Evaluating Trainees in Large Cyber Exercises

Andrea Artioli[1,*,†], Mauro Andreolini[1,†], Luca Ferretti[1,†] and Mirco Marchetti[1,†]

[1]*University of Modena and Reggio Emilia, Modena, Italy*

Abstract

Cyber ranges are becoming a widely used alternative to teach trainees security through practice on realistic systems. They support evaluation and awareness through dashboards that display how many training objectives have been achieved in a cyber exercise. In our previous paper [1] we have outlined all the limitations of standard dashboards and proposed a new framework for modeling and assessing trainee activity through trainee graphs, reference graphs and scoring functions. In particular, we have introduced a scoring function based on the symmetric difference between a trainee and a reference graph, which allows to pinpoint quite effectively the inefficiencies of a trainee in an exercise.

In this paper, we show how that model, while working well for small and coherent exercises, poses several problems when applied to a larger labs made of several, heterogeneous challenges. The accuracy of the symmetrical difference rapidly drops down as the number of nodes and edges in the reference graph increases, thus making it impossible to use it in large environments. Furthermore, there might be edge cases where trainees that do not complete an exercise obtain a higher score than those who do. This happens because the symmetric difference turns out to be higher if the trainee advances exploring fewer nodes of the reference graph (which is the case of a skilled attacker).

To address these problems, we aim at reducing the complexity of the graphs fed to the scores. We improve the older model by representing an exercise as a set of smaller local graphs (each one for a coherent, intermediate challenge which can be assessed with a specific local score) and a global graph (representing the interconnection between intermediate challenges, that can be assessed with a specific global score). The main benefits of introducing global and local graphs using global and local progress are twofold: (a) having smaller graphs, scores related to precision (symmetric differences) are more performant; (b) we can assign different scores to different parts of the exercise, which is crucial in heterogeneous engagements.

We have implemented a Python-based simulator that generates random exercises and compares the performance of the previous and proposed trainee models under specific scores. Our results empirically show that, on average, the original model fails to scale to sizes in the order of tens of vertices and or edges, while the new is able to preserve precise scores locally and better track overall progress.

Keywords

Cyber range, Graph analytics, Training, Cyber exercise, Monitoring

1. Introduction

General consensus states that security is best learned by practice on realistic systems including workstations and servers, IoT devices and cyber-physical systems. A *cyber range* is a pre-

ITASEC24: Italian Conference on Cybersecurity, April 08–12, 2024, Salerno, Italy

*Corresponding author.

†These authors contributed equally.

andrea.artioli@unimore.it (A. Artioli); mauro.andreolini@unimore.it (M. Andreolini); luca.ferretti@unimore.it (L. Ferretti); mirco.marchetti@unimore.it (M. Marchetti)

0009-0007-7965-1322 (A. Artioli); 0000-0003-3671-6927 (M. Andreolini); 0000-0001-5824-2297 (L. Ferretti); 0000-0002-7408-6906 (M. Marchetti)

arranged virtual environment that allows trainees to simulate realistic attack and defense scenarios on an architecture resembling some original system. Cyber ranges typically support evaluation and awareness through dashboards that display how many training goals have been fulfilled. This approach does not allow to identify the reasons behind a bad trainee performance. To overcome this limitation, Andreolini et al. [1] introduce a novel, modular and extensible framework that captures trainee activity into a trainee graph and compares it with a reference graph provided by the instructor. Given specific learning objectives, novel scores are proposed in order to measure trainee proficiency in achieving them. A score is a scalar function over a set of graphs. Many scoring strategies have been considered to assess trainee speed (shortest path to a goal) and precision (combined symmetric difference between trainee edges/vertices and reference graph edges/vertices).

While these scores work well for small and coherent challenge environments (e.g., a single host executing a Web server), several problems arise when trying to apply this approach to a larger environment (e.g, heterogeneous multi-host labs located in different subnets). The accuracy of the symmetrical difference decreases abruptly as the number of nodes and edges in the reference graph increases. Furthermore, there might be edge cases where trainees that do not reach the exercise main goal obtain a higher score than those who do. To address these problems, we aim at reducing the complexity of the graphs fed to the scores. We model an exercise as a set of smaller local graphs (representing a coherent, intermediate challenge which can be assessed with a specific local score) and a global graph (representing the interconnection between intermediate challenges, that can be assessed with a specific global score). The main benefits of introducing global and local graphs using global and local progress are twofold: (a) having smaller graphs, scores related to precision (symmetric differences) are more performant; (b) we can assign different scores to different parts of the exercise, which is crucial in heterogeneous engagements.

We have implemented a Python-based simulator that generates random exercises and compares the performance of the previous and proposed trainee models under specific scores. Our results empirically show that, on average, the original model fails to scale to sizes in the order of tens of vertices and or edges, while the new is able to preserve precise scores locally and better track overall progress.

The paper is organized as follows. Section 2 briefly recalls the original scoring model introduced in [1], while Section 3 points out the limitations of the old model and introduces the new one. Section 4 describes our testbed and compares the performance of the old and new models under different scenarios. Section 5 discusses related work. Finally, Section 6 concludes the paper and outlines possible future work.

2. Background

2.1. Modeling trainee activities

In this section we briefly recall the original scoring model introduced in [1]. We model trainee activities through oriented graphs where vertices represent intermediate states that are reached by a trainee during an exercise, while edges represent the actions performed by a trainee to move from a particular state to the next one. Unintended actions are modeled as edges towards special

vertices representing errors. The *start* and *end* dummy vertices enclose a specific challenge. Both vertices and edges may be labelled with additional information (timestamps on vertices and edges to track progress over time, or command options on edges to better discern useful from useless commands). A *reference graph* is prepared by an instructor and models the ideal behavior of a trainee during an exercise. Figure 1 shows an example reference graph with intermediate steps S_i (i=1, 2, 3, 4, 5) carried out through actions a_k (k=1, 2, 3, 4, 5, 6, 7). A *trainee*

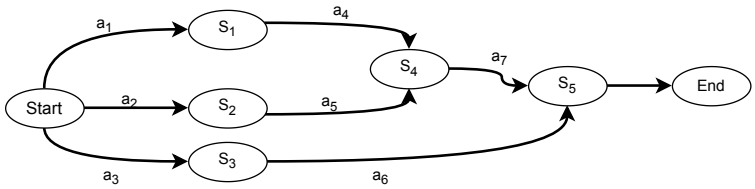

Figure 1: An example reference graph

graph tracks the actions performed by a trainee during an exercise. It is built automatically offline (at the end of an exercise, for a post-mortem performance analysis) or online (during an engagement, to track live trainee progress on a dashboard). The build process uses a reference graph and a set of metrics collected on the game network (e.g., command history and Web browsing history) during the exercise. These metrics allow to built an event timeline and to match it with the intermediate states of a reference graph. Whenever an event in the timeline matches an edge (V_i, V_j) in the reference graph, the trainee graph is updated as follows: vertex V_j is added to the trainee graph; vertex V_i is located in the trainee graph; an edge (V_i, V_j) is added to the trainee graph. Figure 2 shows the incremental update of a trainee graph with vertex S_2 and edge e_2. Matching of timeline events with trainee actions is done in ordered fashion on all nodes of the reference graph. If, on the other hand, a timeline event e cannot be

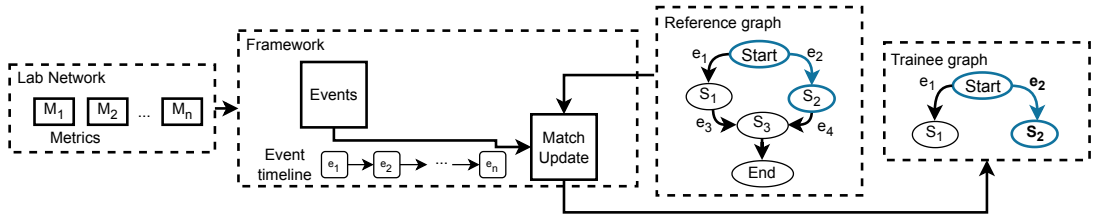

Figure 2: Incremental update of the trainee graph

matched against any edge in the reference graph, the trainee graph is updated as follows: the current vertex V_i in the trainee graph is located; the next expected vertex V_j in the reference graph is identified (such a vertex always exists, be it "start" if the trainee hasn't yet followed the recommended solution, or an intermediate one if the trainee has followed some steps of the exercise); the label N_j of the vertex V_j is identified; a new *dummy* vertex V_j is added to the trainee graph with label N_j_err (if it already exists, omit vertex insertion); an edge (V_i, V_j) is

added with label set to e. Figure 3 shows the insertion of a dummy node in a trainee graph with event e.

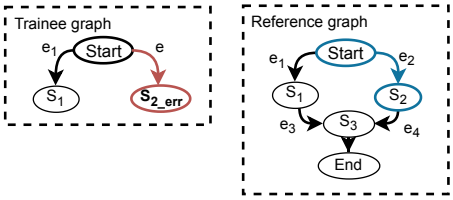

Figure 3: Insertion of a dummy node in the trainee graph

2.2. Scoring functions

A *score* is a function $f : G^n \to \mathbb{R}$ that takes as input a set of n oriented graphs (including at least one trainee and a reference graph), and outputs a real number s in a specified interval $[a, b]$ (e.g., $[0, 1]$). Different learning objectives call for different scores. We pick the following scores from [1] that will be used in the rest of the paper.

Score s_0. Let G_u be a trainee graph and G_r a reference graph. Let l_u and l_r be the length of the shortest path from the starts vertex to the end vertex in the trainee and reference graph, respectively. We define s_0 in the following equation:

$$s_0 = \begin{cases} \frac{l_r}{l_u} & \text{if the trainee completes the exercise} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

The rationale is as follows. If the trainee completes the exercise, the $\frac{l_r}{l_u}$ fraction rewards shorter exploitation paths (l_u is lower). An unsolved exercise is not rewarded.

Score s_1. Let $G_u = (V_u, E_u)$ be a trainee graph and $G_r = (V_r, E_r)$ a reference graph. We consider the *symmetric difference* of two sets A, B: $A \triangle B = (A \setminus B) \cup (B \setminus A)$. In other words, $A \triangle B$ contains all members of A not in B and all members of B not in A. We use two interesting properties of symmetric difference: (a) if two sets A and B coincide, $A \triangle B = \emptyset$; (b) if two sets A and B are disjoint, $A \triangle B = A \cup B$. We define the *efficacy* ν as following:

$$\nu = 1 - \frac{|V_u \triangle V_r|}{|V_u \cup V_r|} \tag{2}$$

If the trainee and reference graph coincide, we have $V_u \triangle V_r = \emptyset$, thus $\nu = 1$; the trainee has followed exactly the sequence of intermediate states modeled by the reference graph, and obtains the highest possible score. On the other hand, if the trainee and reference graph are completely disjoint, we have $V_u \triangle V_r = V_u \cup V_r$, thus $\nu = 0$; the trainee hasn't reached one single intermediate state of those in the reference graph, and obtains the lowest possible score. Intermediate performance produces scores in the $[0, 1]$ interval. Efficacy measures the ability of a trainee to follow the paths of a solution described in the reference graph.

Similarly, we define the *efficiency* η as following:

$$\eta = 1 - \frac{|E_u \triangle E_r|}{|E_u \cup E_r|} \tag{3}$$

The same observations hold for η. A trainee that performs the exact actions modeled in the reference graph is assigned the highest score, while a trainee that misses every action is assigned the lowest score. Intermediate performance produces scores in the $[0, 1]$ interval.

Score s_1 is defined as a linear combination of ν and η:

$$s_1(G_u, G_r) = \alpha\nu + (1 - \alpha)\eta \tag{4}$$

In this paper, we choose $\alpha = 0.5$, but the score may be tuned to weigh more efficacy or efficiency, according to the specific learning objective. This score rewards trainees who are able to reach the final goal of an exercise following the path defined by the reference graph. Trainees who could not reach the goal or make many mistakes are penalized with a low score. This is a good candidate for teaching exercises, since it tracks trainee actions during the exercise, taking into account the ability to follow the taught path to reach a goal.

3. A trainee model for large environments

3.1. Limitations of the original trainee model

While the original scoring model introduced in [1] works well for small and coherent engagements that can be evaluated consistently with a single scoring function, it poses several challenges in larger environments. With "larger environment" here we refer for example to labs made of multiple hosts with heterogeneous architectures, operating systems, libraries and applications, where a trainee has to showcase different skills (speed, precision, defense evasion, stealthiness, discovery of novel exploitation paths) in different contexts (UNIX, Windows, Web applications, networks, cryptography, reverse engineering). Under these circumstances it is often impossible to sum up trainee performance with a single scoring function. Furthermore, larger environments imply a larger number of interconnected hosts and services, resulting in larger trainee and reference graphs. Unfortunately, the accuracy of the symmetrical difference drops significantly as the number of vertices and edges in the reference graph increases. Even worse, in some pathological cases trainees that do not complete an exercise obtain a higher score than those who do. This happens because the symmetric difference turns out to be higher if the trainee explores fewer nodes of the reference graph (which is the case of a skilled attacker). These factors contribute to making the s_1 score useless.

To address these limitations we change the scoring model with the aim to reduce the complexity of the graphs fed to the scoring functions, as shown in the following section.

3.2. The proposed trainee model

Instead of modeling an exercise as a single, potentially large graph we split it as a set of k smaller, *local graphs* LG_i ($i \in [1, k]$) representing a single intermediate step. Figure 4 shows a

possible local reference graph with local start, end and intermediate vertices $LS_{i,start}$, $LS_{i,end}$ and $LS_{i,j}$ respectively ($i \in [1,k]$, $j \in [0,7]$). Local trainee graphs are built exactly as in the older model. Each local graph is assigned a scoring function $ls_i : G^n \to \mathbb{R}$ ($i \in [1,k]$) that assesses trainee performance in that specific intermediate step.

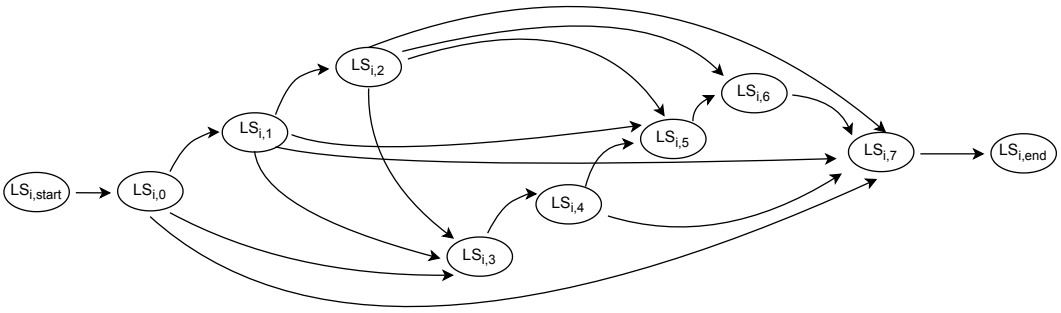

Figure 4: A local reference graph

Similarly, we create a *global graph* GG that models the interconnection between the intermediate steps. Figure 5 shows a possible global graph with global start, end and intermediate vertices GS_{start}, GS_{end} and GS_j respectively ($j \in [0,X]$). Global trainee graphs are built exactly as in the older model by observing transition events from an intermediate challenge to another. The global graph is assigned a scoring function $gs : G^n \to \mathbb{R}$ that assesses trainee performance in advancing through the intermediate challenges.

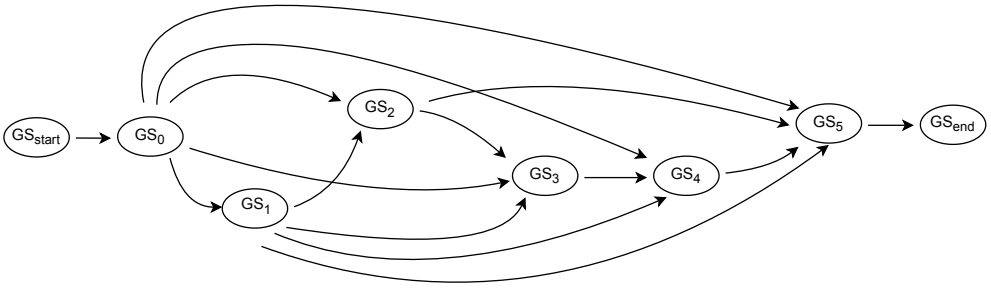

Figure 5: A global reference graph

In principle there is no impediment to assigning multiple scores to a specific graph (e.g., to simultaneously capture multiple skills such as speed and precision); however, for reasons of simplicity in this paper we will use only one score per graph. At the end of an exercise, the instructor has available:

- a set of $k+1$ graphs (GG, LG_1, LG_2, …, LG_k) that track out the overall advancement of a trainee through the intermediate challenges and in every single challenge;
- a set of $k+1$ scores (gs, ls_1, ls_2, …, ls_k) that quantify the aforementioned advancement and allow to compare it against the performance of other trainees (or even the same trainees over several, repeated exercises).

Aggregating these scores to produce a leaderboard is a very interesting topic that is out of scope for the current paper. Here, we show that the scores produced by the new model are more consistent than those of the old one in large graphs.

4. Results

In Section 4.1 we briefly discuss the simulator, the hardware/software testbed and the simulation scenarios carried out. In Section 4 we compare the performance of the s_1 score in the old and new model under different simulation scenarios.

4.1. Testbed

We have implemented a simulator that models an exercise based on a multitude of hosts organized in multiple networks. Each host exhibits a specific challenge which the trainee is supposed to solve. The trainee is also supposed to advance through the lab in a specific sequence. This is a quite popular model in the cyber exercise world (e.g. HackTheBox Pro Labs [2] and Offensive Security Proving Grounds [3]). The simulator automates the analysis proposed in this paper performing the following operations. It receives an input parameter the number of challenges k which, in this paper, is also the lab size in terms of number of hosts. We range $k \in [5, 60]$ hosts because current lab deployments are similarly sized. Then, it generates k nodes of the global graph and connects them randomly using the Watts-Strogatz algorithm which is used to create small world graphs. Then, for each node in the global graph a local graph is generated which represents the intermediate challenge. For simplicity in each local graph we use a constant number of ten nodes; we have verified that this design choice does not alter the main results of the paper. We also connect the nodes of the local graph randomly using the Watts-Strogatz algorithm. At this point, we have represented an exercise in the new model. To perform a fair comparison, we also generate the equivalent large reference graph in the old model by connecting the local graphs as shown by the global graph.

After setting up the graphs, the simulator performs random walks over the global graph and, for each node of the global graph, performs random walks into the associated local graph. To speed up simulations, local random walks are performed with two different error rates: 0.95 (related to a beginner) and 0.0 (related to an expert). The goal is twofold: (a) simulate the progress of a trainee through different hosts in the lab; (b) simulate the progress of both a high skilled and a rookie trainee into a specific challenge. At every step of the global and local random walks we compute the appropriate global and local scoring functions. Finally, at the end of an exercise the simulator saves the test configuration (k, the local and global reference and trainee graphs, the global and local scores) into separate files.

The simulator has been developed in Python 3.11.7 and uses the *networkx* package to handle large graphs. Simulations have been run in a host with an Intel Core 12700H CPU, 16 GB RAM and 1 TB SSD, running the ArchLinux GNU/Linux distribution.

4.2. Simulation results

Figure 6 shows the evaluation of scores s_0 and s_1 under different simulation scenarios. Labels have the form score/model/stats, where "score" is either the s_0 or s_1 score, "model" refers to the old-style or new-style trainee graphs (local or global in the latter case) and "stats" is either the minimum or maximum score value observed during a specific simulation. Let's focus on

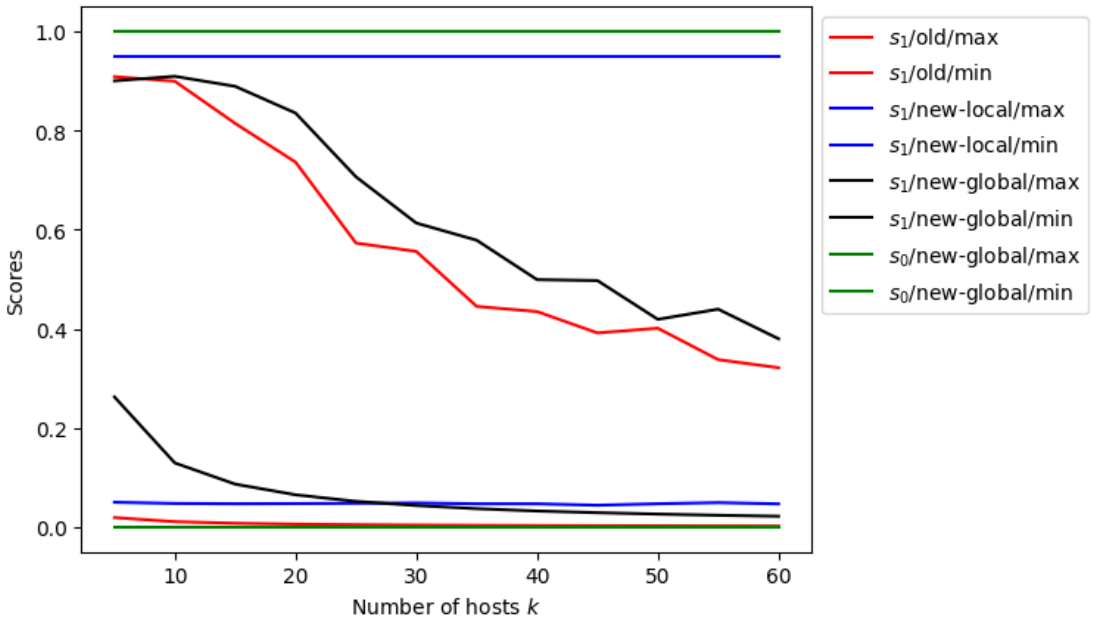

Figure 6:

the s_1/old/max and s_1/old/min labels that represent the maximum and minimum values of s_1 computed with the old model in exercises of increasing complexity in terms of vertices. As can be seen, the maximum value of the symmetric difference drops with k, making s_1 unusable starting with approximately $k = 15$ machines. The ratio of vertices in common between a trainee graph and a reference graph tends to drop as the number of vertices and edges increases. Keeping a high s_1 score would mean that an attacker exploits every node in exactly every possible way as defined by an instructor, which is very hard and often simply not needed. On the other hand, the minimum value stabilizes to zero for pretty much the whole range of considered hosts; this means that bad performance is consistently evaluated. A very similar behavior is exhibited by the s_1 score in the new model (labels s_1/new-global/*) when operating on larger and larger global trainee graphs.

Let us now focus on the other s_0/new-global/* and s_1/new-local/* labels that represent respectively s_0 in the new model (applied on a global trainee graph) and s_1 in the new model (applied to local trainee graphs). Here, the aforementioned limitations does not apply; both good and bad performance is consistently evaluated for the whole range of considered hosts. In particular, the s_0/new-global/max plot is consistently high because s_0 rewards very highly a trainee that is able to find the shortest exploitation path (which almost always happens in

repeated simulations). Regarding s_1/new-local/max, s_1 performs extremely well due to the constant and low number of ten nodes in every local graph generated by the simulator.

Table 4.2 shows the values of the s_1 score in a pathological simulation representing an exercise run in a lab of $k = 50$ nodes. We show only this value for reasons of space, but we have verified that the problem persists basically for every value of k, and is more frequent with increasing values of k. The old reference graph has a shortest path to the goal of length 1. Two trainees carry out the exercise, each one exhibiting different skills (a beginner with error rate 0.95 and an expert with error rate 0). The expert trainee finds the shortest path and completes the challenge, while the beginner trainee develops an incomplete exploitation path of length 9 and fails to complete the challenge. One would expect a higher score for the expert trainee with respect to the beginner trainee; however, this is not the case. The expert trainee receives a score of 0.035, while the beginner trainee receives a score of 0.07 (exactly doubled). The reason for this anomaly lies again in the way symmetrical graph difference works. The beginner trainee graph shares a longer path with the reference graph than the expert trainee would; this leads to a higher similarity of graphs and a higher s_1 score.

Error rate	Reference shortest path	Trainee shortest path	Completed	s_1/old
0	1	1	True	0.035
0.95	1	9	False	0.07

Table 1
Scoring anomalies in the old model

Table 4.2 compares the values of the s_0 and s_1 scores in the old and new model under the same pathological simulation. As we can see, the s_1 score still exhibits the same limitation and is thus useless as a scoring function. On the other hand, the s_0 score behaves correctly since it only focuses on path lengths and not on graph similarities. This leaves the question open as on how to fix s_1 to remove this anomaly. We reserve to address this problem in future work.

Error rate	Completed	s_1/new-local/[min-max]	s_1/new-global	s_0/new-global
0	True	$[0.95 - 0.95]$	0.027	1.0
0.95	False	$[0.10 - 0.20]$	0.07	0.0

Table 2
Scoring anomalies - old vs. new models

5. Related Work

Previous literature has explored formal representations to articulate an attacker's maneuvers in terms of the techniques utilized and vulnerabilities exploited within systems and configurations. Attack trees [4], bring the first structured representation of attacks on a system and the corresponding countermeasures, organized in a hierarchical tree format. The concept of attack trees has been broadened in academic research. Attack-defense trees [5] explicitly model interactions between attackers and defenders, enabling a more comprehensive and precise

security assessment compared to traditional attack trees. Attack-response trees [6] enhance attack-defense trees to account for uncertainties in intrusion detection such as false positives and negatives.

It is worth pointing out that as the number of vertices and edges in attack trees increases, their complexity escalates rapidly. Specifically, it becomes increasingly resource-intensive to trace all pathways from a leaf node to the root node. In realistic scenarios, the network of nodes and connections often surpasses thousands, where the mere addition of a single node significantly expands the number of arcs and potential attack routes. Moreover, given that the root node symbolizes the ultimate aim of the attack, complex multi-stage attacks may necessitate the utilization of multiple attack trees for accurate representation. Attack graphs [7] represent the infrastructure requiring protection, including network topology, vulnerable assets, and available exploits. They point out the pathways an attacker must traverse to achieve specific objectives. All the aforementioned approaches offer a static perspective of attacks and mitigation strategies, but they fail to track the progress of an attacker on a live system.

In the nascent stages of scoring systems research for cyber ranges, the main approaches are focused on signaling goal completion rather than evaluating trainee performance, as in [8], [9], [10]. To the best of our knowledge, [1] is the first proposal that models and assesses user activity in cyber exercises with the help of trainee graphs, reference graphs and scoring functions. However, as this paper shows, the original trainee and scoring model may perform poorly in large environments.

6. Conclusions

In this paper we have discussed our previously published trainee and scoring model [1] in the context of larger exercise labs. We have shown how the accuracy of the symmetrical difference decreases abruptly as the number of nodes and edges in the reference graph increases. Furthermore, there are pathological corner cases where trainees performing worse receive higher scores than those who perform better. We have tracked down the failures in the way the symmetric difference works and have devised a strategy to mitigate some of its flaws by feeding smaller graphs to the scoring functions. We have verified through simulations that our approach indeed improves the accuracy of the s_1 score.

Future work will be centered on fixing the scoring anomaly produced by s_1 in some corner case simulations and on implementing the new scoring model in the prototype presented in [1].

References

[1] M. Andreolini, V. G. Colacino, M. Colajanni, M. Marchetti, A framework for the evaluation of trainee performance in cyber range exercises, Mob. Netw. Appl. 25 (2020) 236–247. URL: https://doi.org/10.1007/s11036-019-01442-0. doi:10.1007/s11036-019-01442-0.

[2] htbprolabs, HackTheBox Pro Labs, https://www.hackthebox.com/hacker/pro-labs, 2017.

[3] ospg, Offensive Security Proving Grounds, https://www.offsec.com/labs/, 2020.

[4] B. Schneier, Attack trees, Dr. Dobb's journal 24 (1999) 21–29.

[5] B. Kordy, P. Kordy, S. Mauw, P. Schweitzer, Adtool: security analysis with attack–defense trees, in: International conference on quantitative evaluation of systems, Springer, 2013, pp. 173–176.
[6] S. A. Zonouz, H. Khurana, W. H. Sanders, T. M. Yardley, Rre: A game-theoretic intrusion response and recovery engine, IEEE Transactions on Parallel and Distributed Systems 25 (2013) 395–406.
[7] X. Ou, W. F. Boyer, M. A. McQueen, A scalable approach to attack graph generation, in: Proceedings of the 13th ACM conference on Computer and communications security, ACM, 2006, pp. 336–345.
[8] P. Čeleda, J. Čegan, J. Vykopal, D. Tovarňák, Kypo–a platform for cyber defence exercises, M&S Support to Operational Tasks Including War Gaming, Logistics, Cyber Defence. NATO Science and Technology Organization (2015).
[9] J. Vykopal, M. Vizváry, R. Oslejsek, P. Celeda, D. Tovarnak, Lessons learned from complex hands-on defence exercises in a cyber range, in: 2017 IEEE Frontiers in Education Conference (FIE), IEEE, 2017, pp. 1–8.
[10] M. Carlisle, M. Chiaramonte, D. Caswell, Using ctfs for an undergraduate cyber education, in: 2015 {USENIX} Summit on Gaming, Games, and Gamification in Security Education (3GSE 15), 2015.