Correlating Intrusion Detection with Attack Graph on Virtual Computer Networkings

Hanwen Zhang[1][a], Wenyong Wang[1][b], Lisheng Huang[1][c], Junrui Wu[1][d], Fengjun Zhang[2] and Kai Shi[2]

[1]*School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China*

[2]*The 30th Institute of CETC, China Electronics Technology Cyber Security Co., Ltd, Chengdu, China*

Keywords: Cybersecurity, Attack Graph, Intrusion Detection, Virtual Networks.

Abstract: Securing a computer networking system requires the ability to gather and organise information about potential vulnerabilities existing in the system. One way of utilising the information above is to generate an attack graph of all possible attack paths. Current attack graph generation methods reach scalability issue with the growth of network devices and links, and one solution is to correlate attack graph with intrusion detection systems. However, correlation techniques are rarely studied especially on generating attack graphs on virtual computer networks, as correlations are inflexible to be integrated to existing attack graph generators. Previously we proposed mAGG, an attack graph generation framework on virtual networkings; and LSAFID, an intrusion detection system based on doc-word. In this paper, we propose a new method for correlating intrusion detection algorithm for attack graph generation on virtual networkings. Our new proposed method is flexible in network architectures and functionalities, and shortens the scale of generated attack graph.

1 INTRODUCTION

Attack graph generations are an effective way of demonstrating the possible paths that an intruder might take in action in the form of graph. Specifically, the intruder may exploit exposed vulnerabilities on the attack surface (Theisen et al., 2018) of the computer network, then lead to a chain of intrusion attacks. Each attack causes a post-condition of privilege escalation, firewall bypassing, etc. These post-conditions are also considered as the pre-conditions of the next attacks. Step by step, the chains of intrusion attack actions form an attack graph of the target computer network (Phillips & Swiler, 1998). On securing a computer network, attack graph provides a visualised perspective of in which routes, or formally attack paths, that the intruder might take in action, and classified by vertex, there are four types of attack graph models, state-based, vulnerability-based, host-based and attack scenario-based models respectively (Kaynar, 2016).

[a] https://orcid.org/0000-0001-7224-8230
[b] https://orcid.org/0000-0003-4095-547X
[c] https://orcid.org/0000-0003-3652-3279
[d] https://orcid.org/0000-0003-3658-9243

1.1 Scalability Problems of Generating Attack Graphs

Despite that different model consider different attributes of the network entities, attack graph generation is tougher with the growth of network scale and functionalities, and generation algorithms may easily reach scalability problems over large computer network systems (Hong & Kim, 2013). Most commonly used attack graph generators have the polynomial time complexity of from $O(N^2)$ to $O(N^3)$, with N respect to the number of vertices and edges. (Yi et al., 2013). Moreover, as illustrated in Figure 1, physical topologies of the computer networking are various, from the original bus, star or ring architecture to today's tree hierarchy architecture (Liu & Liu, 2014) or even a "flat" virtualised architecture, causing attack graph generator implementations need a targeted consideration on each independent architecture on deployment.

Zhang, H., Wang, W., Huang, L., Wu, J., Zhang, F. and Shi, K.
Correlating Intrusion Detection with Attack Graph on Virtual Computer Networkings.
DOI: 10.5220/0011784100003405
In *Proceedings of the 9th International Conference on Information Systems Security and Privacy (ICISSP 2023)*, pages 651-658
ISBN: 978-989-758-624-8; ISSN: 2184-4356

Scalability problems may also caused by redanduncy. From January to November of 2022 alone, records of vulnerability items reaches a total 22265 (*Browse Cve Vulnerabilities by Date*, n.d.), while most of the items are scored low-severity and irrelevant to the target network. Keeping the system refreshing on all vulnerabilities will cause a huge time waste, and thus, instant warnings of the intruder's possible attack paths are impratical. Besides, a computer node in a target network may contain multiple vulnerabilities, generating a full attack graph would traverse through all possible exploitable vulnerabilities, causing complex loop problems. Hence, a better way of utilising the attack graph is to devide a full graph into partial (Sawilla & Skillicorn, 2012), (Ingols et al., 2006).

Correlating intrusion detection systems (IDS) (Soni et al., 2015) with attack graphs will limit the attack paths for generation, and therefore, a partial but accurate attack graph reflecting the current undergoing intrusion threats is considered more useful. While correlating IDS with attack graph has been studied by multiple researchers before, none of the researches are available for a virtual networking architecture.

1.2 Virtual Networking: An Emerging Trend

Networking virtualisation was developed by in the 2000s. Virtualising a network and resources takes great advantage in multiplexing the physical infrastructures, providing less waste of computing resources. Besides, virtualisation over physical infrastructures hides the hardware differences from advanced software developments. By hiding hardware differences and physical infrastructures, a virtual machine is not aware of which actual server it is on, and is provided with virtual resources on demand, and idle resources are optimised (Qian et al., 2009).

In fact, virtual environments and networkings, e.g. cloud, IaaS (Infrastructure as a Service) are becoming an emerging trend, and global market of IaaS reaches 90,894 million dollars with a 41.4% growth in 2021 (Meghan Rimol, 2022). The emerging trend of "cloud network integration" has brought benefits in costs, but the less-studied emerging security issues leaves a hidden danger.

A container-based virtualisation technology , e.g. Docker, adopts type-II hypervisor, and it is more light-weight. Container virtualisation builds on top of OS-level instead of hardware level, and the guest containers are isolated by name spaces, so they can directly use system calls. Each container also has its own volumes and virtual network interfaces. Due to the lightweight features of the containers, network services are evolved from a server-based style to service-based style. A single host can run thousands of composed services in parallel.

However, a virtual networking is different from a physical networking in nature, the nodes of the network are "flattened" and network interfaces and connections are virtualized. Straight to the point, we summarise a virtual network of highly polymorphic, i.e., with high "structure definability" to all layers of network (Y. Hu et al., 2020). And thus, a virtual network is much more dynamic than a physical network. Thus, managing and generating topologies and attack graphs on virtual networks are different from the "fixed" physical architectures, especially where:

- A group of service nodes may run the same service as a cluster, and the scale of the cluster are expanded or reduced elastically according to current traffic. Hardware resources are virtualised and can be allocated by every service node.
- Each virtualised node does not have its own physical network interface card, and their virtual networkings adopt dynamic technologies like container orchestration (*Orchestration*, 2022). Virtual management systems create edges between nodes dynamically.

1.3 Our Contributions

As generating attack graphs on virtual networks is rarely studied, correlation of IDS and attack graph generation on virtual networks has never been studied before. As security problems of virtual networks worsening every day, it is critical to propose a method of correlating IDS and attack graph.

In this paper, we propose an IDS and attack graph generation correlation method for virtual networks. Compared with existing methods, our proposed method is built on our flexible and extendable attack graph generation framework, taking the advantage of network virtualisation and abstraction. Our proposed method can instant respond to network changes, especially on virtual networkings. Compared to a full attack graph, our proposed method is a partial graph, corresponding to the current intrusion situations and network connections. To summarise, our proposed method is concluded as follows:

- We formalised the problem of decomposing a full attack graph into partial, with respect to correlation IDS and attack path choices.
- Based on the formalisation of the problem, we proposed an algorithm of generating partial attack graphs based on IDS.
- Based on the generation algorithm, we proposed a method of integrating IDS correlations in our attack graph generation framework.

In this paper, chapters are organised as follows: Chapter 2 the backgrounds of networking virtualisation, the related works on attack graph generation, and IDS correlations. Chapter 3 specifies our proposed method. The experiments and results are listed in Chapter 4. Finally, this paper is concluded in Chapter 5.

2 BACKGROUND AND PREVIOUS WORKS

2.1 Previous Works on Attack Graph Generation and mAGG

Early in 2001, Swiler et al. proposed the attack graph generation tool for computer networks, in order to create a system-level design and assessment tool for vulnerabilities (Swiler et al., 2001). It is aimed to replace network-level and application-level defence system driven by checklists and compliance standards. MulVAL framework (Ou et al., 2005) uses an attack model definition system, (OVAL-Community, 2018/2022) that solved problems of automatically integrating vulnerability from the bug-reporting system. Besides MulVAL, TVA (Jajodia et al., 2005) focused on the topology relationships of exploited vulnerabilities, and estimated actual exploit combinations of vulnerabilities among multiple hosts to advance an attack on a network.

In revolution of attack graphs, Bayesian attack graphs (Cozman, 2000) enables the estimation of the risk of compromise to the system's components given their vulnerabilities and interconnections. Later on, more researches of Bayesian probabilities are introduced into attack graph generation problems (Frigault & Wang, 2008). Bayesian attack graph provides dynamicity risk assessments of decisions (Poolsappasit et al., 2012). Calculating Bayesian probabilities for MulVAL attack graphs also employing dependency of vulnerabilities (School of Electrical Engineering and Informatics, Institut Teknologi Bandung et al., 2015). Recent researches are based on virtualised infrastructures, and provide an effective risk assessment (Asvija et al., 2020), but they face the problem of coping with huge amount or relations.

Formerly, based on polymorphism, we proposed mAGG, a managed multi-layer attack graph generator available for virtual networks with flexibility (Zhang, 2022/2022). Compared with existing attack graph generators, mAGG framework can rapidly respond to virtual network changes in real time, providing an instant view of attack graphs. Besides generating attack graphs, mAGG contains a light-weight management system of vulnerabilities, topologies and attack paths, and has the ability to calculate Bayesian probabilities and to deploy honeypots.

Our framework mAGG uses additional abstract layers over a real topology, and thus reaches instant responds with respect to dynamic changes. The framework decomposes a full topology into autonomous subnets, with each subnet independent to each other. mAGG has solved many problems regarding generating attack graphs on virtual networks, but it is still a full attack graph generator. To the best of our knowledge, mAGG is flexible enough to correlate with IDS, and therefore we propose a correlation algorithm based on mAGG.

2.2 Previous Works on IDS Correlations with Attack Graph and LSAFID

Correlating attack graph with IDS (Rajput & Thakkar, 2019) was first researched to handle missed detections through the analysis of network vulnerability dependencies, with a simple scalable attack graph representation (Noel et al., 2004). Later on, researches focus on high-speed networks using a queue to correlate new and in-memory alerts (Wang et al., 2006). Recently researchers have correlate MulVAL with IDS (H. Hu et al., 2020), and have been working on automations with severe alerts (Nadeem et al., 2022).

Previous researches, however, is not practical on emerging trends of network virtualisation, as is stated before. They all lack the flexibility and extendibility that cannot be applied with instant changes.

Previously, we proposed LSAFID, a topic model-based IDS framework, which construct a doc-word matrix from statistical features and then analyses latent semantic information to determine whether an aggregated flow is malicious (Wu et al., 2022). The results indicates that LSAFID achieves higher performance and better ROC curves than other competing methods.

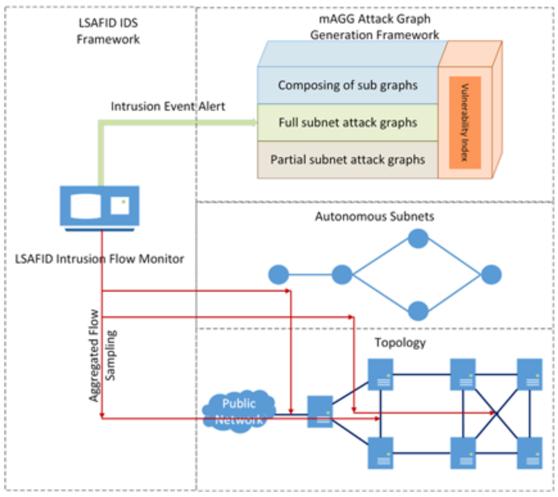

Figure 2: Overall of the proposed method.

3 THE PROPOSED METHOD

As is stated before, while mAGG provides attack graph generations on a virtual network with extendibility, the framework itself cannot aware of the current cyber situations. LSAFID provides accurate and efficient classification of aggregated flow, as the classification results are flow-based instead host-based. Hence, correlation attack graph generations with IDS need a method of deciding on which link should the IDS be deployed and how the flow-based results are integrated into host-based or state-based attack graphs. And therefore, the problem is decomposed as follows:

(1) Traffic Flow Sampling: On a virtual network, links between virtual machine nodes are also virtualised. The proposed method should decide on which point the traffic flow are sampled for IDS.

(2) Attack Graph Decomposing: Based on the flow-based results of IDS classifications, the proposed method should be able to utilise the results and make partial attack graphs accordingly.

(3) Dynamic Update: For persistent intrusion detections, the proposed method should be able to provide instant update of above partial attack graph.

In the following sections, we will demonstrate our proposed method step by step. The overall architecture of our proposed method is illustrated in Figure 2.

3.1 Threat Modelling

Threat model put constraint on the ability of intruders, as well as define the boundary of cybersecurity problems.

- **Attack Targets:** In this work, attack targets are defined as virtual service entities with its own virtual network interfaces, where the intruder could exploit potential vulnerabilities.
- **Attack Goals:** An attack goal is an attack target, of which the intruder is intended on, e.g., a database, or a core server.
- **Intruder:** An intruder is a malicious individual from public network that tries to intrude the attack targets. The intruder only has the access to the attack surfaces at the beginning. The information of open recorded vulnerabilities is symmetric to both intruder and the network manager, but the intruder cannot obtain a global view of topologies as priori. Assume that the intruders are rational individuals.
- **Attack Processes:** In each step, the intruder has a certain pre-condition, and the intruder will try to penetrate attack targets that are adjacent to already intruded attack targets, in the form of exploiting vulnerabilities, according to the pre-conditions. Intruder will then acquire post-condition as the result of success exploiting. The intruder will stop intruding if and only if the attack goals are intruded, or no further nodes can be compromised.

3.2 Bottom-to-Top Construction Layering System

In previous sections, we discussed attack graphs in coverage of partial and full, and in form of state-based or host-based, etc., respectively. Also, current IDS are flow-based, and all of above indicates that if each algorithm works individually, the composed framework will be too complex to make conversions between different interfaces. The problem states that the proposed method should be built from bottom to top, with different abstract layers overlapping each other. In this way, each framework works in its own abstract layer space, and a latter-processed framework won't affect a former-processed framework, since it is on "top" of the former one.

Specifically, we consider a topology layer that matches with actual network topology, an autonomous subnet layer that is correspond to subnets on the topology, an mAGG attack graph generation layer, and LSAFID feedback layer that is respond to monitor intrusion events, respectively.

In a private network, a subnet is often referred to a LAN (local-area network), where all network devices share the same local IP address prefix. The reason why we treat subnetting as an independent layer is that by its definition (J. Postel, 1988), all nodes are adjacent to each other in a LAN, in perspective of network layer, so we can say that a LAN in a full-connected complete graph with respect to network layer. As nodes of different LANs are not adjacent, and the connection links must come across the **gateway** nodes between LANs. Therefore, a subnet is a small enough autonomous group of nodes that changes within the subnet won't affect the other subnets. For a formal definition, we have:

A subnet $sub = \{n_1, n_2, \ldots, n_n\}$ is a set of nodes that are adjacent to each other, and a topology $T = \{sub_1, sub_2, \ldots, sub_n\}$ is a set of subnets, which satisfies:

$$\forall\, m \in sub, \nexists n \in T \text{ and } n \notin sub, \\ \text{s.t. } m, n \text{ adjacent} \quad (1)$$

Also, subnets may overlap each other, i.e., a node may belong to multiple subnets at the same time. We call this kind of special node as **gateways**. Gateway nodes works as cut set of the overall topology, that the autonomous subnets are divided via the cut set of gateways. Based on subnet definitions, we de-composed an overall topology into autonomous subnets, such that the attack graphs can be generated for each subnet, and the influence of topology changes will be limited within one subnet.

Based on a layer of subnets, full attack graph for each subnet can be built by mAGG framework as an overlay layer.

3.3 Partial Attack Graph Generation of Subnet

The original results of mAGG framework leads to a full attack graph considering every possible vulnerability presenting in the system. However, as network scale grows large, the even attack graph within a single subnet will be too complex. And so, even low-scored vulnerabilities are taking part in attack graph generations. In short, a full attack graph is too redundant to practical usage, and is too complex in generation process.

Instead, a partial attack graph only considers a countable number of vulnerabilities that are truly threatening the system. Besides, the vulnerability databases and indices can be considerably smaller than a full record. In mechanism, each node on a network is needed to be scanned, and then the attack graph generator will try to lookup the database for vulnerability records. In performance, a partial vulnerability database will outcome a full database.

Based on the assumption of partial attack graph as above, we propose a method of generating partial attack graph for a full-connected subnet as Algorithm 1.

Algorithm 1: Generate Attack Graph for Subnet.

```
Input:
$sub$: a subnet
$g[]$: gateways of $sub$
$V := \{n: v\}$: partial vulnerability index
$stack$: a last-in first-out stack of nodes
Output:
$sAG$: attack graph for a subnet
function generate_sub_graph:
  for $gateway$ in $g[]$:
    for $cond_{post}$ in $V.\mathrm{get}(gateway)$:
      $sAG.\mathrm{add\_vertex}(gateway, cond_{post})$;
      $stack.\mathrm{push}(gateway, cond_{post})$;
    end for
  end for
  depth_first_search();
  return $sAG$;
end function
function depth_first_search:
  while $stack \neq \emptyset$:
    $n, n.cond_{post} \leftarrow stack.\mathrm{pop}()$;
    for $neighbour$ in $sub$:
      for each $value$ in $n.cond_{post}$:
        $vn \leftarrow V.\mathrm{get}(value)$:
          if $ea := (n, neighbour, vn.cond_{post}) \notin sAG$:
            $sAG.\mathrm{add\_vertex}(neighbour, vn.cond_{post})$;
            $sAG.\mathrm{add\_edge}(ea)$;
            $stack.\mathrm{push}(neighbour, vn.cond_{post})$;
          end if
      end for
    end for
  end while
end function
```

Algorithm 2: IDS Sampling and Feedback.

```
Input:
$g[]$: gateways nodes
$T_s$: sampling period
$V := \{n: v\}$: partial vulnerability index
function correlate:
  while true do:
    sleep($T_s$);
    $V \leftarrow \emptyset$;
    for $gateway$ in $g[]$:
      $flow \leftarrow$ sample_flow($gateway$);
      $is\_malicious \leftarrow$ call_LSAFID(flow);
      if $is\_malicious$ = true do:
        $V.\mathrm{put}(gateway)$;
        send_alert($gateway.subnet$);
      end if
    end for
  end while
end function
```

3.4 Flow Sampling and Intrusion Event Alert Feedback

In the previous section, we proposed a method of generating partial attack graph for a certain subnet. Notice that for input, a partial vulnerability index is required. The way of generating partial vulnerability index will definitely result in lowering the recall rate, but with proper method, this result will rise the precision rate, as redundant information of vulnerabilities is reduced.

The key of properly reducing vulnerability indices is apply traffic flow monitoring on the target network. By detecting intrusion events based on sampled flow of traffic, which set of vulnerabilities can be loaded for attack graph generation properly.

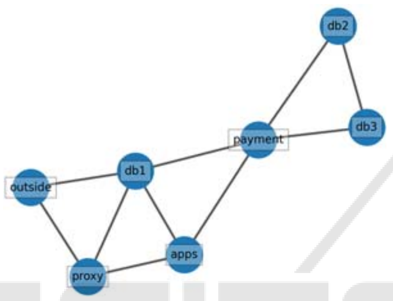

Figure 3: The experimental network.

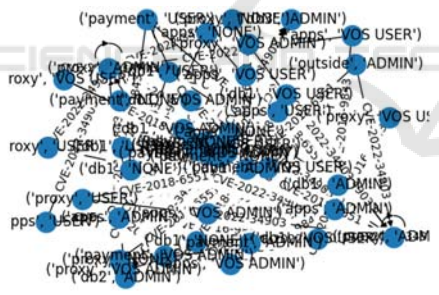

Figure 4: Full Attack graph.

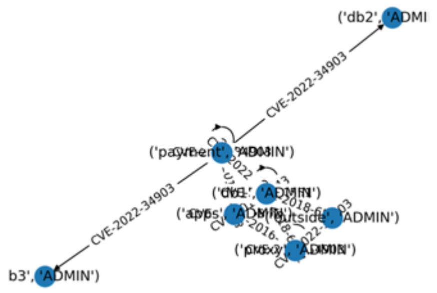

Figure 5: Partial attack graph 1.

The LSAFID process is applicable on aggregation traffic flows, and we deploy monitors on gateway nodes of subnets, that traffic is aggregated at most, such that the system burden is the least. Algorithm 2 explains sampling process and IDS feedback.

Once in every sampling period, the framework would call LSAFID once for aggregated flow intrusion detection. Based on LSAFID classifier's output, if the flow of a node is malicious, its corresponding vulnerability records will be loaded locally, and all the affected subnets will update the attack graph as partial. Since subnets are autonomous systems, this process can be done in parallel.

4 EXPERIMENTS AND RESULTS

In this paper, we conduct our experiments with a designed virtual network built on Docker Compose environment, and the program runs with Python 3.10 on a laptop with Intel i7-9750H CPU and 16GB of RAM. The experimental network is illustrated in Figure 3. Each node except "outside" is an attack target, instead, it is marked as the point connecting to public networks, and node "payment-db" is the attack goal.

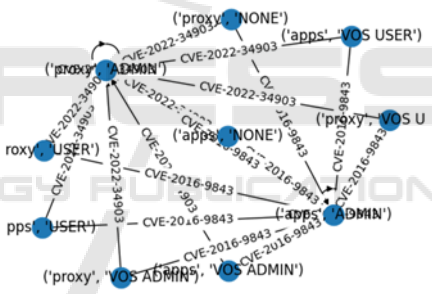

Figure 6: Partial attack graph 2.

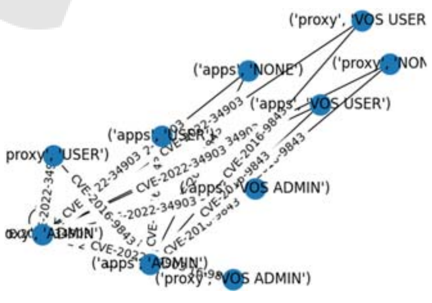

Figure 7: Partial attack graph 3.

Figure 4 illustrates the attack graph without correlations with IDS, and Figure 5, Figure 6 and Figure 7 shows the partial attack graph according to correlated with different intrusion events, respectively.

5 CONCLUSIONS

Attack graph generation systems have been researched over years, and they give good results of illustrating the computer network security problem as a whole, while current methods also face scalability problems. In this work, we proposed a method for correlating IDS with attack graph generation systems on virtual networkings, especially where the network is built on a cloud. By integrating LSAFID system with mAGG system, we achieve high time efficiency for intrusion detection correlation with vulnerability indexing, and high dynamicity for partial attack graph generations. The proposed method could be applied to a virtual network with a large scale of deployments.

We will keep working on cyber security problems, including the fields of IDS, attack graph generations, etc. And this work will be a part of our final model of cyber security intelligence framework.

REFERENCES

Asvija, B., Eswari, R., & Bijoy, M. B. (2020). Bayesian attack graphs for platform virtualized infrastructures in clouds. *Journal of Information Security and Applications*, *51*, 102455. https://doi.org/10.1016/j.jisa.2020.102455

Browse cve vulnerabilities by date. (n.d.). Retrieved November 22, 2022, from https://www.cvedetails.com/browse-by-date.php

Cozman, F. G. (2000). *Generalizing variable elimination in bayesian networks*. 27–32.

Frigault, M., & Wang, L. (2008). Measuring network security using bayesian network-based attack graphs. *2008 32nd Annual IEEE International Computer Software and Applications Conference*, 698–703. https://doi.org/10.1109/COMPSAC.2008.88

Hanwen Zhang. (2022). *mAGG: A managed multi-layer attack graph generator* [Python 3]. https://github.com/Cforcharming/mAGG (Original work published 2022)

Hong, J. B., & Kim, D. S. (2013). Performance Analysis of Scalable Attack Representation Models. In L. J. Janczewski, H. B. Wolfe, & S. Shenoi (Eds.), *Security and Privacy Protection in Information Processing Systems* (pp. 330–343). Springer. https://doi.org/10.1007/978-3-642-39218-4_25

Hu, H., Liu, J., Zhang, Y., Liu, Y., Xu, X., & Tan, J. (2020). Attack scenario reconstruction approach using attack graph and alert data mining. *Journal of Information Security and Applications*, *54*, 102522. https://doi.org/10.1016/j.jisa.2020.102522

Hu, Y., Li, D., Sun, P., Yi, P., & Wu, J. (2020). Polymorphic smart network: An open, flexible and universal architecture for future heterogeneous networks. *IEEE Transactions on Network Science and Engineering*, *7*(4), 2515–2525. https://doi.org/10.1109/TNSE.2020.3006249

Ingols, K., Lippmann, R., & Piwowarski, K. (2006). Practical Attack Graph Generation for Network Defense. *2006 22nd Annual Computer Security Applications Conference (ACSAC'06)*, 121–130. https://doi.org/10.1109/ACSAC.2006.39

J. Postel. (1988). *Standard for the transmission of IP datagrams over IEEE 802 networks* (Request for Comments RFC 1042). Internet Engineering Task Force. https://doi.org/10.17487/RFC1042

Jajodia, S., Noel, S., & O'Berry, B. (2005). Topological Analysis of Network Attack Vulnerability. In V. Kumar, J. Srivastava, & A. Lazarevic (Eds.), *Managing Cyber Threats: Issues, Approaches, and Challenges* (pp. 247–266). Springer US. https://doi.org/10.1007/0-387-24230-9_9

Kaynar, K. (2016). A taxonomy for attack graph generation and usage in network security. *Journal of Information Security and Applications*, *29*, 27–56. https://doi.org/10.1016/j.jisa.2016.02.001

Liu, Q., & Liu, Q. (2014). A study on topology in computer network. *2014 7th International Conference on Intelligent Computation Technology and Automation*, 45–48. https://doi.org/10.1109/ICICTA.2014.18

Meghan Rimol. (2022, June 2). *Gartner says worldwide IaaS public cloud services market grew 41.4% in 2021*. Gartner. https://www.gartner.com/en/newsroom/press-releases/2022-06-02-gartner-says-worldwide-iaas-public-cloud-services-market-grew-41-percent-in-2021

Nadeem, A., Verwer, S., Moskal, S., & Yang, S. J. (2022). Alert-driven attack graph generation using S-PDFA. *IEEE Transactions on Dependable and Secure Computing*, *19*(2), 731–746. https://doi.org/10.1109/TDSC.2021.3117348

Noel, S., Robertson, E., & Jajodia, S. (2004). Correlating intrusion events and building attack scenarios through attack graph distances. *20th Annual Computer Security Applications Conference*, 350–359. https://doi.org/10.1109/CSAC.2004.11

Orchestration. (2022, November 17). Docker Documentation. https://docs.docker.com/get-started/orchestration/

Ou, X., Govindavajhala, S., & Appel, A. W. (2005). MulVAL: A logic-based network security analyzer. *14th USENIX Security Symposium (USENIX Security 05)*. 14th USENIX Security Symposium (USENIX Security 05). https://www.usenix.org/conference/14th-usenix-security-symposium/mulval-logic-based-network-security-analyzer

OVAL-Community. (2022). *The OVAL community repository* [XSLT]. OVAL-Community. https://github.com/OVAL-Community/OVAL (Original work published 2018)

Phillips, C., & Swiler, L. P. (1998). A graph-based system for network-vulnerability analysis. *Proceedings of the 1998 Workshop on New Security Paradigms - NSPW '98*, 71–79. https://doi.org/10.1145/310889.310919

Poolsappasit, N., Dewri, R., & Ray, I. (2012). Dynamic security risk management using bayesian attack graphs.

IEEE Transactions on Dependable and Secure Computing, *9*(1), 61–74. https://doi.org/10.1109/TDSC.2011.34

Qian, L., Luo, Z., Du, Y., & Guo, L. (2009). Cloud computing: An overview. In M. G. Jaatun, G. Zhao, & C. Rong (Eds.), *Cloud Computing* (pp. 626–631). Springer. https://doi.org/10.1007/978-3-642-10665-1_63

Rajput, D., & Thakkar, A. (2019). A survey on different network intrusion detection systems and CounterMeasure. In N. R. Shetty, L. M. Patnaik, H. C. Nagaraj, P. N. Hamsavath, & N. Nalini (Eds.), *Emerging Research in Computing, Information, Communication and Applications* (pp. 497–506). Springer. https://doi.org/10.1007/978-981-13-6001-5_41

Sawilla, R., & Skillicorn, D. (2012). Partial cuts in attack graphs for cost effective network defence. *2012 IEEE Conference on Technologies for Homeland Security (HST)*, 291–297. https://doi.org/10.1109/THS.2012.6459864

School of Electrical Engineering and Informatics, Institut Teknologi Bandung, Sembiring, J., Ramadhan, M., School of Electrical Engineering and Informatics, Institut Teknologi Bandung, S Gondokaryono, Y., School of Electrical Engineering and Informatics, Institut Teknologi Bandung, A Arman, A., & School of Electrical Engineering and Informatics, Institut Teknologi Bandung. (2015). Network security risk analysis using improved MulVAL bayesian attack graphs. *International Journal on Electrical Engineering and Informatics*, *7*(4), 735–753. https://doi.org/10.15676/ijeei.2015.7.4.15

Soni, M., Ahirwa, M., & Agrawal, S. (2015). A survey on intrusion detection techniques in MANET. *2015 International Conference on Computational Intelligence and Communication Networks (CICN)*, 1027–1032. https://doi.org/10.1109/CICN.2015.204

Swiler, L. P., Phillips, C., Ellis, D., & Chakerian, S. (2001). Computer-attack graph generation tool. *Proceedings DARPA Information Survivability Conference and Exposition II. DISCEX'01*, *2*, 307–321 vol.2. https://doi.org/10.1109/DISCEX.2001.932182

Theisen, C., Munaiah, N., Al-Zyoud, M., Carver, J. C., Meneely, A., & Williams, L. (2018). Attack surface definitions: A systematic literature review. *Information and Software Technology*, *104*, 94–103. https://doi.org/10.1016/j.infsof.2018.07.008

Wang, L., Liu, A., & Jajodia, S. (2006). Using attack graphs for correlating, hypothesizing, and predicting intrusion alerts. *Computer Communications*, *29*(15), 2917–2933. https://doi.org/10.1016/j.comcom.2006.04.001

What is a container? - Docker. (2021, November 11). https://www.docker.com/resources/what-container/

Wu, J., Wang, W., Huang, L., & Zhang, F. (2022). Intrusion detection technique based on flow aggregation and latent semantic analysis. *Applied Soft Computing*, *127*, 109375. https://doi.org/10.1016/j.asoc.2022.109375

Yi, S., Peng, Y., Qi Xiong, Wang, T., Dai, Z., Gao, H., Xu, J., Wang, J., & Xu, L. (2013). Overview on attack graph generation and visualization technology. *2013 International Conference on Anti-Counterfeiting, Security and Identification (ASID)*, 1–6. https://doi.org/10.1109/ICASID.2013.6825274.