding a permutation of N objects $\{a_i\}_{i=1}^{N}$ which conforms to a partial order defined by a binary relation R having the properties of asymmetry and transitivity [4].

From the graph point of view it is a complex of operations for finding a linear order of vertices in a way such that if the edge uv exists between node u and v and is directed from u to v, u comes before v in the sorted set [5]. Topological sorting is applicable only to graphs with no cycles and with directed edges. This type of graph is called the Directed Acyclic Graph or DAG for short and it is used in many applications to indicate precedence among events. For example, applications of DAGs include the following:

- Inheritance between C++ classes or Java interfaces;
- Prerequisites between courses of a degree program;
- Scheduling constraints between the tasks of a projects.

Thus, a DAG is good for modeling processes and structures that have a partial order: $a > b$ and $b > c$ imply $a > c$. But may have a and b such that neither $a > b$ or $b > a$. One can always make a total order (either $a > b$ or $b > a$ for all $a \neq b$) from a partial order. In fact, that's what a topological sort will do.

The most common approach to performing a topological sort is to implement an algorithm based on either breadth first search (BFS) or depth first search (DFS). The DFS approach was first described by Tarjan in 1976 [6]. However, research has shown that parallelization of DFS algorithms is extremely difficult and performance gains are limited and hard to achieve [7]. The BFS approach was first described by Kahn in 1962 [5].

An example of a topological sorting for a DAG is given below. The sorting had been performed according to the legacy algorithm proposed by Kahn, thus it's the BFS. The following steps are involved in the algorithm:

Step 1: Compute in-degree (number of incoming edges) for each of the vertex present in the DAG and initialize the count of visited nodes as 0.

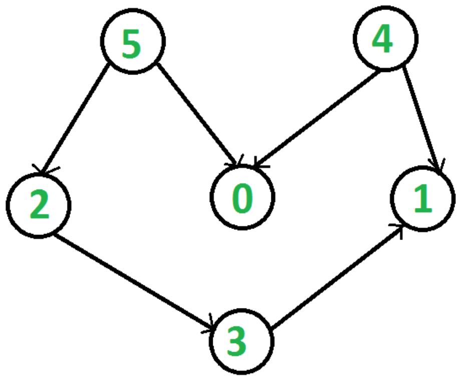
Step 2: Pick all the vertices with in-degree as 0 and add them into a queue (Enqueue operation).

Step 3: Remove a vertex from the queue (Dequeue operation) and then:

– Increment count of visited nodes by 1.
– Decrease in-degree by 1 for all its neighboring nodes.
– If in-degree of neighboring nodes is reduced to zero, then add it to the queue.

Step 4: Repeat Step 3 until the queue is empty.

Step 5: If count of visited nodes is not equal to the number of nodes in the graph then the topological sort is not possible for the given graph, because it is not acyclic.



Source: geeksforgeeks.org

Fig.1. Sample graph for the example

Having these steps performed we obtain a set of graph vertices which are permuted according to topological order.

5 4 2 3 1 0

There can be more than one topological sorting for a graph. For example, another topological sorting of the following graph is:

4 5 2 0 3 1

The first vertex in topological sorting is always a vertex with in-degree as 0 (a vertex with no in-coming edges). Thus, the valid topological sorts for the graph depicted above are:

```
4 5 0 2 3 1
4 5 2 0 3 1
4 5 2 3 0 1
4 5 2 3 1 0
5 2 3 4 0 1
5 2 3 4 1 0
5 2 4 0 3 1
5 2 4 3 0 1
5 2 4 3 1 0
5 4 0 2 3 1
5 4 2 0 3 1
5 4 2 3 0 1
5 4 2 3 1 0
```

As we can see, with application of some additional conditions to graph traversing during the process of topological sorting we may apply rules according to which the ordered permutations will be obtained and, thus, unambiguously determine the graph it has been obtained from.

III. The intrusion detection system concept

This brings us to the basis of the concept we present in this paper. With the adequate representation of the service delivery structure we are interested in as a DAG we may detect anomalies in this system's functioning process (intrusions, node failures) by performing a topological sorting of a graph representing this service delivery structure. And by performing exactly topological sorting and not any other type of graph traversing the accurate location of the functioning anomaly may be detected by the intrusion detection watcher and corresponding measures may be applied to prevent any consequences the anomalies may cause.

As Martin Eklund and David Svantesson from KTH Royal Institute of Technology had proven in their latest research [8], topological sorting is a rather complex computing task with a linear computing time ($\mathrm{O} = \left(|V| + |E|\right)$, where V – set of graph vertices, E – set of graph edges) for the presented above Kahn's algorithm. However, it may be performed relatively efficiently for large, shallow graphs. Thus the service delivery system integrity check may be performed comparatively often and remain relevant for high performance systems.

$$S^t - S^{t-1} \begin{cases} = \varnothing, then & \text{proceed in normal mode} \\ \neq \varnothing, then & \text{initiate event handling subsystem} \end{cases} \quad (1)$$

Therefore, upon detection of topological anomaly on the graph, by performing the comparison of array obtained as a result of topological sort in the current moment S^t with the array representing the previous moment S^{t-1}, the system may engage the event handling subsystem to define the type of event occurred and measures intended to be taken.

IV. Conclusion

Summing up the abovementioned, here are presented only the basics of the proposed concept. With addition of more functions the presented concept may develop into sophisticated intrusion detection system applicable for modern content delivery networks. The following features deserve a more thorough investigation:

- A systems graph representation complexity may be reduced using hypergraph theory, therefore reducing the certain vertices and edges groups to a single node, changing the scale of observation;
- The methods of transforming directed cyclic to directed acyclic graphs, since topological sorting may only be applied to DAG;
- The instance determining the event types that occurred in the system and deciding on safety measures to prevent damage to user data or system itself is a separate topic and a background for the future research.

References

[1] P. Yatskiv, O. Lavriv, Z. Kharkhalis, V. Mosorov, M.Niedzwiedzinski, "Security Provisions for Ensuring Telecommunication Networks' Protection from Malicious Influences", Agile Commerce – zarządzanie informacją i technologią w biznesie, tome XVII, vol. 11, part 1, pp. 111-125.

[2] R. Fox., "Software Installation and Maintenance" in *Linux with operating system concepts*, Boca Raton, CRC Press, 2014, ch. 13, sec. 4.6, pp. 544-546.

[3] A. Jarry, H. Casanova and F. Berman, "DAGsim: A simulator for DAG scheduling algorithms", LIP, Lyon, Rep. 2000-46, Dec. 2000.

[4] Y. L. Varol, D. Rotem, "An Algorithm to Generate All Topological Sorting Arrangements", The Computer Journal, vol. 24, no. 1, pp. 83-84, Feb. 1981.

[5] A. B. Kahn, "Topological sorting of large networks", Communications of the ACM, Vol. 5, no. 11, pp. 558-562, Nov 1962.

[6] R.E. Tarjan. "Edge-disjoint spanning trees and depth-first search", Acta Informatica, vol. 6, no. 2, pp. 171-185 June 1976.

[7] P. Harish and P.J. Narayanan, "Accelerating large graph algorithms on the GPU using CUDA", in *High performance computing–HiPC 2007 14th Interantional Conference Proceedings, Goa, India, December 18-21*, 2007 © Springer-Verlag Berlin Heidelberg. doi: 10.1007/978-3-540-77220-0_21

[8] M. Eklund, D. Svantesson. "A naive implementation of Topological Sort on GPU", Degree project in computer engineering. KTH Royal Institute of Technology, Stockholm, Sweden, 2016.