Tracking the Bad Guys: An Efficient Forensic Methodology To Trace Multi-step Attacks Using Core Attack Graphs

Martín Barrère, Rodrigo Vieira Steiner, Rabih Mohsen, Emil C. Lupu
Department of Computing, Imperial College London, UK
{m.barrere, r.v.steiner, r.mohsen, e.c.lupu}@imperial.ac.uk

***Abstract*—In this paper, we describe an efficient methodology to guide investigators during network forensic analysis. To this end, we introduce the concept of *core attack graph*, a compact representation of the main routes an attacker can take towards specific network targets. Such compactness allows forensic investigators to focus their efforts on critical nodes that are more likely to be part of attack paths, thus reducing the overall number of nodes (devices, network privileges) that need to be examined. Nevertheless, core graphs also allow investigators to hierarchically explore the graph in order to retrieve different levels of summarised information. We have evaluated our approach over different network topologies varying parameters such as network size, density, and forensic evaluation threshold. Our results demonstrate that we can achieve the same level of accuracy provided by standard logical attack graphs while significantly reducing the exploration rate of the network.**

I. Introduction

Over the last years, the rise in cyber security threats combined with the sustained growth of networks involving multiple classes of devices has become an extremely challenging problem to network and security administrators. In 2016 alone, 6449 vulnerabilities were published by NIST and approximately 58% of them have been classified as high severity or critical vulnerabilities [29]. That is roughly 10 new non-negligible vulnerabilities per day on average. As the Internet of Things continues to evolve and the involvement of cyber-physical systems comes into scene, understanding security risks at a large scale becomes even more convoluted [11], [31]. Such reality makes it just too hard for security teams to keep up with the speed and diversity of the threat landscape over large complex networks. Therefore, there is a critical need for efficient tools able to handle available security information and help practitioners to broadly analyse network weaknesses and timely understand how security issues might be combined to amplify the attack surface (e.g. APTs [38]).

In that context, forensic investigations constitute a critical activity to ensure the health of a network at every operational level [43]. When an attack is identified, time becomes of the essence, and forensic investigations try to answer questions such as *how* the attack took place, *who* was the perpetrator, *when* the intrusion happened, and many more. Attack graphs [18], [2], [35], [21], [33] constitute a very powerful tool in this regard, primarily on the *how*, by assisting the investigator with information about the routes the attacker could have taken in order to get to his current location in the network [22]. The main objective of an attack graph is to depict the many ways in which an intruder may compromise assets in a network. However, today's complex networks pose hard challenges to the practical use of standard attack graphs which can easily increase in complexity as networks become denser and larger.

In this paper, we focus on a novel type of attack graph called *core attack graph* and its use in the context of network forensic investigations. Core attack graphs (or simply core graphs) aim at addressing attack graph complexity by identifying the main attack avenues towards specific targets in the network. By structurally summarising alternative routes between any two directly connected vulnerable network nodes, core graphs render compact representations able to reduce attack analysis complexity, ease visualisation aspects, and support efficient subsequent analysis. Our theoretical and practical results strongly suggest that core attack graphs can widely support efficient forensic investigations and therefore, having a significant impact in network security terms. Our main contributions are: (1) a mathematical model that formalises the generation and use of core graphs, (2) practical algorithms for the generation and exploration of core graphs, (3) an efficient forensic methodology based on attack graphs, and (4) a thorough comparative evaluation that shows the benefits of core attack graphs within forensic investigations, including significant reductions on the network surface exploration rate.

II. Related work

Attack graphs appear in many forms within the available literature and are well analysed and classified in scientific surveys [18], [2], [35], [21], [33]. Generation and visualisation tools are also covered [44], [17], [6], [16], [30], [13]. Well-known approaches include logical attack graphs [39], [30], state-based attack graphs [34], hierarchical attack graphs [15], [27], conservative attack graphs [8], multiple-prerequisite graphs [16], exploit dependency graphs [28], among others. While many studies essentially focus on analysis techniques, the mechanics and underlying objectives considered to produce attack graphs have a profound impact on the type of analysis that can be done over the generated graphs. In that context, the process of generating attack graphs faces hard challenges that have been well identified by the research community [5], [18], [19]. Though no consensus exists on a universal definition of attack graphs, logical attack graphs (LAGs) are probably the

978-3-901882-98-2 © 2017 Crown

most widely used [39]. LAGs provide a clear representation of potential attack paths which makes them highly useful for forensic investigations. However, as the number of devices and vulnerabilities increases, logical attack graphs become rapidly complex, thus challenging their use in practical settings. Core attack graphs aim at addressing such complexity.

While attack graphs model vulnerabilities and potential attack paths in a network, Wang and Daniels [41], [42] proposed a new graph model for network forensic analysis based on real intrusion evidence. These so-called *evidence graphs* are constructed *after* an attack occurs, mainly using network intrusion detection systems alerts, but also taking into account network flow and host logs as secondary evidence. Nevertheless, the results obtained from such graphs are constrained by the integrity of the collected evidence. Liu et al. [22] describe how to map evidence graphs to attack graphs and how they can be used to refine one another. While evidence information can be used to fine-tune probabilities in attack graphs, attack paths can be used to remove unrelated evidence tracks from evidence graphs. Liu et al. [23] also propose to augment attack graphs with anti-forensic information that can be used during forensic investigations to explain absent evidence. Zhang et al. [45] further demonstrate how attack graphs can be used to detect tampered evidence (falsified timestamps) in evidence graphs.

III. Core attack graphs

A. Core graph formalisation

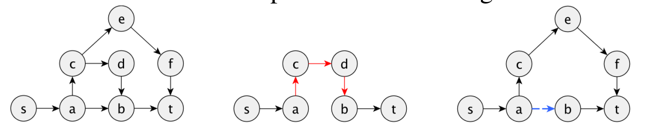
Given a directed graph G, a source node s, and a target node t, the objective of a core graph is to structurally grasp the main attack routes from s to t. Like standard attack graphs, nodes in a core graph represent host privileges (e.g. user, root privileges) while edges are based on reachable host vulnerabilities that require certain privileges and provide others. Core graphs are built in a systematic manner by summarising multiple alternative paths between any two connected nodes in the input graph G, and keeping in the core graph only the attack paths that cannot be summarised into any other graph link. We call these paths *core paths*. Mathematically, a core path coincides with what in graph theory is known as an induced path [26]. A path p is an induced path of G if any two adjacent nodes in p are connected in G and any two non-adjacent nodes in p are not connected in G. In other words, a core path between two nodes v_0 and v_n in G is a path p where each node $v_i \in p$ only connects to its immediate subsequent node v_{i+1} and there are no *shortcuts* to any other node ahead in the sequence over G.

Inflated paths, on the other hand, are paths that contain at least two non-adjacent nodes that are connected in G. In graph-theoretical terms, the relationship between core paths and inflated paths is an homeomorphism [20]. We use $p' \preceq p$ to denote that p is an inflated path of p'. We now formally define the concepts of core paths and core graphs.

Definition 1 (**Core Path**). *A path $p(v_0, \ldots, v_n)$ in $G = (V, E)$, $n > 0$, is a core path if and only if there is no other path $p' \neq p$ such that $p' \preceq p$, i.e.:*

$$\forall v_i \in p, \forall k \in [2, n-i], (v_i, v_{i+k}) \notin E \tag{1}$$

In addition, each edge $(v_i, v_{i+1}) \in p$ structurally reflects that v_i can not only reach v_{i+1} directly but also summarises all the alternative paths in which v_i can reach v_{i+1} over G. Let us consider the example illustrated in Figure 1.



(a) Paths $s \to t$ (b) Inflated path p_i (c) Core paths p_{c_1}, p_{c_2}

Figure 1: Superimposed paths, inflated path, and core path.

Figure 1a shows a small graph G with multiple paths from s to t. Figure 1b depicts one possible path p_i from s to t. It can be observed in the original graph G that a is connected to b and therefore path p_i violates Definition 1 since $(a, b) \in E$. On the other hand, path p_{c_1} ($\{s, a, b, t\}$), illustrated in Figure 1c, constitutes a core path from s to t since no shortcuts occur according to G, thus $p_{c_1} \preceq p_i$. In addition, the core edge (a, b) summarises both paths in G, namely, $\{a, b\}$ and $\{a, c, d, b\}$. Path p_{c_2} ($\{s, a, c, e, f, t\}$) is also a core path since nodes a and t are not directly connected and therefore $\{c, e, f\}$ cannot be summarised into any edge. Then, it holds that $p_{c_1} \not\preceq p_{c_2}$ and $p_{c_2} \not\preceq p_{c_1}$. Note that core paths are not about shortest paths: p_{c_1} has length 3 while p_{c_2} has length 5.

Definition 2 (**Core Attack Graph**). *Given a digraph $G = (V, E) \in \mathcal{G}$, a source node $s \in V$, and a target node $t \in V$, the corresponding core graph $C^G_{s,t} = (V_c \subseteq V, E_c \subseteq E, \omega_c)$ is the result of a transformation $\tau_{cg}(G, s, t)$ defined as the union of core paths from s to t in G as follows:*

$$\tau_{cg}(G, s, t) \equiv C^G_{s,t} \equiv \bigcup_{p \in P^G_{s,t}} p \quad s.t. \quad \nexists p' \in P^G_{s,t},\, p' \preceq p \tag{2}$$

where $P^G_{s,t}$ is the set of all paths from node s to node t in G; and $\omega_c : E_c \to \mathcal{G}$ is a function that given an edge $(v_i, v_j) \in E_c$ returns a subgraph $H \subseteq G$ that encodes the set of all possible paths $\{p_1, \ldots, p_k\}$ from v_i to v_j in G. Function ω_c is especially important as it allows to further explore the information summarised in the core graph in a hierarchical manner as explained in Section III-C.

```
Input: $G(V,E)$, Nodes $s, t$, TemporalPath $p$, Set $L$, $C^G_{s,t}$
Result: Complete core graph $C^G_{s,t}(V_C, E_C)$
$p' \leftarrow p \oplus s$;                          // add $s$ to the end of tmp path $p$
if $s \in V_C$ then                                  // node $s$ already visited
    add path $p'$ to core graph $C^G_{s,t}$;
else
    $N \leftarrow outgoingNodes(s, G)$;              // nodes reached by $s$
    if $t \in N$ then                                // target $t$ reachable from $s$
        $p' \leftarrow p' \oplus t$;                 // add $t$ to the end of tmp path $p'$
        add path $p'$ to core graph $C^G_{s,t}$;
    else
        $L' \leftarrow L \cup N \cup \{s\}$;         // reachable nodes by ancestors
        foreach node $n \in N$ do                    // explore neighbours of $s$
            if $n \notin L$ then
                // recursive call towards node $t$ with $n$ as source
                $genCoreGraph(G, n, t, p', L', C^G_{s,t})$;
            end
        end
    end
end
```

Algorithm 1: $genCoreGraph(args)$ (RECURSIVE).

B. Core graph generation algorithm

Core graphs are generated upon an input graph $G = (V, E)$ (e.g. logical attack graph), a source node s, and a target node t. The generation method, described in Algorithm 1, involves a systematic and efficient exploration of the input graph G that yields as a result the corresponding core graph $C^G_{s,t} = (V_C, E_C)$. Essentially, Algorithm 1 performs a recursive depth-first search (DFS) while carrying, at each recursive call, the set of nodes reachable by ancestors L in a breadth-first search (BFS) fashion (neighbours). In the worst case, the algorithm visits every node in G only once (ensured by step 2), and thus every outgoing connection (edge) is expanded and recursively analysed only once as well. Therefore, Algorithm 1 runs linearly in the size of the original graph $G = (V, E)$ with complexity $O(|V| + |E|)$ in the worst case. The lower bound is $\Omega(1)$ when the target is adjacent to the source node.

Note that a core graph is the result of a transformation of the original graph that is merely structural, i.e., there are no specific semantic considerations at this point other than the connectivity properties encoded in the original graph. However, our framework provides a solid basis for stochastic aspects as explained in sections IV-B and VI-A.

C. Core graph hierarchical expansion

Each edge (u, v) in a core graph $C^G_{s,t}$ summarises paths between u and v. However, these paths might be in turn composed of edges that summarise further paths. Therefore, there exists an intrinsic structural hierarchy induced by the way in which edges are summarised into others. In that context, we define an expansion procedure that progressively augments the original core graph by expanding its core edges in a hierarchical manner. This allows us to control the level of complexity of the core graph. More formally, given a core graph $C^G_{s,t} = (V_C, E_C, \omega_c)$, its complete expansion to the next level in the core edge hierarchy is controlled by the transformation $\tau_{hexp} : \mathcal{C} \to \mathcal{C}$ defined as:

$$\tau_{hexp}(C^G_{s,t}) = C^G_{s,t} \cup \Big(\bigcup_{e=(u,v)\in E_C} \{\tau_{cg}(\omega_c(e) - e, u, v)\} \Big) \quad (3)$$

For each core edge (u, v) in $C^G_{s,t}$, τ_{hexp} builds a sub-core graph from u to v using the original information encoded in G (retrieved with function ω_c) without the edge (u, v). The edge (u, v) is not considered since it summarises the sub-core graph we are trying to build. In this manner, τ_{hexp} computes the next level of summarised information for each edge and unifies the sub-core graphs with the input graph as the resulting one-level expansion. We denote as $C^{G,\delta}_{s,t}$ the graph obtained after expanding the original core graph δ times. Hence, $C^{G,0}_{s,t} = C^G_{s,t}$ (δ=0 means no expansion) and $C^{G,\delta}_{s,t} = \tau_{hexp}(C^{G,\delta-1}_{s,t})$. A full expansion ends when $C^{G,\delta}_{s,t} = LAG(G, s, t)$ for some $\delta \in \mathcal{N}$.

IV. Forensic model formalisation

A. Investigating graph nodes

Given an attack graph $G = (V, E)$, we model the forensic investigation of a node $m \in V$ (performed by an individual or an automated tool) as a function $F : V \to [0, 1] \times [0..1]$ that returns a tuple $F(m) = (r_m, c_m)$. The first element, $r_m \in [0, 1]$, is a boolean value that represents the compromise status of node m, that is, the outcome of the forensic examination over m ($0 \to$ *not compromised*, $1 \to$ *compromised*). The second element, $c_m \in [0..1]$, is a real value between 0 and 1 that captures the confidence level of the forensic investigator about the compromise status of node m. That is, a value of 0 indicates that the forensic investigator is completely unsure about the accuracy of the obtained results while a value of 1 indicates full confidence about the compromise status of m. Such uncertainty may arise due to factors such as lack of evidence, attack complexity, time constraints, among others. However, different investigators may also have different levels of expertise. As such, their opinions should be weighed differently. We model the skills of the investigator with the parameter $\lambda \in [0..1]$. Therefore, the dependability of the forensic result over m is defined as the combination of both the skills level of the investigator and his confidence about that specific result as $t^\lambda_m = \lambda \cdot c_m$.

If the investigator is very good but is not confident on a particular evaluation, the trust level of that result will be low. Conversely, if the investigator is not that good but his confidence is high, the trust level will be still balanced. The objective is to provide an overall forensic evaluation able to balance these aspects on a scale from 0 to 1. An evaluation of 0.5 (middle point) means that the investigator cannot confirm nor deny that node m has been compromised (complete uncertainty). An evaluation value towards 1 indicates that the investigator is more inclined to think the node has been compromised, whereas evaluations toward 0 mean the opposite. The overall forensic evaluation of a node is defined as follows.

Definition 3 (Forensic evaluation $e(m)$). *Given a node $m \in V$, the result of a forensic examination $F(m) = (r_m, c_m)$ and the trust level t^λ_m of that result, we define the overall forensic evaluation of m, $e(m) \to [0..1]$ as:*

$$e(m) = \begin{cases} \frac{(1-t^\lambda_m)}{2}, & \textit{if } r_m = 0 \\ \frac{(1+t^\lambda_m)}{2}, & \textit{if } r_m = 1 \end{cases} \quad (4)$$

The evaluation function $e(m)$ returns a real value between 0 and 1 which indicates the level of compromise of node m. The formulation looks to balance the boolean forensic result (r_m) according to how much we trust in the result. The closer the result is to the extremes (0 or 1), the more we trust the forensic evaluation. Note that the evaluation function always returns 0.5 when $t^\lambda_m = 0$ whether r_m is 0 or 1.

From a theoretical perspective, examining each required node in the network would provide enough information to find the most likely exploited attack path. Then, the problem could be technically solved, for example, using a weighted variation of the Dijkstra algorithm [9]. However, forensic examinations are complex and time-consuming, making such perspective infeasible in real networks. Our approach instead looks to leverage probabilistic attack graphs combined with the forensic evidence found along the process in order to reduce the network surface explored by the forensic investigator.

B. Probabilistic attack graphs

Probabilistic attack graphs are attack graphs enriched with probabilities that model the likelihood of compromise of each node in the graph based on their specific characteristics [14], [35], [32], [40], [12]. CVSS scores [7] are normally used to model the likelihood of an attacker moving among nodes in the attack graph. That is, given an edge $(m, n) \in E(G)$, the probability of compromising node n being at node m (conditional probability $p(n \mid m)$) is computed based on the CVSS scores of the vulnerabilities present in n that can be reached from m. The unconditional probability of a node m on the other hand, when computed from the source of the attack graph, denoted as p_m, represents how likely it is for an attacker to traverse the network and reach m given the current conditional probabilities encoded in the graph. To compute unconditional probabilities, we use Bayesian Networks in the case of acyclic directed graphs [24], [32], [12], and Monte Carlo simulations [3] otherwise. In the case of core graphs, the probability of a core link can be computed by analysing the subgraph encoded in it using the same techniques.

V. Forensic methodology

A. Exploration strategy

Let $G = (V, E)$ be the input graph of the exploration procedure (e.g. LAG). Our threat model positions the attacker in the source node $s \in V$. The forensic process starts when a graph node $n \in V$ is identified as compromised and the objective is to detect the route that the attacker could have taken from s to n. The first step is to identify the incoming links to n and order the neighbours to be explored $M = \{m|(m, n) \in E\}$ by analysing how likely it is for the attacker having reached n passing through each node $m \in M$. To do so, we use the unconditional probability of each incoming node m combined with the likelihood of reaching n as follows:

$$\begin{aligned} FP_{back}(n) = [m_1, m_2, \ldots],\ m_i \in M,\ s.t. \\ p_{m_i} p(n \mid m_i) \geq p_{m_{i+1}} p(n \mid m_{i+1}) \end{aligned} \tag{5}$$

Equation 5 defines the back forensic perimeter to be analysed from node n, arranged in descending order of likelihood. The forensic procedure first takes the most likely node m (the first in the list) and performs an individual forensic evaluation as described in Equation 4. Based on this forensic examination, the attack graph G is updated by setting the forensic result as the unconditional probability of the analysed node m.

As explained before, forensic results might not be conclusive or 100% certain. In that context, we define a confidence acceptance threshold T as the lower limit to accept nodes as *compromised*. Nodes to be considered for further exploration during the attack tracing procedure are selected as follows:

$$candidate(m, T) = \begin{cases} 1, & e(m) \geq T \\ 0, & otherwise \end{cases} \tag{6}$$

If node m classifies as a candidate, the procedure continues from m applying the same technique until the source s is reached. In order to provide a fair comparison, Algorithm 2 describes the overall strategy to be used with both LAGs and core graphs. In the case of core graphs, the input graph $G(V, E)$ is replaced with $C^G_{s,n}$. In the recursive call, G is replaced by the sub-core graph $C^G_{s,m}$ from s to m. This sub-core graph is needed because the construction of core graphs is target-dependent, and therefore, not all of the main attack avenues from s to m might be captured at the first level of the core graph $C^G_{s,n}$ since the target of the latter is n and not m.

```
Input: $G(V,E)$, Nodes $s, n$, Set $visited$, Threshold $T$, Path $p$
Result: Attack path $p$
if $s == n$ then                                    // source node reached
    $p \leftarrow p \oplus s$;                      // add $s$ to the end of path $p$
    return $success$;
else
    $L \leftarrow \{m|(m,n) \in E(G)\}$;            // incoming core nodes
    $prioritise(L)$;                   // order by likelihood as in Equation 5
    while $m = next(L)$ do                          // explore backwards
        if $m \notin visited$ then
            $visited \leftarrow visited \cup \{m\}$;
            if $candidate(m, T)$ then   // as defined in Equation 6
                $traceBack(G, s, m, visited, T, p)$;
                if $success$ then
                    $p \leftarrow p \oplus n$;   // add $n$ to the end of path $p$
                    return $success$;                  // route found
                end
            end
        end
    end
end
```

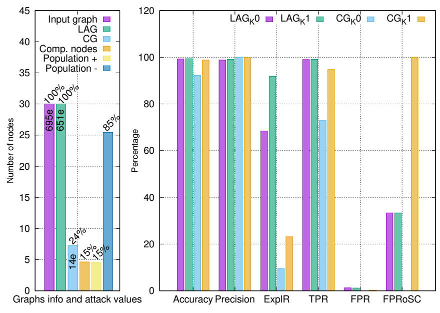
Algorithm 2: $traceBack(args)$ (RECURSIVE).

At a technical level, we also integrate a fall-back mode into the main strategy in case no attack path is found above the threshold. Therefore, the main algorithm always returns a path, even with low probabilities. The proposed exploration strategy is directed by the evidence found in the network and the advices provided by the probabilities encoded in the attack graph. Even though the likelihood encoded in the graph might represent the average attacker (based on CVSS), the evidence found during the process allows to adjust the search according to the actions taken by the attacker and thus, adapting to the attacker's behaviour to some extent. The trade-off between accuracy and performance applies to both LAGs and core graphs: the more we explore the network, the more accurate the results will be. Using core graphs however, we can analyse the main attack routes and refine the analysis only on those that are most likely to have been used by the attacker.

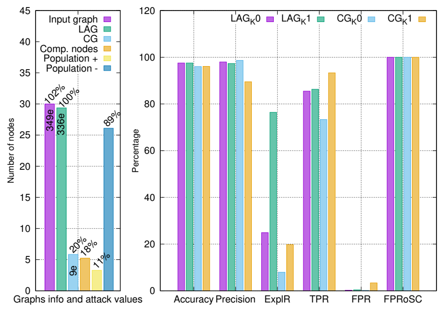
B. Refining most likely exploited attack paths

The main advantage of using core graphs is that forensic investigations are mostly carried out over the first level of the core graphs, thus avoiding the forensic evaluation of a significant number of nodes encoded in the core links. This technique allows to trace the skeleton of the main attack path which can be later refined by analysing the subgraphs encoded in the core links of the path.

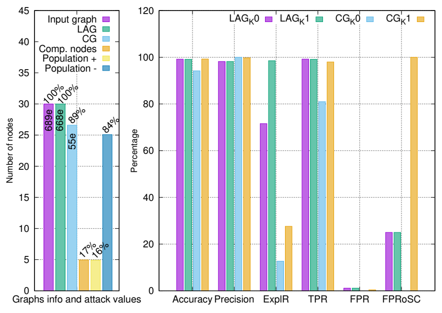
In order to fairly compare LAGs with core graphs, we propose a general refinement method that can be applied to both approaches. Once a main attack path has been obtained using Algorithm 2, the objective is to explore each link (m, n) of such path in order to identify whether the attacker has



(a) Pseudo-random cyclic graph



(b) Pseudo-random DAG



(c) Pseudo-random scale-free network

Figure 2: Performance analysis over different topologies.

exploited the direct link from m to n, or conversely, if an inflated path between the two nodes has been used instead. To do so, we define a parameter k that defines the number of neighbours of node n that will be additionally investigated in order to determine whether an inflated path from m to n was used or not. The exploration procedure is similar to Algorithm 2 and looks for the potential routes between m and n. However, this time the procedure can fail when no route is found, in which case the direct link between m and n is assumed to be the one used by the attacker.

This mechanism allows the refinement of the attack path found by Algorithm 2 but also alleviates the algorithmic greediness observed with LAGs. Since the probability of going from m to n directly will be, in most cases, higher than that of an inflated path from m to n, Algorithm 2 will often choose direct links over inflated paths. This happens because LAGs encode all the information on the same plane as opposed to core graphs which organise the information in a hierarchical manner. Therefore, this approach gives the chance to both LAGs and core graphs to further refine the main attack avenues and provide more accurate attack paths. We refer to these methods as LAG_{ki} or CG_{ki} where i is the number of additional neighbors investigated. As we show in Section VI, the refinement method does not need to consider too many neighbours in order to provide accurate results.

VI. Experimental Evaluation

A. *Technical and experimental setup*

The experiments presented in this section have been performed using Naggen, our attack graph generation tool [4], [25]. We evaluate three different topologies: pseudo-random cyclic graphs based on the Erdős-Rényi (ER) model [10], directed acyclic graphs (DAGs), and scale-free networks based on the Barabási–Albert (BA) model [1], [37]. Attacks are simulated with a random walk-based mechanism which uses the information encoded in the attack graph (CVSS probabilities).

B. *Performance analysis over different topologies*

Figure 2 illustrates the behaviour of our methodology over cyclic graphs, directed acyclic graphs, and scale-free networks. The size of the input graph is set to 30 nodes, the forensic expertise λ is set to 1, the edge creation probability is 0.8 (density of 80%), the probability distribution is based on CVSS, and the evaluation threshold T is 0.51. Each experiment performs 50 iterations (50 different input graphs) and averages the results using LAGs and core graphs for k=0 and k=1. The first block of each experiment depicts the average sizes of the input graph, LAG and core graph, as well as the average number of compromised nodes during attack simulations, nodes present (positive population) and not present (negative population) in the actual attack path. The second block shows six measurements of the exploration method including accuracy, precision, exploration rate, true positive rate (TPR, hits: nodes in the attack path), false positive rate (FPR, false alarms: nodes outside the attack path), and false positive rate over the compromised surface (how many false alarms are compromised nodes but not part of the attack path).

Figure 2a illustrates the results for cyclic input graphs. In the first block, we can observe that the core graph $CG(|V|{\sim}7, |E|{\sim}14)$ (third column) is considerably smaller than the logical attack graph $LAG(|V|{\sim}30, |E|{\sim}651)$ (second column). This is because the input graph is highly connected (ec=0.8) and the core graph is able to summarise a large portion of it. In the second block, we can see that the accuracy (first group) with LAG_{k0} and LAG_{k1} is close to 100% while the accuracy with core graphs is 92.27% with CG_{k0} and 98.80% with CG_{k1}. However, the exploration rate (third group) is significantly lower when core graphs are used. For k=1, the network surface explored with the LAG is about 91.87% (∼27 nodes) while only 23.13% (∼7 nodes) of the input network is explored using the core graph. The TPR with CG_{k1} (94.78%) is slightly lower though still acceptable considering the reduction on the exploration rate. The FPR is quite low in all of the cases while the FPR over the compromised surface is 100% with the core graph, which means that every node misclassified as part of the attack path was actually compromised nonetheless.

Figure 2b depicts the results over DAGs where the base line is defined using the LAG size as 100%. LAGs present a good performance though core graphs still show a significant reduction on the exploration rate. While the precision with CG_{k1} is a bit lower and there are some false alarms (FPR), the price to pay is still acceptable since the TPR is higher and the exploration rate (19.76%) is much lower than that of LAG_{k1} (76.43%). Figure 2c shows the results over scale-free networks [1], [37] where the core graph $CG(|V|{\sim}26, |E|{\sim}55)$ does not

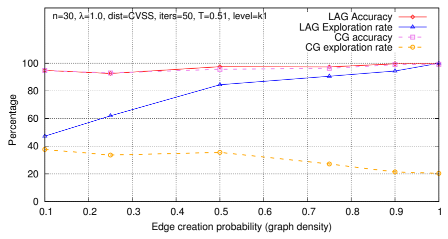


Figure 3: Edge creation probabilities.

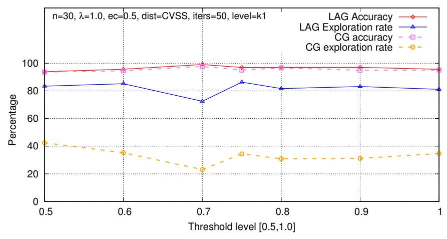


Figure 4: Various forensic thresholds.

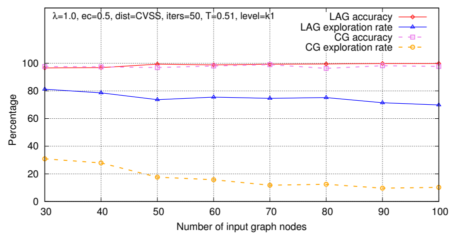


Figure 5: Scalability analysis.

reduce much in the number of nodes but it does reduce the number of edges compared with the $LAG(|V|{\sim}30, |E|{\sim}668)$. The results are very accurate with all of the methods though core graphs provide more efficient explorations as observed with CG_{k1} (27.60%) against LAG_{k0} (71.60%) and LAG_{k1} (98.53%). In summary, CG_{k1} dramatically reduces the exploration rate while keeping the same accuracy as LAGs.

C. Modifying the densities of the input graphs

The behaviour of the exploration method highly depends on the shape and size of the core graph, which in turn highly depends on the structure and density of the input graph. When the input graph tends to be less dense, with fewer cycles and more independent paths, the core graph will not show a significant structure reduction and the exploration rate will be pretty similar to that obtained with the LAG. However, real networks tend to be the opposite, i.e. denser and cyclic, in which case core graphs provide high levels of reduction when compared with standard attack graphs. Figure 3 shows the behaviour of the methodology for LAG_{k1} and CG_{k1} over cyclic input graphs where the creation of graph edges is controlled by the parameter ec. It can be observed that the accuracy level is quite similar for both methods. However, the LAG-based exploration method requires to investigate a huge portion of the network and almost every node when the input network tends to be fully connected. Core graphs, on the other hand, work better (summarise more information) when the input graph is denser, and therefore present lower exploration rates. This is due to the compactness of core graphs which permits to deliver more efficient forensic investigations since the exploration surface is highly reduced in most of the cases.

D. Changing the acceptance forensic threshold T

The experiment illustrated in Figure 4 shows the behaviour of both methods, LAG_{k1} and CG_{k1}, when the threshold T varies between 0.5 and 1. In this experiment, we consider a highly skilled investigator (λ=1) and an edge creation probability of 0.5. We can observe that both methods show similar levels of accuracy and also the same patterns on the exploration rates. This is due to the fact that the investigator is always certain about the individual forensic evaluations and therefore, increasing the forensic threshold does not significantly affect the exploration mechanism. However, reducing the confidence of the investigator tends to increase the exploration rate to some extent since more nodes may fall under the threshold. These factors equally affect both methods so it is expected to observe a similar behaviour on both of them.

E. Scalability analysis

We have also analysed scalability aspects over cyclic input graphs of various sizes. The objective here is to understand whether the size of the input network actually affects the reduction ratio between both methods, LAG_{k1} and CG_{k1}. Figure 5 shows the results on accuracy and exploration rate for both methods while varying the size of the input graph between 30 and 100 nodes. For n=30, the average sizes are $LAG(|V|{\sim}30, |E|{\sim}408)$ and $CG(|V|{\sim}18, |E|{\sim}54)$ while for n=100 we have $LAG(|V|{\sim}100, |E|{\sim}4853)$ and $CG(|V|{\sim}47, |E|{\sim}397)$. We can observe similar levels of accuracy for both methods while the exploration rates also present the same behaviour. However, core graphs consistently reduce the network surface to be explored in a significant manner which confirms our previous observations. We have also experimented with a P2P-Gnutella network dataset taken from the Stanford SNAP project [36], which consisted in 6301 nodes and 20777 edges. The obtained core graph, from s=0 to t=6299, has 1577 nodes (25.03%) and 3212 edges (15.46%).

VII. Conclusions and Future Work

In this paper, we show that forensic investigations supported by core attack graphs achieve the same level of accuracy provided by standard logical attack graphs while significantly decreasing the network exploration rate, and therefore, considerably reducing forensic efforts. We have evaluated both core and logical attack graphs over various network topologies, varying parameters such as number of nodes, density, and forensic evaluation threshold. In particular, core graphs perform even better when the network has a higher number of nodes and higher density. This is due to the hierarchical levels of summarised information provided by core graphs, which allow investigators to focus their efforts on critical nodes that are more likely to be part of attack paths, but also to explore deeper and refine their results when necessary.

We plan to further optimize the use of core attack graphs in network forensic investigations combining it with evidence graphs and anti-forensic information. We will also investigate scenarios where the output of the forensic evaluation of nodes could be wrong, due to factors such as low-skilled investigators or zero-day vulnerabilities.

Acknowledgements

This work was supported by the Engineering and Physical Sciences Research Council (EPSRC), UK, under grant number EP/L022729/1, and the Brazilian National Council for Scientific and Technological Development (CNPq) under Grant No. 206707/2014-2 awarded to Rodrigo Vieira Steiner.

REFERENCES

[1] R. Albert and A. L. Barabási. Statistical mechanics of complex networks. *Rev. Mod. Phys.*, 74(1):47–97, January 2002.

[2] M. A. Alhomidi and M. J. Reed. Attack graphs representations. In *Computer Science and Electronic Engineering Conference (CEEC), 2012 4th*, pages 83–88, Sept 2012.

[3] F. Baiardi and D. Sgandurra. Assessing ICT Risk through a Monte Carlo Method. *Environment Systems and Decisions*, 33(4):486–499, 2013.

[4] M. Barrère and E. C. Lupu. Naggen: a Network Attack Graph GENeration Tool. In *Proceedings of the IEEE Conference on Communications and Network Security (CNS'17)*, Oct 2017.

[5] G. S. Bopche and B. M. Mehtre. Attack Graph Generation, Visualization and Analysis: Issues and Challenges. In J. L. Mauri, S. M. Thampi, D. B. Rawat, and D. Jin, editors, *Security in Computing and Communications. SSCC 2014*, pages 379–390. Springer Berlin Heidelberg, 2014.

[6] M. Chu, K. Ingols, R. Lippmann, S. Webster, and S. Boyer. Visualizing Attack Graphs, Reachability, and Trust Relationships with NAVIGATOR. In *Proceedings of the 7th International Symposium on Visualization for Cyber Security*, pages 22–33. ACM, 2010.

[7] CVSS, Common Vulnerability Scoring System. http://www.first.org/cvss/. Last visited on October, 2017.

[8] S. A. DeLoach, X. Ou, R. Zhuang, and S. Zhang. Model-driven, Moving-Target Defense for Enterprise Network Security. In *Models@ run. time*, pages 137–161. Springer, 2014.

[9] E. W. Dijkstra. A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik*, 1(1):269–271, December 1959.

[10] P. Erdős and A. Rényi. On Random Graphs I. *Publicationes Mathematicae (Debrecen)*, 6:290–297, 1959.

[11] Questions and Answers: Responding to the 2017 Security Landscape. https://www2.fireeye.com/WEB-RPT-2017-Cyber-Security-Predictions.html. Last visited on October, 2017.

[12] M. Frigault and L. Wang. Measuring Network Security Using Bayesian Network-Based Attack Graphs. In *Proceedings of the 32nd Annual IEEE International Computer Software and Applications Conference COMPSAC, July 2008*, pages 698–703, 2008.

[13] N. Ghosh, I. Chokshi, M. Sarkar, S. K. Ghosh, A. K. Kaushik, and S. K. Das. NetSecuritas: An Integrated Attack Graph-based Security Assessment Tool for Enterprise Networks. In *Proceedings of the 2015 International Conference on Distributed Computing and Networking ICDCN, January 2015*, pages 30:1–30:10, 2015.

[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 Comp. Security*, 21(4):561–597, 2013.

[15] J. Hong and D. S. Kim. HARMs: Hierarchical Attack Representation Models for Network Security Analysis. In *Proc. of the 10th Australian Information Security Management Conference*, Dec 2012.

[16] K. Ingols, R. Lippmann, and K. Piwowarski. Practical Attack Graph Generation for Network Defense. In *In Proc. of the 22nd Annual Comp. Sec. Applications Conf., 2006. ACSAC'06*, pages 121–130, Dec 2006.

[17] S. Jajodia, S. Noel, P. Kalapa, M. Albanese, and J. Williams. Cauldron Mission-Centric Cyber Situational Awareness with Defense in Depth. In *Military Comm. Conf. MILCOM 2011*, pages 1339–1344, Nov 2011.

[18] K. Kaynar. A taxonomy for attack graph generation and usage in network security. *Journal of Inf. Security and Applications*, 2016.

[19] K. Kaynar and F. Sivrikaya. Distributed Attack Graph Generation. *IEEE Trans. on Dependable and Secure Computing*, 13(5):519–532, Sep 2016.

[20] A. S. Lapaugh and R. L. Rivest. The Subgraph Homeomorphism Problem. *Journal of Comp. and Syst. Sciences*, 20(2):133 – 149, 1980.

[21] R.P. Lippmann, K.W. Ingols, and Lincoln Laboratory. *An Annotated Review of Past Papers on Attack Graphs*. Project report IA. Massachusetts Institute of Technology, Lincoln Laboratory, 2005.

[22] C. Liu, A. Singhal, and D. Wijesekera. Mapping Evidence Graphs to Attack Graphs. In *IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 121–126, Dec 2012.

[23] C. Liu, A. Singhal, and D. Wijesekera. Using Attack Graphs in Forensic Examinations. In *7th International Conference on Availability, Reliability and Security*, pages 596–603, Aug 2012.

[24] L. Muñoz-González, D. Sgandurra, M. Barrère, and E. C. Lupu. Exact Inference Techniques for the Dynamic Analysis of Attack Graphs. *CoRR*, abs/1510.02427, 2015.

[25] Naggen - Network Attack Graph GENerator. http://demo.naggen.org/. Last visited on October, 2017.

[26] J. Nesetril and P. O. de Mendez. *Sparsity: Graphs, Structures, and Algorithms*. Springer Publishing Company, Incorporated, 2014.

[27] S. Noel and S. Jajodia. Managing Attack Graph Complexity Through Visual Hierarchical Aggregation. In *Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security*, VizSEC/DMSEC '04, pages 109–118, New York, NY, USA, 2004.

[28] S. Noel, S. Jajodia, B. O'Berry, and M. Jacobs. Efficient Minimum-Cost Network Hardening Via Exploit Dependency Graphs. In *Proc. of the 19th Annual Comp. Sec. Applications Conf.*, pages 86–95, Dec 2003.

[29] National Vulnerability Database, NIST. https://nvd.nist.gov/. Last visited on October, 2017.

[30] X. Ou, W. F. Boyer, and M. A. McQueen. A Scalable Approach to Attack Graph Generation. In *Proceedings of the 13th ACM Conference on Computer and Comm. Security (CCS'06)*, pages 336–345, 2006.

[31] PETRAS Internet of Things Research Hub. https://www.petrashub.org/. Last visited on October, 2017.

[32] N. Poolsappasit, R. Dewri, and I. Ray. Dynamic Security Risk Management Using Bayesian Attack Graphs. *IEEE Transactions on Dependable and Secure Computing*, 9(1):61–74, 2012.

[33] V. Shandilya, C. B. Simmons, and S. Shiva. Use of Attack Graphs in Security Systems. *Journal of Comp. Net. and Comm.*, 1(1), Sep 2014.

[34] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. M. Wing. Automated Generation and Analysis of Attack Graphs. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, SP'02, Washington, DC, USA, 2002. IEEE Computer Society.

[35] A. Singhal and X. Ou. Security Risk Analysis of Enterprise Networks Using Probabilistic Attack Graphs. NIST, Report 7788, 2011.

[36] Stanford Large Network Dataset Collection. https://snap.stanford.edu/data/index.html. Last visited on October, 2017.

[37] M. van Steen. *Graph Theory and Complex Networks: An Introduction*. Maarten van Steen, April 2010.

[38] N. Virvilis and D. Gritzalis. The Big Four - What We Did Wrong in Advanced Persistent Threat Detection? In *2013 International Conference on Availability, Reliability and Security*, pages 248–254, Sept 2013.

[39] L. Wang, M. Albanese, and S. Jajodia. *Network Hardening - An Automated Approach to Improving Network Security*. Springer Briefs in Computer Science. Springer, 2014.

[40] L. Wang, T. Islam, T. Long, A. Singhal, and S. Jajodia. An Attack Graph-Based Probabilistic Security Metric. In *Proceedings of the 22nd Annual IFIP WG 11.3 Working Conference on Data and Applications Security*, pages 283–296, Berlin, Heidelberg, 2008. Springer-Verlag.

[41] W. Wang and T. E. Daniels. Building Evidence Graphs for Network Forensics Analysis. In *21st Annual Computer Security Applications Conference (ACSAC'05)*, pages 11 pp.–266, Dec 2005.

[42] W. Wang and T. E. Daniels. A Graph Based Approach Toward Network Forensics Analysis. *ACM Transactions on Information and System Security (TISSEC)*, 12(1):4:1–4:33, October 2008.

[43] Collective work of all DFRWS attendees. A Road Map for Digital Forensic Research. In *Report From the First Digital Forensic Research Workshop (DFRWS)*, New York, NY, USA, August 2001.

[44] S. Yi, Y. Peng, Q. Xiong, T. Wang, Z. Dai, H. Gao, J. Xu, J. Wang, and L. Xu. Overview on Attack Graph Generation and Visualization Technology. In *2013 IEEE International Conference on Anti-Counterfeiting, Security and Identification (ASID)*, pages 1–6, Oct 2013.

[45] Y. Zhang, J. He, and J. Xu. Detecting falsified timestamps in evidence graph via attack graph. In *8th International Symposium on Computational Intelligence and Design (ISCID)*, volume 2, pages 369–374, Dec 2015.