Multi-target Risk Score Aggregation for Security Evaluation of Network Environments

Ming Lei*, Taous A. Madi†, Matthew Nitschke*, Lianying Zhao*, Makan Pourzandi†,
*Carleton University †Ericsson Research Canada

***Abstract*—Scoring computer systems/networks in terms of specific threats or concerns can enable the comparison of their security level, in a quantitative manner, to facilitate decision making, e.g., mitigation prioritization. The state-of-the-art approaches have mostly focused on scoring the security of a given target, while aggregating scores of multiple systems where each system can be a potential target remains less explored, e.g., whether network A is relatively more secure than network B. In this paper, we take advantage of the well-established attack path representation and use such paths as inter-system influences to derive a risk score of the entire network. We consider the security semantics of various forms of score aggregation, which has not been studied by prior work, and propose to use what we call pairwise path aggregation. We evaluate our approach with a typical fifth Generation (5G) core network, supplemented by evaluations for other network types. The results show that our approach is able to reflect how the overall security varies with multiple factors in common operational scenarios of IT environments.**

***Index Terms*—Security Metrics, Attack Path, Attack Graph, Bayesian Network, Score Aggregation.**

I. Introduction

To enable informed and sound decision making to maintain and improve security in modern complex IT environments, there is a need to quantitatively evaluate the security posture of computer/network systems. Traditionally, there has been the paradigm of security metrics [1], [2], that defines a system of measures for such quantification. Despite the diverse and wide variety of aspects that can and need to be measured [3] (eventually determined by the stakeholders' need and priority), and the subtlety of the definition of security [4], having a single score representing the relative security level is still useful, as it *enables comparison*. Examples of security factors that can be measured include but are not limited to software vulnerabilities, compliance with certain benchmarks/guidelines [5], password strength/policies [6], and exposed network interfaces [7]. Aside from the multiple factors, a score representing the security level usually needs to involve the aggregation from *multiple systems* to enable meaningful comparisons, because of the increased number of connected systems and the evolving hosting technologies. In particular, virtualization and containerization as used in the latest telecommunication technologies (e.g., 5G [8] with network function virtualization) continue to contribute to such complexity. For example, knowing the score of a virtual machine (VM) instance may not help much with learning the overall security posture until the score of the entire environment (e.g., with multiple VMs) and the underlying network connectivity) is available, which cannot be simply a sum of all the VMs' scores.

An intuitive principle is to ensure that the influences between individual systems are reflected in the aggregation (e.g., a system's security level might be different if placed behind a firewall, compared to exposed to the Internet). Nonetheless, such influences would only apply when the compromise of a security factor on one system can enable or facilitate the potential compromise of another system through network connections. In this case, the influences are often in the form of causal relationship between systems. Attack Graphs (AGs) [9] are a well-established and popular way of representing the steps an attacker needs to take (the causal relationship), in the form of Attack Paths (APs) of vulnerability exploits. To allow quantitative aggregation, the potential compromise of the security factors (e.g., through vulnerability exploit) is often converted into a *probability*, then, the likelihood of a system being attacked/compromised is calculated using Bayesian Networks (BNs) [10]. This is in line with risk management in the cyber insurance industry [11] (the risk prediction phase via measurable indicators). However, the state-of-the-art is still one step away from the aforementioned aggregation from multiple systems, because existing works [12]–[16] mostly consider one specific target and the results of the aggregation only represent the security level of the given target system, not that of the entire network.

To derive a network-wide security score, we need to consider, on one side, the causal relationship between security factors within different systems, and on the other side, the fact that every single networked system is a potential target, contributing hence to the overall network security score. To address these aspects, we propose to use APs and BNs in order to reflect the causal relationship between networked systems while considering each system as a potential attack target. More specifically, in our approach, we consider each system as a potential source of multi-stage attacks to generate APs. Afterwards, we devise what we call pairwise *(source, target)* AP aggregation to account for all possible APs from any specific source towards a specific target (becoming pairwise AGs). We further use BNs to evaluate the pairwise AGs for a risk score for each potential target system. Finally, we aggregate the systems' risk scores to obtain a network level multi-target score. Using a 5G core network example, we show that our multi-target risk score is useful in examining the impact of different factors (e.g., vulnerability patching and topology change) on the networks' security level evolution.

Contributions. To the best of our knowledge, this is the first effort to derive, in a principled way, a single risk score for measuring the network-wide security level. First, we examine multiple aggregation choices and discuss their issues to show none is universally optimal. Then, we apply different aggregation choices at different levels (including our proposed pairwise AP aggregation at the target node level). We use AG Bayesian network calculations to capture the impact of the inter-system vulnerability causal relationship and derive a per-target risk score, which is further aggregated into a single network score. We use a 5G core network to demonstrate the usefulness and effectiveness of our multi-target risk score in reporting the operational networks' security-level evolution.

II. Preliminaries

A. Background

To facilitate discussion, we briefly explain several concepts that are used in our proposed approach.

CVSS. The Common Vulnerability Scoring System (CVSS) [17] is a widely accepted metric for scoring and rating the severity of computer system security vulnerabilities (reported by the community into maintained repositories [18], [19]). It provides a numerical score that reflects the likelihood and potential impact of a vulnerability if exploited, allowing organizations to prioritize patching certain vulnerabilities.

Attack Graph. An attack graph (AG) [9] is a directed acyclic graph (DAG) which shows a visual representation of the potential APs that an attacker might take to compromise a system (the target). It shows the relationship between multiple vulnerability exploits. To generate an attack graph, one needs to collect information about vulnerabilities in the systems/network and then identify the pre-conditions (requirements) and post-conditions (consequences) of exploiting a vulnerability. These conditions are used to generate APs by connecting all the exploits.

Bayesian Network. A Bayesian Network (BN) is a probabilistic model [10], for analyzing complex systems and making probabilistic predictions between different events. Although an attack graph can reflect all the possible paths to attack a given target, it does not provide any numeric value by itself to reflect the likelihood of the target being attacked. Wang et al. [20] proposed a method to calculate the overall likelihood that an attacker can successfully reach and exploit the target node in the given attack graph using Bayesian networks. For the target, the cumulative probability of being compromised can be derived along the APs via conditional probability calculation.

B. Terminology

We clarify the meaning of several terms used in our context as follows.

Node. As we are considering a network of computing systems, be it each a physical computer, a VM, or even a container in some cases, we refer to each computing system as a node. Multiple nodes are interconnected via network links, e.g., wireless, wired or virtual connections. We call a node that is considered for potential compromise by an attack, a **Target**.

Network. We refer to the ensemble of the interconnected nodes for which we derive a score as the "network". Our approach is not specific to the paradigm of network security, but the security of networked nodes.

Source-target node pair. The source node and the target node in an AG. The source-target node pair paths refer to all APs that have the same source and target nodes.

Risk score. To ensure consistency throughout the aggregation process, we use the probability of compromise as the risk score (i.e., greater scores indicating higher risks), which is also in line with the meaning of the CVSS [17] scores.

C. Threat model

We assume that an attacker can start from any vulnerable components in a network enabled by various attack vectors, e.g., a careless employee clicking on a link/attachment in a phishing email, Internet-facing components with exploitable vulnerabilities compromised from an attacker-controlled device on the Internet. Thereafter, this component becomes the landing point (source node) of subsequent attack paths.

Once the attack has started, we base our security scoring only on technical factors enabling the adversary to gain unauthorized access to resources including taking full control of a system, from an already-compromised node. Other non-technical but security-related factors (e.g., past security incidents, employee training level and institutional factors) are out-of-scope for attack path derivation.

In line with the Dolev-Yao threat model [21], nodes are isolated from each other, unless there exist vulnerabilities breaking/escaping the isolation, and they can only reach each other through network connections, if applicable and existent. We also generalize risk factors to *vulnerabilities* that allow the attack to land on the next node through network connections.

III. Methodology

In this section, we present an overview of our proposed approach with a workflow, then, we discuss the steps along with the aggregation choices in more detail.

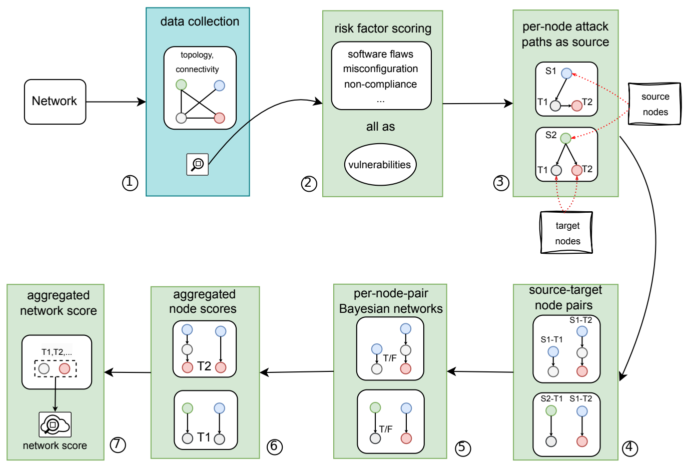

Fig. 1. Overview and work flow of the proposed aggregation

Workflow. The first step is data collection (step ① in Figure 1), which involves the state-of-the-art/practice ways of quantifying security factors (e.g., CVSS). Also, in this step, connectivity information is prepared in the form of network topology (vertices and edges in a file) and access control information (e.g., firewall rules) that can be collected by a script. Next, risk factor scoring (step ②) is conducted to make probability scores of risk factors on each node available for subsequent aggregation. With both risk factor scores and connectivity information, APs can be identified (step ③). For instance, if A can reach B, there is a network path, but if B does not have an exploitable vulnerability from A, there is no AP. The outcome of this step is all APs with each given node as the *source*. Now, we consider each node as a potential target, being attacked based on the identified APs (step ④). At the end of this step, each node as a target will be assigned all the APs originating from other nodes. These APs are used to build per-pair <*source, target*> AGs. In step ⑤, BNs are used to calculate a pairwise risk score, i.e., probability of compromise, for each pair <*source, target*> AG. Per-node aggregation is then performed in step ⑥ to derive a single node score by aggregating all pairwise scores with the same target. Last, the network level aggregation will further aggregate all the node scores to a single score (step ⑦), representing the security level of the entire network. In the following, we detail the steps and our aggregation choices.

A. Risk Factor Scoring

For simplicity, we generalize exploitable security problems to be vulnerabilities, and thus consider reported and publicly accessible common vulnerabilities (e.g., CVEs [18]) in network environments that can be identified by various scanners such as Nessus [22], OpenVAS [23], etc. Note that the existence of such CVEs does not mean they can be or are already exploited. The CVSS metrics reflect the severity/impact of the vulnerabilities if exploited.

More importantly, the CVSS metrics need to be converted to probabilities. We employ the approach proposed by Zhang et al. [24] to assign a probability of exploitation to a specific vulnerability (we will discuss aggregating attack probabilities of multiple vulnerabilities in Section III-D).

B. Source-Oriented Attack Path Generation

We set out to generate all possible APs originating from every single node (i.e., source) of the network. APs allow to establish causal relationships between vulnerabilities exploited in multi-step attacks. Therefore, in the quest of building APs, we filter out only vulnerabilities whose post-conditions, leverage connectivity for lateral movement between nodes through code execution (e.g., CVE-2020-17530 [25]). Hereafter, we call the vulnerabilities that satisfy this requirement, *eligible vulnerabilities*.

Pre-condition and post-condition association. There exist various methods to derive and connect the pre-conditions and post-conditions, mostly based on the CVE description text, e.g., using natural language processing (NLP) [26], [27].

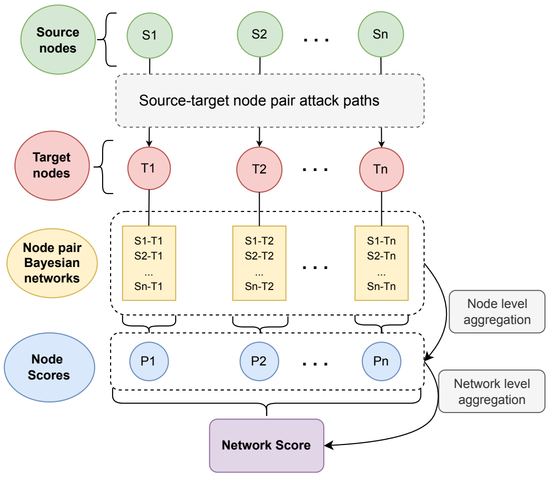

Fig. 2. Aggregating source-target attack paths (single source and single target, each)

In this paper, we employ the approach proposed by Aksu et al. [28] to demonstrate how the pre-conditions and post-conditions of a vulnerability exploitation can be identified as one step of our overall approach. It is rule-based, and the rules check both the description text and impact metrics of a vulnerability. If the privilege of exploiting a vulnerability is mentioned in the description, then post-conditions can be identified as obtaining user or admin privilege. Otherwise, we further check the impact metrics to determine the post-conditions. While in practice being a user could pertain to various accounts or different software entities, e.g., a web server, but not necessarily the operating system, we simplify the post-conditions of eligible vulnerability exploits to be the ability to run code under the privilege of either *User* or *Admin*.

Attack Path generation. Once pre-conditions and post-conditions are derived, we aim at automatically generating the APs. To this end, we adapt the AG generation tool proposed in [29]. This tool assumes that the attacker is located outside the network to start the attack. Given a vulnerable component in the network, it searches around its neighbours and generates APs by identifying required pre-conditions to exploit any adjacent components. Considering our assumption of the attacker possibly located at any node, and every single node in the network as a potential starting point of the attack, i.e., source, we then generate all APs originating from each node as the source.

An example of generated APs is shown in Figure 3 (left) considering the privilege post-conditions *user* and *admin*. Nodes A, B, C and D represent vulnerable components, and edges indicate both network connectivity and vulnerability exploitability. The privileges indicate the pre-conditions and post-conditions. For example, an attacker can start with the user privilege on node A to exploit vulnerabilities on node B or node C to obtain admin privileges. From either B or C, the attacker can continue to exploit the vulnerabilities on D and obtain the admin privilege there.

C. Pairwise Attack Path Aggregation

So far, we have generated all source-oriented APs with each node in the network considered as a source. Our purpose now is to enumerate and aggregate all the APs that start from different source nodes and terminate at the same node as the target. Several options to aggregate APs towards a target can be accounted for: **1) Merging**. By simply merging all nodes and edges, we will ignore all the intermediate attack sources (whose probability used to be "1" before merging but has been overridden by another longer path). This reduces the source nodes to the border nodes only, causing hence shorter APs to be overlooked in risk score calculation. **2) Combining**. In this case, all probability-one source nodes are preserved, but due to the way Bayesian's conditional probability is calculated, the probability-one nodes will "short-circuit" longer APs and downgrade the aggregation to almost only consider individual nodes as opposed to paths. **3) Pairwise aggregation**. As a trade-off, we consider what we call <*source, target*> pairs so that each AG only involves a single attack source (one or multiple APs) towards a given target. Then, we conduct a per-pair aggregation without ignoring sources (i.e., the probability-one nodes) and later aggregate all pairs. Our pairwise aggregation approach allows to both accommodate the fact that the attacker can be on any node (as opposed to border nodes only achieved with merging) and avoid neighbor-node (closest) attackers dominating the probability/score (achieved by the combining approach). We illustrate this approach in Figure 2.

To form APs, each node can be either a source or a target. Therefore, we need to enumerate all attack possibilities. Given a source node $Si \in \{S1, S2, ..., Sn\}$ and a target node $Tj \in \{T1, T2, ..., Tn\}$, we have the <*source, target*> node pair <Si, Tj>. We then find the corresponding APs which start from Si and terminate at Tj. Hence, with each individual node as a target, we can find a list of <*source, target*> node pairs that have the same target but different source nodes (see boxes in Figure 2), where, each <*source, target*> pair defines an AG. Next, we calculate the likelihood of each target node being compromised from different source nodes.

D. Per-target Node Aggregation

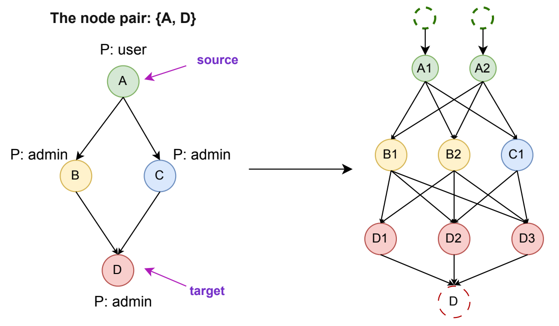

Fig. 3. Constructing the AG. Multiple vulnerabilities are considered as separate exploit nodes, e.g., A1 and A2.

At this point, each node in the network has all possible AGs towards it, each with a single source.

Multiple exploits within a non-target node. A source/intermediate node might have multiple exploitable vulnerabilities. To handle this case, we represent each exploit as a node. On the right-hand part of Figure 3, all vulnerabilities on node A (i.e., $A1$ and $A2$) are connected to all vulnerabilities on node B (i.e., $B1$ and $B2$) and on node C (i.e., $C1$), and the same applies to node D. This way, their individual effects contributing to the target compromise are counted with the BN calculation.

Multiple exploits within a target node. In our example (Figure 3), when any of the vulnerability exploits $D1$, $D2$ or $D3$ is successful, the target node, D, can be considered as compromised (e.g., an attacker has obtained the ability of code execution). To reflect this semantics, we choose to use a *dummy node* (whose exploit probability is always 1, shown as the red dotted circles in the Figure 3) to represent all exploits ($D1$, $D2$ and $D3$) on the same node, and perform the regular Bayesian calculation and the cumulative conditional probability to obtain a single score on the target node D. The calculated score (probability) of the dummy node represents the likelihood of node D being compromised from the source node A.

Node score aggregation from all sources. Once the remaining node pair scores for the target D are calculated, we use an arithmetic mean value of all node pair scores to obtain the aggregated target node score for node D.

E. Multi-target Aggregation

At this point, each node score ($P1, P2, ...$) already includes influences (causal relationship) from all other nodes (see Figure 2). Next, we need to convert the node scores into a single score as a representation of the security level of the entire network. To this end, again, we use an arithmetic mean of individual node scores. Alternatively, another option would be to treat these node scores as exploits and use a dummy target node to represent the entire network, as in Section III-D. This dummy node means the cumulative conditional probability of any one of the nodes being compromised.

However, the calculated score for the dummy node does not correctly reflect the security level of the network. Given a dummy node with n parents $(v_1, v_2, ...v_n)$, the score of the dummy node v_t is determined by its parent nodes as a conditional probability, i.e. $P(v_t = T) = \sum P(v_t = T|v_1, v_2, ...v_n)$, where T means exploit success. Its implication is that compromising any one node leads to the entire network being considered compromised, which does not reflect the fact.

We illustrate this through a simple example: assume that networks A and B both have three nodes, and with node scores of A: $\{0.1, 0.1, 0.99\}$ and B: $\{0.99, 0.99, 0.99\}$. If we build a BN with a dummy node, the aggregated scores of the entire network (i.e., $P(v_t = T|v_1, v_2, v_3)$) for A and B are respectively, 0.991 and 0.999. However, there is only one highly risky node in network A as opposed to network B, which has three severely vulnerable nodes. Using the arithmetic mean as an example, the risk score of network A is 0.397, while the score of network B remains 0.99, which aligns with the intuition regarding the security level of the two networks.

It should be noted that other than the non-optimal dummy node approach, there exist various options to aggregate scores of multiple network nodes, e.g., topologically or statistically. We go with the arithmetic mean both for simplicity and considering that it can well satisfy our evaluation need.

F. Summary of Aggregation Types

To better explain why we choose a specific type of aggregation at each level/step instead of another, and reflect on the corresponding security implications, in the following, we summarize and briefly discuss them: 1) *Multiple exploits within a non-target node*. The straightforward way of averaging the CVSS scores of all the exploits contradicts with the fact that the more vulnerabilities existing in a node, the larger is the likelihood of it to be attacked. Therefore, taking an average of multiple CVSS scores does not reflect the actual likelihood, e.g., the average of 1000×0.3 will still be 0.3 (i.e., many vulnerabilities did not make the attack easier). We choose to represent each exploit as a node, as shown on the right-hand part of Figure 3. This way, their individual effects contributing to the target compromise are counted equally. 2) *Multiple exploits within a target node*. When it comes to aggregation at the target node, we cannot simply leave $D1$, $D2$ and $D3$ which are not one single score. When any of the exploits is successful, this target node can be considered as compromised (e.g., an attacker has obtained the ability to run code). This effectively means an OR relationship, which is why the dummy node is used with the regular Bayesian calculation. 3) *Multiple nodes in a network*. The case for multiple targets in the network is further different as compromising one node (computer/server) should not be deemed as a compromise of the entire network (see Section III-E). 4) *Multiple APs towards a target node*. The purpose is that we want to quantify the influence of all the APs towards a target for later aggregation at the network level. But the attacker can be on any node (as opposed to border nodes only, a limitation of "merging") and we should avoid neighbor-node (closest) attackers dominating the probability/score (a limitation of "combining") (see Section III-C for details of our pairwise AP aggregation).

IV. Evaluation

We evaluate the effectiveness of the multi-target risk score on a test instance of a 5G core network. We show through a set of experiments how the metric we propose aligns with the expected trends while varying different parameters.

A. Evaluation Setup

We use a 5G core network [30] to evaluate our approach, as commonly prototyped in projects like Open5GS [31], Free5GC [32] or OpenAirInterface [33]. In mobile telecom networks, a 5G core is composed of various Network Functions (NFs) traditionally hosted in a dedicated physical "box". In today's virtualized deployments, such NFs become virtualized Network Functions (VNFs) backed by various hosting technologies (e.g., VMs or containers).

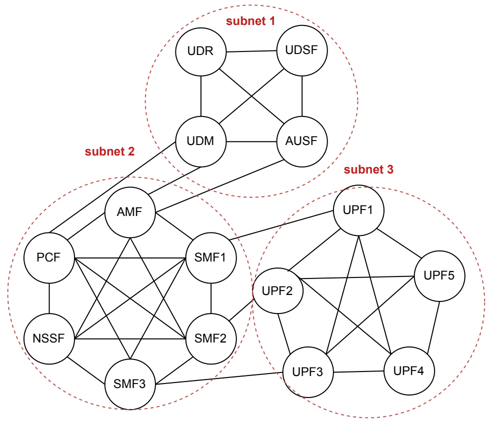

Fig. 4. A 5G Core Network Topology

Figure 4 shows our test instance of a 5G core network with three subnets, namely, subnet-1 and subnet-2, which are in charge of subscriber data storage and management in the control plane (e.g., UDR, SMF), and subnet-3, which includes the user plane components (e.g., UPF) [30]. Nodes within a subnet have direct network access to each other (unless restricted explicitly) while the connections between subnets are subject to typical logical connections (e.g., through firewalls).

Vulnerability assignment. We use a real-world dataset of vulnerability scans from a large telecom network cluster, containing 95 vulnerability scan results from October 2021 to May 2022. The virtual machines are well-maintained, i.e., out of the 518 VMs, 425 had no vulnerabilities identified. Many VMs have common vulnerabilities due to the same or similar installed software packages or libraries. We randomly assign vulnerability sets from this dataset to network nodes.

Algorithm 1: General steps of calculating a network score

```
Input: $G \leftarrow$ Network topology
Output: P $\leftarrow$ Network score
// 1. Generate per node pair attack graphs and
   calculate their scores
for $T_i$ in $T$                                        ▷ T: target node set
do
    for $S_j$ in $S$                                    ▷ S: source node set
    do
        $AG(S_jT_i) \leftarrow$ generate per node attack graphs from $S_j$ to $T_i$
        $BN(S_jT_i) \leftarrow$ build a Bayesian network for $AG(S_jT_i)$
        $P(S_jT_i) \leftarrow$ calculate the node pair score from $S_j$ to $T_i$
    end
end
// 2. Calculate the target node score
for $T_i$ in $T$ do
    $P(Ti) = (P_{s1Ti} + P_{s2Ti} ... + P_{snTi})/n$   ▷ $P_{Ti}$: target node score
end
// 3. Calculate the network score
P $\leftarrow (P_{T1} + P_{T2} ... + P_{Tn})/n$          ▷ n: total number of nodes
```

Calculation. The steps for deriving a network score are summarized in Algorithm 1. We use a 5G core (Figure 4) to explain them. The network G has 15 nodes, all being potential targets. 1) For each target T_i, there are 15 source-target node pairs ($S_1 - T_i, ..., S_{15} - T_i$), making 225 pairs in total. After all APs for each node pair ($S_j - T_i$) have been generated, an AG is built to represent all attacks from S_j to T_i. The corresponding BN is created with assigned conditional probability tables (CPTs) [10]. Then node pair score $P(S_j - T_i)$ is calculated. 2) A per-target score, denoted as $P(T_i)$, is derived by averaging all node pair scores from all other sources. 3) The final network score is obtained by taking the average value of all aggregated node scores.

B. Evaluation Results

We evaluate the way our multi-target risk score reflects the expected trends in response to varying the number of vulnerabilities per node, their severity, and the node degree, which is related to the network topology change. We compare the results of our metric, that we refer to as *network score* in the rest of this section, to a *naive method* averaging the attack probability of all vulnerabilities in the network nodes. All measurements are conducted on a computer with an AMD 5800X CPU and 32GB RAM running Ubuntu 20.04 LTS.

# of vuls per node	**Network Score**	Naive method score
Number of vulnerabilities test		
1	**.171**	.390
2	**.476**	.390
3	**.569**	.390
Severity of vulnerabilities test		
1	**.065**	.317
1	**.189**	.508
1	**.275**	.575
1	**.492**	.734

TABLE I
TEST RESULTS FOR THE NUMBER OF VULNERABILITIES AND SEVERITY

Scan Date	**Network Score**	Naive method score
Increasing		
Apr 26	**.285**	.228
May 3	**.417**	.382
May 9	**.449**	.365
Decreasing		
May 10	**.402**	.290
May 17	**.326**	.220

TABLE II
CHRONOLOGICAL COMPARISON

Number of vulnerabilities per node. In this test, we assign a different number of vulnerabilities (with the same probability of 0.39) per node in each run. From the top half of Table I, we observe that the network score increases from 0.171 to 0.569 as the number of vulnerabilities per node increases. Our approach captures the cumulative impact of multiple vulnerabilities when those are added up, while the naive method score does not capture this trend because averaging will not reflect the increased risk related to the existence of multiple vulnerabilities.

Severity of vulnerabilities. We increase the severity of vulnerabilities for the same node from each run to test its impact on the network score. Interestingly, although both our network score and the naive method score are increasing as expected, we observe that the network score increases less significantly than the naive method. This is because our approach takes into account the causality between nodes moderating the increase, as opposed to the linear increase of the naive scores regardless of the connectivity (see the bottom half of Table I).

Network degree. As illustrated in Figure 4, our original 5G core network instance has three subnets. In this test, we merge the subnets to see the effect of topological (degree) changes, which may happen when the logical configuration (e.g., firewall rules) is updated. The 2-subnet version merges the control plane (subnets 1 and 2) into a singular subnet leaving the user plane (subnet 3) still separate. The 1-subnet version merges all three subnets into a singular subnet altogether. The overall network score typically increases when merging these subnets. As we merge subnets and thus network segmentation decreases, i.e., the degree of the nodes becoming higher, network security gets worse. The general trend is that the more paths an attacker can have to a target node, the higher score the node will have. By contrast, the naive method only considers the average probability of exploitation, unable to capture the inter-node influences (see Figure 5).

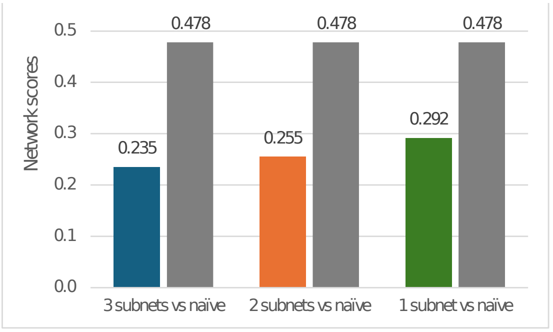

Fig. 5. Network score changes as subnets are being merged

Vulnerability removal. We examine the extent to which removing vulnerabilities in different components of a 5G network can affect the overall security score. Put another way, that is to evaluate the security improvements when vulnerabilities are patched. Initially, all nodes have two vulnerabilities and the network score is 0.520. We remove vulnerabilities from each node one by one and we present the results in Table III. We observe that the network score decreases as vulnerabilities are removed as expected, with the AMF contributing the most to the improvement with a drop of 15.5% in the network score. This is due to the high node degree and centrality of the AMF node (see Figure 4). From this, the system admin can conclude that patching vulnerabilities in the AMF is more effective, and should be prioritized over patching other nodes.

Vulnerability addition. One can also find out the impact of a node if the attacker discovers a new vulnerability of it. We start with a base case where each node has only one vulnerability (the corresponding network score is 0.61). Then, we add one more vulnerability per node to observe the change in the network score in percentage. Furthermore, to also accommodate the severity's effect, we use vulnerabilities with four different severity levels: 0.12, 0.49, 0.65 and 0.86. We select five nodes and present the results in Figure 6. Likewise, we can see that AMF has the most critical security impact on the network score because it is located at the centre of the network with a high degree of connectivity. Furthermore, the network score increases as the severity level assigned to the added vulnerability goes up, which is in line with the results from the previous severity test (higher severity leading to higher impact).

Remove from	Network score	% change	Naive score
AMF	0.439	↓15.7%	0.67
NSSF	0.493	↓5.2%	0.69
UDSF	0.510	↓1.9%	0.70
UDM	0.441	↓15.2%	0.67
UDR	0.496	↓4.7%	0.69
AUSF	0.489	↓6.0%	0.69
UPF1	0.470	↓9.7%	0.68
UPF2	0.473	↓9.2%	0.68
UPF3	0.474	↓8.9%	0.68
UPF4	0.494	↓5.0%	0.68
UPF5	0.509	↓2.1%	0.70
PCF	0.464	↓10.7%	0.68
SMF1	0.456	↓12.3%	0.67
SMF2	0.470	↓9.7%	0.68
SMF3	0.470	↓9.6%	0.68
None (Original)	0.520	0%	0.73

TABLE III
NETWORK SCORE CHANGE IN PERCENTAGE AS VULNERABILITIES ARE BEING REMOVED PER NODE

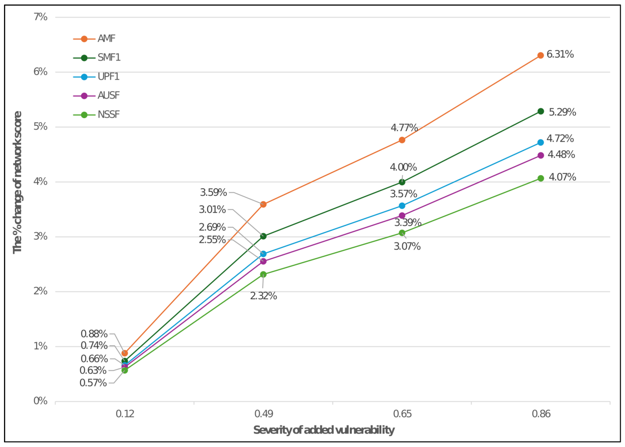

Fig. 6. Network score change in percentage as the severity of the added vulnerability increases

Chronological comparison. To demonstrate how the risk scores changes over time in the same 5G core network, we select two sets of vulnerability scans on different dates from the real-world dataset, with the same VMs to enable comparison. The top half of Table II shows the increase of the aggregated score across three dates, and the decrease of score between two dates. We observe that the increase was caused by the number of vulnerabilities on four of the nodes each changing from zero on April 26, to two on May 3, and then to nine on May 9, with vulnerabilities on all other nodes remaining the same. Likewise, the bottom half shows a decrease in network score, because one node with one vulnerability and another with two vulnerabilities had been patched when the network was scanned again on May 17.

Extended network types. In addition to the 5G core network, we also apply our approach to multiple types of network architectures. Those are: a 5G Mobile Access Edge Computing (MEC) with 100 nodes collected from [34], three hierarchical networks inspired by [35] respectively with 20, 500 and 1000 nodes, and a flat fully connected network with 300 nodes. We

Topology	# of nodes	**Score**	Naive method	AP Gen	BN Calc	Total time(s)
5G Core	15	**.363**	.313	2.19	0.031	2.411
5G MEC	100	**.117**	.069	2.20	19.65	23.54
Hierarchical-20	20	**.449**	.365	2.42	22.84	30.78
Hierarchical-500	500	**.030**	.040	12.0	21.44	89.39
Hierarchical-1000	1000	**.033**	.040	18.3	53.95	244.3
Flat	300	**.048**	.053	10.1	106.45	441.5

TABLE IV
TEST RESULTS FOR DIFFERENT NETWORK TYPES

present the test results in Table IV, including the network score and naive score. We measure the execution time for different steps of our approach including attack path generation (AP Gen), and BN calculation (BN Calc). We see that our approach can be applied to different types of network architectures and output a network score within a reasonable execution time. We observe that large networks take longer execution time because processing large networks (e.g., flat with 300 nodes or hierarchical with 500 and 1000 nodes) involves more computation and path traversal. On the other hand, the Bayesian calculation step takes a longer time if a network has more vulnerabilities and higher node degrees.

C. Discussion

We performed the tests to evaluate our approach from multiple aspects. Our approach accurately reflects not only the changes in the vulnerabilities (the changes in quantity and severity), and their evolution over time (the vulnerability additions and removals) but also reflects the cumulative impact of vulnerabilities as the connectivity and topology changes between different nodes (the network degree). In each of these cases, our approach demonstrated a better representation of the real security posture than the naive approach. For example, in networks with high-degree nodes (i.e., nodes with many neighbouring nodes), our method can show the score increases as the connections go up, but the naive method cannot. Therefore, our approach is effective for network security evaluation.

V. RELATED WORK

There is a large body of research in security metrics, as has been surveyed in the literature [1], [2], [4], [36], [37]. Here we discuss existing graph-based works that perform multi-system aggregation or risk assessments for networks, thus close to our work. Numerous works [12]–[16] use attack graphs and CVSS metrics to aggregate the vulnerability metrics. Noel et al. [12] model the composition of vulnerabilities in an attack through an attack graph, to quantitatively analyze the security risk in a networked system. Cheng et al. [13] propose to drill down to individual CVSS metrics (instead of just the base score) and compare several aggregation options (including average, max and attack graph-based). Homer et al. [14] explicitly propose an approach to aggregating vulnerability metrics through attack graphs where individual CVSS scores are referred to as component metrics, but still based on one specific attack target. Wang et al. [15] propose a CVSS-based dynamic risk assessment model that uses attack paths to model an attacker's capability and estimate the successful probabilities of a vulnerability exploitation. Ahmed et al. [16] conduct a survey on the aggregation of security metrics including gaps and open issues, and put forward a brief reference architecture using "probes" to collect measurement data without further explanation. However, these works only consider a single system as the target, hence not deriving an overall score for the entire network. Additionally, Yusuf et al. [38] proposed a composite analysis for network security metrics based on the cost of attacks. Nevertheless, these composite metrics are not further aggregated to reflect the network-level security.

VI. Concluding Remarks

This paper discussed an approach to deriving a risk score for the entire network using APs and BNs without being restricted to a single target. Notably, various forms of aggregation with their security implications were discussed from which we chose to apply a pair-wise aggregation method. The effectiveness of our approach was demonstrated by applying it to a 5G core network for various use cases. While there could be further improvements, such as using non-CVSS metrics, adaptive sensitivity for score changes, which we consider as future work, we intended in this paper to shed light on the research of multi-target security score aggregation due to its importance for actionable security in today's complex IT environments.

References

[1] M. Pendleton, R. Garcia-Lebron, J.-H. Cho, and S. Xu, "A survey on systems security metrics," *ACM Computing Surveys (CSUR)*, vol. 49, no. 4, pp. 1–35, 2016.

[2] J.-H. Cho, S. Xu, P. M. Hurley, M. Mackay, T. Benjamin, and M. Beaumont, "Stram: Measuring the trustworthiness of computer-based systems," *ACM Computing Surveys (CSUR)*, vol. 51, no. 6, pp. 1–47, 2019.

[3] Z. Abbadi, "Security metrics: What can we measure," in *Open Web Application Security Project (OWASP)*, vol. 2, 2011.

[4] C. C. Herley and P. C. Van Oorschot, "SoK: Science, security and the elusive goal of security as a scientific pursuit," in *IEEE S&P'17*, 2017, pp. 99–120.

[5] M. Ridley, C. Otero, D. Elliott, and X. Merino, "Quantifying the security posture of containerized mission critical systems," in *SoutheastCon 2018*, 2018, pp. 1–4.

[6] X. de Carné de Carnavalet and M. Mannan, "From very weak to very strong: Analyzing password-strength meters," in *Network and Distributed System Security Symposium (NDSS 2014)*. Internet Society, 2014.

[7] America's Cyber Defense Agency, "BOD 23-02: Mitigating the Risk from Internet-Exposed Management Interfaces," available at: https://www.cisa.gov/news-events/directives/binding-operational-directive-23-02.

[8] T. Madi, H. A. Alameddine, M. Pourzandi, and A. Boukhtouta, "NFV security survey in 5G networks: A three-dimensional threat taxonomy," *Computer Networks*, vol. 197, p. 108288, 2021.

[9] L. Wang, A. Singhal, and S. Jajodia, "Toward measuring network security using attack graphs," in *ACM Workshop on Quality of Protection*, ser. QoP '07, Alexandria, Virginia, USA, 2007, p. 49–54.

[10] M. Frigault and L. Wang, "Measuring network security using bayesian network-based attack graphs," in *IEEE International Computer Software and Applications Conference*, 2008, pp. 698–703.

[11] S. Dambra, L. Bilge, and D. Balzarotti, "SoK: Cyber insurance–technical challenges and a system security roadmap," in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020, pp. 1367–1383.

[12] S. Noel, L. Wang, A. Singhal, and S. Jajodia, "Measuring security risk of networks using attack graphs." *IJNGC*, vol. 1, 01 2010.

[13] P. Cheng, L. Wang, S. Jajodia, and A. Singhal, "Aggregating CVSS base scores for semantics-rich network security metrics," in *2012 IEEE 31st Symposium on Reliable Distributed Systems*. IEEE, 2012, pp. 31–40.

[14] J. Homer, S. Zhang, X. Ou, D. Schmidt, Y. Du, S. R. Rajagopalan, and A. Singhal, "Aggregating vulnerability metrics in enterprise networks using attack graphs," *Journal of Computer Security*, vol. 21, no. 4, pp. 561–597, 2013.

[15] T. Wang, Q. Lv, B. Hu, and D. Sun, "Cvss-based multi-factor dynamic risk assessment model for network system," in *2020 IEEE 10th International Conference on Electronics Information and Emergency Communication (ICEIEC)*, 2020, pp. 289–294.

[16] Y. Ahmed, S. Naqvi, and M. Josephs, "Aggregation of security metrics for decision making: a reference architecture," in *Proceedings of the 12th European Conference on Software Architecture: Companion Proceedings*, 2018, pp. 1–7.

[17] P. Mell, K. Scarfone, and S. Romanosky, "Common vulnerability scoring system," *IEEE Security & Privacy*, vol. 4, no. 6, pp. 85–89, 2006.

[18] The MITRE Corporation, "Common vulnerabilities and exposures," available at: https://www.cve.org/.

[19] NIST, "National vulnerability database," available at: https://nvd.nist.gov/.

[20] L. Wang, T. Islam, T. Long, A. Singhal, and S. Jajodia, "An attack graph-based probabilistic security metric," in *Data and Applications Security XXII*, V. Atluri, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 283–296.

[21] D. Dolev and A. Yao, "On the security of public key protocols," *IEEE Transactions on information theory*, vol. 29, no. 2, pp. 198–208, 1983.

[22] Tenable Nessus, available at: https://www.tenable.com/products/nessus.

[23] Greenbone, "OpenVAS – open vulnerability assessment scanner," available at: https://openvas.org/.

[24] H. Zhang, F. Lou, Y. Fu, and Z. Tian, "A conditional probability computation method for vulnerability exploitation based on CVSS," in *IEEE DSC'17*, 2017, pp. 238–241.

[25] NIST, National Vulnerability Database, "CVE-2020-17530," available at: https://nvd.nist.gov/vuln/detail/CVE-2020-17530.

[26] H. Binyamini, R. Bitton, M. Inokuchi, T. Yagyu, Y. Elovici, and A. Shabtai, "An automated, end-to-end framework for modeling attacks from vulnerability descriptions," *arXiv preprint arXiv:2008.04377*, 2020.

[27] Z. Han, X. Li, Z. Xing, H. Liu, and Z. Feng, "Learning to predict severity of software vulnerability using only vulnerability description," in *IEEE International conference on software maintenance and evolution (ICSME)*. IEEE, 2017, pp. 125–136.

[28] M. U. Aksu, K. Bicakci, M. H. Dilek, A. M. Ozbayoglu, and E. i. Tatli, "Automated generation of attack graphs using NVD," in *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy*, ser. CODASPY '18, Tempe, AZ, USA, 2018, p. 135–142.

[29] A. Ibrahim, S. Bozhinoski, and A. Pretschner, "Attack graph generation for microservice architecture," in *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, ser. SAC '19, Limassol, Cyprus, 2019, p. 1235–1242.

[30] 3GPP, "Technical Specification Group Services and System Aspects; System architecture for the 5G System (5GS);," 2023, 3GPP TS 23.501 V18.4.0 (2023-12).

[31] Open5GS, available at: https://open5gs.org/.

[32] Free5GC, available at: https://free5gc.org.

[33] OpenAirInterface, available at: https://gitlab.eurecom.fr/oai/openairinterface5g.

[34] B. Xiang, J. Elias, F. Martignon, and E. Di Nitto, "A dataset for mobile edge computing network topologies," *Data in Brief*, vol. 39, p. 107557, 2021.

[35] Cisco Networking Academy, *Connecting Networks Companion Guide*, 1st ed. Cisco Press, 2014, available at: https://www.ciscopress.com/store/connecting-networks-companion-guide-9781587133329.

[36] A. Ramos, M. Lazar, R. Holanda Filho, and J. J. Rodrigues, "Model-based quantitative network security metrics: A survey," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2704–2734, 2017.

[37] G. O. Yee, "Security metrics: An introduction and literature review," *Computer and Information Security Handbook*, pp. 553–566, 2013.

[38] S. Yusuf, J. Hong, M. Ge, and D. Kim, "Composite Metrics for Network Security Analysis," *Software Networking*, no. 1, pp. 137–160, 2017.