ISSN: 3078-5367 **DOI:** https://doi.org/10.69513/jncs.v2.i2.a2

Al-Noor Journal for Information Technology and Cybersecurity

https://jncs.alnoor.edu.iq/

Simplify Attack Graph to Reduce Attack Graph Complexity Using Critical Path Preserving Graph Reduction

[1]Zaid J. Al-Araji, [1]Balqees Talal Hasan, [1]Fahad Ahmed Shaban, [2]Hussein M. Farhood, [3]Zaid Ali Abdulkadhim, [4]Sharifah Sakinah Syed Ahmad, [5]Ahmed A. Idris, [6]Najwa N. Hazem Al-Sheikh, [1]Ahmed Hayali

[1]Department of Computer Network and internet, College of Information Technology, Ninevah University, Iraq.
[2]Department of Computer Engineering, College of Engineering Mustansiriyah University, Iraq.
[3]Al-Forat Al-Awsat Technical University (ATU), Al-Diwaniyah Technical Institute, Iraq.
[4]Faculty of Information Communication Technology, Universiti Teknikal Malaysia Melaka, Malaysia.
[5]University of Al-Hamdaniya, Iraq
[6]Technical Computer Engineering Department, Al-Hadba University, Iraq.

Article information

Article history:
Received: August, 31,2025
Revised: September, 20,2025
Accepted: November, 21,2025

Keywords:
Attack Graph
Network Security
Graph Pruning
Critical Path Preserving Graph Reduction

Correspondence:

Zaid J. Al-Araji
zaid.jasim@uoninevah.edu.iq

Abstract

In recent years, network technologies have grown more widely used. The network has security issues that need to be fixed, despite the fact that it is advantageous for people to live and work there. Among these problems are cyberattacks. As more devices connect to the internet, hackers' attack surface expands. Attack graphs are one of the many techniques that have been put forth recently to identify and forecast attacks. Predicting the attack and its next move within the network is the main objective of creating the attack graph. However, there are a few problems with the attack graphs that are currently in use. The primary problem with attack graph construction is scalability. In order to minimise the level of complexity of the attack graph, the present research suggests employing personal agents to shorten the reachability time when calculating between the nodes and the critical path preserving graph reduction technique to eliminate superfluous edges. The results demonstrate that the suggested performance outperforms the attack graph that is currently in use. The attack graph complexity and generation time were decreased by the suggested attack graph.

DOI: https://doi.org/10.69513/jncs.v2.i2.a2

1. Introduction

People's lives have been significantly altered by the exponential rise of computer networking technologies [1], [2]. In order to transcend time and distance, the networks have been actively involved in every facet of life, which has immensely benefited humanity. According to [3], the numbers of the devices that connected to the internet is increasing yearly, as shown in Figure 1.1. In 2010, the number of devices was around eight billion; in 2021, the number became 10 billion, which increased by 20% more than in 2010. The network becomes more vulnerable as a result of the rising utilisation of the devices that are connected to the internet. Vulnerabilities are defects in a system's architecture that give an outsider the ability to carry out orders, acquire unauthorised information, and initiate a variety of assaults [4, 5, 6]. In 2020, there were more than 18362 vulnerabilities, according to the National Vulnerability Database (NVD). Additionally, this is more noteworthy than in prior years (17,382 in 2019 and 17,252 in 2018) [7].

Attacks on businesses and personal networks have escalated as a result of these vulnerabilities throughout time [8]. Despite Malaysia's strong commitment to the cyber security and its position as

ISSN: 3078-5367 **DOI:** https://doi.org/10.69513/jncs.v2.i2.a2

the third securest nation in the world, there were 6,274 documented cyberattacks in 2017 [9]. Given the volume of attacks, cyberspace cannot be completely secure. Consequently, as the world's reliance on information technology and the internet develops, cyber security is becoming more and more crucial. In network security analysis, an effective technique for representing all the potential routes attackers could use to enter a network is called an attack graph. The attack graph concept was initially suggested by [10]. An attack graph is a model that shows all potential ways an attacker could break a security policy by taking advantage of connected flaws. In addition to the security policy, host connections and vulnerabilities make up the minimal set of data required to develop an attack graph [11]. The vertices and directed edges constitute an attack graph, with vertices signifying network states and edges signifying conversions between states. Using an attack graph, a network administrator can intuitively see how the network transitions between states [12]. In network attacks, some states refer to the objectives of the attacker. By identifying them in the analysis of an attack graph, security experts can implement hardening measures that prevent attackers from accomplishing their objectives. To construct an attack graph, the security engineer must identify network vulnerabilities, ascertain the requisite conditions for their exploitation, and establish the causal connections between these factors and the vulnerabilities. However, the attack graph performance still suffers from different issues, the generation time is the main challenge of the attack graph [13], [14], [15]. To overcome this issue, in this paper critical path preserving graph reduction (CPPGR) algorithm are used to simplify the attack graph which reduce the complexity and personal agent to calculate the reachability of each node individually which reduce the reachability calculation time which lead to reduce the generation time of the attack graph. The rest of the paper is organized as section 2 is the related work, section 3 is proposed model, section 4 is the experiment and results, and section 5 is the conclusion.

2. Related Work

Several attack graphs have lately been created employing diverse procedures and techniques to enhance their effectiveness. This section will address the latest assault graphs.

[16] presents a network segmentation-based scalable security state (S3) solution. The framework divides the vast network region into smaller, more manageable chunks using the well-known divide-and-conquer tactic. Additionally, [17] provides a graphical model that describes vulnerability-based attacks against IoT edge devices using several attackers and targets. suggests a new technique for improving the detection of cyberattacks in power grid operational technology (OT) networks [18]. In order to identify irregularities in OT communication traffic, the authors provide a hybrid deep learning model that combines a deep convolutional network for time series classification (TSC) with Graph Convolutional Long Short-Term Memory (GC-LSTM). By giving power system operators near-real-time situational awareness, the technique seeks to help them locate and detect active cyberattacks. [19] introduces an innovative methodology for threat modeling in dynamic Internet of Things (IoT) systems by the utilization of dynamic attack graphs. Conventional attack graph methodologies are tailored for static settings, such enterprise networks, and are ill-equipped for IoT environments, which are marked by frequent topological alterations. The suggested methodology utilizes graph database management tools (GDBM), particularly Neo4j, to model and analyze attack vectors in real-time as the IoT landscape evolves. [20] introduces a new Simplified Gradient-based Attack (SGA) architecture for executing adversarial assaults on Graph Neural Networks (GNNs), with a specific focus on the node classification job. The authors tackle the issues of scalability and assessment in adversarial assaults on extensive graphs, prevalent in practical applications. [21] introduces IDERES (Intrusion Detection and Response System), an innovative system aimed at bolstering the security of IoT networks through the integration of network analysis, machine learning (ML), and attack graphs (AG). The system seeks to identify and react to both recognized and unidentified (zero-day) assaults instantaneously, tackling the escalating security threats associated with the expansion of IoT devices. [22] ADSynth is a program developed to create realistic Active Directory (AD) attack graphs for research, development, product testing, and education in AD security. Active Directory is an essential element of numerous firms' IT infrastructure and is often a target for hackers. ADSynth seeks to rectify the deficiency of authentic AD data for research purposes by producing extensive, realistic AD graphs that encompass prevalent misconfigurations and security concerns. [23] presents an optimal attack detection method for Bayesian Attack Graphs (BAGs) under conditions of uncertainty in monitoring and reimaging. BAGs are probabilistic graphical models used to represent the progression of cyberattacks in complex interconnected networks. In conclusion, even though several techniques have been employed to enhance attack graph development, there are still certain problems with attack graphs, particularly scalability, which affects the attack graph generation time.

3. Proposed Model

Reachability analysis, attack graph production, and attack graph analysis by optimisation and attack graph reduction are the three stages that make up the attack graph generation in this work, as illustrated in

ISSN: 3078-5367 **DOI:** https://doi.org/10.69513/jncs.v2.i2.a2

Figure 1. Each step will be thoroughly discussed in the sections that follow.

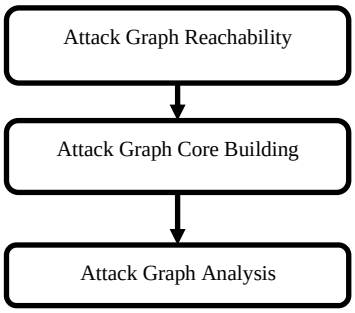

Figure 1: Attack graph steps

3.1 Reachability

Determining reachability is a difficult and complex task. It identifies routes between target and source nodes utilizing network topology data. All filtering devices within the networks must have their rulesets imported and emulated. Multiple methodologies exist for determining reachability. This study will calculate the reachability between nodes in the network using a matrix and a multigraph. The matrix is utilized to identify the relationships among the nodes via their IP addresses, whereas the multi-graph serves to illustrate the network structure and its vulnerabilities.

The accessibility of the applications on the target network is determined upon reachability. The requirements for reachability are dictated by firewall filtering rules, trust relationships, router access control lists, and software applications. All of these aspects will be taken into account to design our reachability multi-graph.

A personal agent collects these components from the nodes, assesses the node's reachability, and transmits the findings to the administrator server, responsible for generating the attack graph. Each personal agent is responsible for collecting information to assess reachability; the personal agent is also tasked with updates to minimize time complexity. The administrator server will be responsible for monitoring information source utilization, vulnerability exploitation, and attacker permissions across many software applications.

The graph vertex denotes the nodes inside the network. A graph edge represents a collection of source and target software applications, wherein the source may access the target if specific conditions are satisfied. The conditions are preserved at the edge, facilitating direct accessibility between software programs. Examples of such scenarios include port numbers, network protocols, and user passwords.

3.2 Attack Graph Core Building

Basically, the attack graph core building is the main step or the core of the attack graph. After calculating the information that needed by the personal agent in each workstation and send it to the administrator, the administrator workstation starts generating the attack graph by connecting each node with other workstations based on the routing table that generated by each workstation. After the generation, the reduction of the complexity of the graph based on the critical paths will be started using Critical Path-Preserving Graph Reduction.

3.2.1 Critical Path-Preserving Graph Reduction

This algorithm simplifies an attack graph by removing redundant nodes and edges while preserving the critical attack paths that represent the most significant threats. It uses a combination of graph pruning and path ranking to achieve this. Unlike traditional graph reduction techniques, this algorithm explicitly prioritizes critical attack paths, ensuring that the most significant threats are preserved. By pruning redundant nodes and edges, the algorithm reduces the size of the graph without sacrificing important information. The algorithm steps are explained as shown in Figure 2.

An attack graph $G = (V, E)$, where V is the set of nodes and E is the set of edges.
Start:
Step 1: Identify Critical Paths
1. Use a shortest-path algorithm
2. Rank these paths based on a risk score

Step 2: Prune Redundant Nodes
Step 3: Merge Similar Paths
Step 4: Simplify Edges
1. Remove edges that do not contribute to any critical path
2. Retain only the edges that are part of the top k critical paths

Output: A simplified attack graph $G' = (V', E')$

Figure 2: Critical path-preserving graph reduction steps

The CPPGR algorithm is designed to simplify attack graphs while preserving the most critical attack paths. It begins by identifying all paths from the attacker's starting point to critical assets using a shortest-path algorithm which in this work used Dijkstra's. The paths are thereafter graded according to a risk score that evaluates criteria like the severity of vulnerabilities as described by the CVSS established by the NVD organization. The top k paths, determined by a user-defined parameter, are identified as the critical paths to be retained. The technique subsequently eliminates unnecessary nodes and edges that do not reside on any key paths, therefore decreasing the graph's size. To enhance the graph's clarity, paths with shared sub-paths are consolidated, minimizing repetition while preserving the general framework. The end output is a simplified attack graph that preserves only the nodes and edges vital to the critical paths, facilitating analysis and visualization.

4. Experiment and Results

The experiment was conducted on a real-time network to assess and confirm our methodology. A core i7 2.8 GHz processor, 16 GB of RAM, Windows 11 as the operating system, and Microsoft Visual Studio 2022 for coding were the specifications utilised in the experiments to evaluate the creating attack graph in this environment.

ISSN: 3078-5367 **DOI:** https://doi.org/10.69513/jncs.v2.i2.a2

According to the findings of using a personal agent to collect the data and relay it to the administrator, there are no missing data when the attack graph is updated if any information has changed. The attack graph's complexity and generation time are reduced by employing a CPPGR technique to simplify it.
Consequently, the network design's operating time for 10 nodes is 0.239 seconds following the use of the CPPGR algorithm. These results encourage us to test more attack graph components and larger nodes and servers (large-scale networks) in order to assess the graph's complexity and execution time utilising the CPPGR algorithm and personal agent.
The attack graph creation time is the first factor. The running times at various phases of attack graph construction are compared in the experiment. According to Table 1, the full graph is used in the first stage without any pruning algorithms, and the CPPGR algorithm is used in the second stage. The outcomes demonstrate how each level differs. By using a parallel process with the algorithm, CPPGR minimises the amount of edges used in the generation and the attack graph's generation time, resulting in shorter generation times.

Table 1: Running time of attack graph generation

Node numbers	Full attack graph	CPPGR algorithm
10 nodes	12.604	0.239
50 nodes	27.489	4.851
100 nodes	62.480	11.605
250 nodes	113.631	35.418
400 nodes	201.170	79.40

The attack graph is created using different techniques, such as depth-first and breadth-first search, on an identical number of nodes with a comparable topology. Because these path-pruning techniques have been employed to prune the attack graph in other research, they were selected for comparison. As seen in Figure 3, the results demonstrate that the CPPGR algorithm outperforms other algorithms.

Next, the attack graph's complexity is calculated. The attack graph shows the connection between the nodes, but it usually depends on the reachability calculation, so in the worst case, the general complexity is $O(V^2)$, where V is the number of nodes. In this paper, the reachability is calculated for each node by a personal agent, so the attack graph's complexity is $O(V)$ for each host, and $O(V * logV)$ for each administrator's calculation of each reachability, so the complexity is $O(V + (V * logV))$. Also, the CPPGR algorithm is used to simplify the attack graph by pruning the unnecessary edge. The complexity of the algorithm is divided to many parts, first part identifies the critical paths which records $O(m \cdot (|E| + |V| log |V|))$ where m critical assets, $|E|$ number of edges, the second part is rank the path which records $O(plogp)$, where p is the total number of paths found, the third part is prune redundant nodes which recode $O(|V|)$, as each node is checked once, the fourth part is merge similar paths which recode $O(k \cdot L)$, where L is the average length of the paths and k is the number of critical paths. This involves comparing and merging paths. The last part is simplifying the edges which recode $O(|E|)$, as each edge is checked once. At the end the attack graph complexity will be

$$O(m \cdot (|E| + |V| log |V|) + plogp + |V| + k \cdot L + |E|)$$

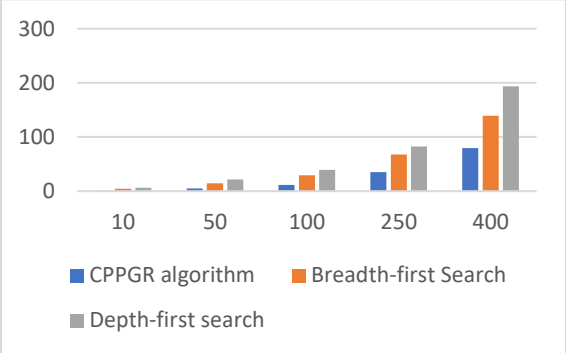

Figure 3: Comparison results between the CPPGR and other algorithms

6. Conclusion

Currently, networks have expanded significantly in both complexity and scale. Nevertheless, the extensive expansion of network connectivity has significantly increased the incidence of cyber-attacks on corporations and government institutions, resulting in operational disruptions and jeopardizing the reputation and financial stability of these entities. The researcher employs a graph model to forecast, detect, and find attack paths to represent the network in order to get around this problem, but the attack graph creation still has a generation time problem. By cutting down on reachability computations and eliminating superfluous edges from the graph, the CPPGR method and personal agent are utilised in this study to improve the generation time. The administrator server is in charge of creating the attack graph after a personal agent collects data from the nodes, assesses the node's reachability, and transmits the findings. This approach balances graph reduction with the preservation of high-risk attack scenarios, ensuring that security analysts can focus on the most significant threats. CPPGR provides a computationally efficient and flexible approach for attack graph simplification by integrating path ranking, pruning, and merging, hence tackling the scalability issues frequently faced in extensive and intricate networks. Future research will integrate Reinforcement Learning (RL) and Graph Neural

ISSN: 3078-5367 **DOI:** https://doi.org/10.69513/jncs.v2.i2.a2

Networks (GNNs) to enhance attack path analysis in attack graphs.

References

[1] Z. J. Al-araji, S. S. A. Syed, M. W. Al-salihi, H. A. Al-lamy, M. Ahmed, and W. Raad, "Network Traffic Classification for Attack Detection Using Big Data Tools : A Review," Intelligent and Interactive Computing, Lecture Notes in Networks and Systems 67, vol. 67, pp. 355–363, 2019, doi: DOI: 10.1007/978-981-13-6031-2_37.

[2] Z. J. Al-Araji, S. S. S. Ahmad, and R. S. Abdullah, "Comparison Study Between Attack Graph Path Based Metrics," in Proceedings of the 3rd International Conference on Intelligent and Interactive Computing 2021, 2021, p. 10.

[3] L. S. Vailshery, "non-IoT connections worldwide 2010-2025," Statista. Accessed: Mar. 18, 2022. [Online]. Available: https://www.statista.com/statistics/1101442/iot-number-of-connected-devices-worldwide/

[4] E. Bertino, L. D. Martino, F. Paci, and A. C. Squicciarini, "Web services threats, vulnerabilities, and countermeasures," in Security for web services and service-oriented architectures, Springer, 2009, pp. 25–44.

[5] J. M. Kizza, Kizza, and Wheeler, Guide to computer network security, vol. 8. Springer, 2013.

[6] M. Abomhara and G. M. Køien, "Cyber security and the internet of things: vulnerabilities, threats, intruders and attacks," Journal of Cyber Security and Mobility, vol. 4, no. 1, pp. 65–88, 2015.

[7] A. O'DRISCOLL, "25+ cyber security vulnerability statistics and facts of 2021," Comparitech. Accessed: Apr. 02, 2022. [Online]. Available: https://www.comparitech.com/blog/information-security/cybersecurity-vulnerability-statistics/

[8] Y. Yang, L. Wu, G. Yin, L. Li, and H. Zhao, "A Survey on Security and Privacy Issues in Internet-of-Things," IEEE Internet Things J, vol. 4, no. 5, pp. 1250–1258, 2017, doi: 10.1109/JIOT.2017.2694844.

[9] N. Rahim, Bibliometric Analysis of Cyber Threat and Cyber Attack Literature: Exploring the Higher Education Context. 2021.

[10] 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. doi: 10.1145/310889.310919.

[11] A. Ramos, M. Lazar, R. Holanda Filho, J. J. P. C. P. C. Rodrigues, R. H. Filho, and J. J. P. C. P. C. Rodrigues, "Model-based quantitative network security metrics: A survey," IEEE Communications Surveys and Tutorials, vol. 19, no. 4, pp. 2704–2734, 2017, doi: 10.1109/COMST.2017.2745505.

[12] Zaid. J. Al-Araji, S. S. S. Ahmad, and R. S. Abdullah, "Propose Vulnerability Metrics to Measure Network Secure using Attack Graph," International Journal of Advanced Computer Science and Applications, vol. 12, no. 5, pp. 51–58, 2021, doi: 10.14569/IJACSA.2021.0120508.

[13] Zaid. J. Al-Araji, S. S. Syed Ahmad, and R. S. Abdullah, "Attack Prediction to Enhance Attack Path Discovery Using Improved Attack Graph," Karbala International Journal of Modern Science, vol. 8, no. 3, pp. 313–329, Aug. 2022, doi: 10.33640/2405-609X.3235.

[14] Z. Alaaraji and A. Mutlag, "Implement Edge pruning to Enhance attack graph generation using Naïve approach algorithm," El-Cezeri Fen ve Mühendislik Dergisi, Jan. 2024, doi: 10.31202/ecjse.1375755.

[15] Z. J. Al-Araji, S. S. S. Ahmed, R. S. Abdullah, A. A. Mutlag, H. A. A. Raheem, and S. R. H. Basri, "Attack graph reachability: concept, analysis, challenges and issues," Network Security, vol. 2021, no. 6, pp. 13–19, 2021, doi: 10.1016/S1353-4858(21)00065-9.

[16] A. Sabur, A. Chowdhary, D. Huang, and A. Alshamrani, "Toward scalable graph-based security analysis for cloud networks," Computer Networks, vol. 206, no. April, pp. 108795–108815, Apr. 2022, doi: 10.1016/j.comnet.2022.108795.

[17] G. George and S. M. Thampi, "Vulnerability-based risk assessment and mitigation strategies for edge devices in the Internet of Things," Pervasive Mob Comput, vol. 59, no. October, p. 101068, 2019, doi: 10.1016/j.pmcj.2019.101068.

[18] A. Presekal, A. Stefanov, V. S. Rajkumar, and P. Palensky, "Attack Graph Model for Cyber-Physical Power Systems Using Hybrid Deep Learning," IEEE Trans Smart Grid, vol. 14, no. 5, pp. 4007–4020, Sep. 2023, doi: 10.1109/TSG.2023.3237011.

[19] M. Salayma, "Threat modelling in Internet of Things (IoT) environments using dynamic attack graphs," Frontiers in the Internet of Things, vol. 3, May 2024, doi: 10.3389/friot.2024.1306465.

[20] J. Li, T. Xie, L. Chen, F. Xie, X. He, and Z. Zheng, "Adversarial Attack on Large Scale Graph," IEEE Trans Knowl Data Eng, vol. 35, no. 1, pp. 82–95, Sep. 2020, [Online]. Available: http://arxiv.org/abs/2009.03488

[21] J. R. Rose et al., "IDERES: Intrusion Detection and Response System Using Machine Learning and Attack Graphs," Journal of Systems Architecture, vol. 131, 2022.

[22] N. L. Nguyen, N. Falkner, and H. Nguyen, "Demo: Synthesizing Realistic Enterprise Active Directory Attack Graphs with ADSynth," in SIGCOMM Posters and Demos 2024 - Proceedings of the 2024 SIGCOMM Poster and Demo Sessions, Part of: SIGCOMM 2024, Association for Computing Machinery, Inc, Aug. 2024, pp. 107–109. doi: 10.1145/3672202.3673732.

[23] A. Kazeminajafabadi, S. F. Ghoreishi, and M. Imani, "Optimal Detection for Bayesian Attack Graphs Under Uncertainty in Monitoring and Reimaging," in American Control Conference (ACC, Institute of Electrical and Electronics Engineers (IEEE), Sep. 2024, pp. 3927–3934. doi: 10.23919/acc60939.2024.10644873.