Critical Attacks Set Identification in Attack Graphs for Computer and SCADA/ICS Networks

Alaa T. Al Ghazo, *Member, IEEE*, and Ratnesh Kumar, *Fellow, IEEE*

***Abstract*—Supervisory control and data acquisition/industrial control systems (SCADA/ICSs) networks are becoming more vulnerable to attacks that exploit the interdependence of security weaknesses at the atomic level to compromise system-level security. Attack graphs are an effective approach to depict these complex attack scenarios, assisting security administrators in determining how to best safeguard their systems. However, due to time and financial constraints, it is frequently not possible to address all atomic-level flaws at the same time. In this article, we propose a method for automatically detecting a minimal set of critical attacks that, when defended against, render the system secure. Finding a minimal label cut is typically an NP-complete problem. However, we propose a linear complexity approximation that uses the attack graph's strongly connected components (SCCs) to create a simplified version of the graph in the form of a tree over the SCCs. Then, we perform an iterative backward search over this tree to find a set of backward-reachable SCCs, as well as their outward edges and labels, in order to find a cut of the tree with the fewest labels, which is a critical attack set. We put our proposed method to the test on real-world case studies, such as IT and SCADA networks for a cyber–physical system for water treatment, and outperformed previous state-of-the-art algorithms in terms of approximation accuracy and/or computational speed. Our solution provides security administrators with a practical and efficient method for prioritizing efforts to address vulnerabilities in SCADA/ICS networks.**

***Index Terms*—Attack graph, critical attacks set, cyber–physical systems, industrial control systems (ICSs), Internet of Things (IoT), min label cut (MLC), SCADA systems, security.**

I. Introduction

SUPERVISORY control and data acquisition/industrial control systems (SCADA/ICSs) are utilized by numerous infrastructure monitoring and control systems, such as energy grids, nuclear facilities, transportation systems, and water/gas distribution networks. With the advent of cloud computing and the Internet of Things (IoT), these systems have become increasingly interconnected and Internet accessible, making them susceptible to cyber attacks [1], [2]. During the first half of 2018, 41.2% of ICS computers were targeted at least once, according to Kaspersky Lab researchers [3]. In 2016, the National Cybersecurity and Communications Integration Center (NCCIC) documented 290 cyber attacks against industrial control systems in the United States [4].

Attack graphs are a valuable tool for identifying complex attack scenarios and system-level vulnerabilities [5]. These graphs are directed, labeled graphs that illustrate the possible sequence of attacks an attacker may employ to exploit a system. Each vertex on the graph represents a specific system's status, such as the privileges of an adversary on various devices and assets. In addition, each edge represents a distinct attack that an adversary could execute to obtain additional privileges. By analyzing an attack graph, system administrators can gain an in-depth understanding of how an attacker could potentially compromise their system by exploiting the interdependence between component-level atomic vulnerabilities. Furthermore, attack graphs assist system administrators in optimizing the allocation of defensive resources by providing an informative representation of potential attack scenarios.

An attack path within an attack graph that an adversary can use to compromise a system-level security property is the path from the starting node to the final node. The objective of a system administrator is to define a minimum number of attacks that can be blocked so that there is no viable path between the initial and final nodes. In graph theory, we refer to this as a "cut." Since the attack graph is a directed labeled graph, determining the minimal number of attacks whose prevention would result in disconnection of the initial node from the final nodes is an example of a min label cut (MLC) [6], [7] that requires determining the minimal number of atomic attack labels whose edges, when removed from the graph, eliminate all paths from the initial node to the final node. A direct application of an MLC to define a set of critical assaults is computationally infeasible due to the fact that MLC is typically an NP-complete problem [6], [7], necessitating an approximation strategy.

In this article, we propose an approach to enhance the accuracy and computational efficiency of the MLC approximation. Specifically, we introduce an automated method to identify critical attack sets that has linear complexity. The first step involves computing and generating an abstract version of the attack graph, which is a tree over its strongly connected components (SCCs). SCCs are subgraphs of a directed graph that are strongly connected, meaning that there is a directed path from any vertex in the component to every other vertex in

Manuscript received 17 October 2022; revised 6 March 2023; accepted 1 May 2023. This work was supported in part by the National Science Foundation (NSF) under Grant NSF-CSSI-2004766 and Grant NSF-PFI-2141084, and in part by the Rockwell Collins through the NSF-IUCRC, S2ERC. This article was recommended by Associate Editor M. Qiu. *(Corresponding author: Alaa T. Al Ghazo.)*

Alaa T. Al Ghazo is with the Department of Mechatronics Engineering, Faculty of Engineering, The Hashemite University, Zarqa 13133, Jordan (e-mail: alghazo@hu.edu.jo).

Ratnesh Kumar is with the Department of Electrical and Computer Engineering, Iowa State University, Ames, IA 50014 USA (e-mail: rkumar@iastate.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TSMC.2023.3274613.

Digital Object Identifier 10.1109/TSMC.2023.3274613

the component. Next, we use a backward search over the abstracted attack graph, starting from the terminal node and progressing one hop per step, to identify the set of nodes that can reach the terminal node in an increasing number of steps. We also identify the labels of their outgoing edges, each of which creates a cut. Finally, we choose the cut with the minimal labels as the SCC-induced min label cut (SCCiMLC). By adopting this approach, we can substantially improve the accuracy and computational efficiency of the MLC approximation. Furthermore, our method is computationally tractable, making it a viable solution for practical applications.

To validate the proposed SCCiMLC algorithm, we present two case studies: 1) a computer network system and 2) a water treatment SCADA CPS system from the iTrust Lab [8]. To further determine the SCCiMLC algorithm's efficiency, we compare it against both an exact and an approximation algorithm for computing an MLC. For the former, we first generalize the exact MLC algorithm proposed by [9], which works only for a disjoint undirected graph with a label frequency of at most 2, to general labeled graphs. We also compare our SCCiMLC algorithm to the state-of-the-art approximation algorithm of [10], one shown to possess the best approximation factor reported in the literature. It proposes computing an approximation to an MLC by optimizing over a 0/1-polytope corresponding to an "approximate" hitting set (more details are included in Section II-C).

The main contributions of this article can be summarized as follows.

1) A novel linear, automated, SCCs based, a critical attack set identification algorithm called SCCiMLC.
2) An exact MLC algorithm inspired by [9], by way of its generalization.
3) Implementation of the proposed algorithm, the generalized exact algorithm, and a state-of-art approximation algorithm.
4) A comparison of the proposed SCCiMLC algorithm's performance to the other two implemented algorithms, by way of their validation against an IT network example and a water treatment CPS provided by the iTrust Lab [8]. The comparison and validation results are quite encouraging: the proposed SCCiMLC has the same accuracy as that of the exact algorithm and the same speed as that of the approximation algorithm, yet is more than 65 times faster than the exact algorithm.

A. Related Work

There are a few papers in the literature that have demonstrated that the MLC is an NP-complete problem and given alternate approximation algorithms to solve it. Sheyner et al. [6] demonstrated the NP-completeness of identifying the minimum critical attacks set by reducing the problem of minimum coverage to MLC. NP-completeness was established by [7] by reducing the Hitting-Set problem to MLC, and they also introduced a greedy algorithm to the Hitting-Set problem that selects the elements with the highest hit first. Zhang et al. [11] showed that a relaxed version of MLC (one that seeks a certain approximation to MLC) is itself NP-Complete. They proposed an approximation algorithm for finding a solution to MLC within a factor of $O(\sqrt{n})$, where n is the number of edges in the input graph. Dutta et al. [10] improved upon this work by proposing an approximating solution to MLC within a factor of $O(n^{2/3})$. Their approach computes a 0/1 polytope corresponding to an "approximate" hitting set. Zhang and Fu [9] proposed an exact algorithm for computing an MLC for the special case where the graph is a disjoint undirected graph with a label frequency of at most 2, in polynomial complexity. This article also proposed an approximation algorithm for a more general undirected graph (however, the attack graphs are directed graphs).

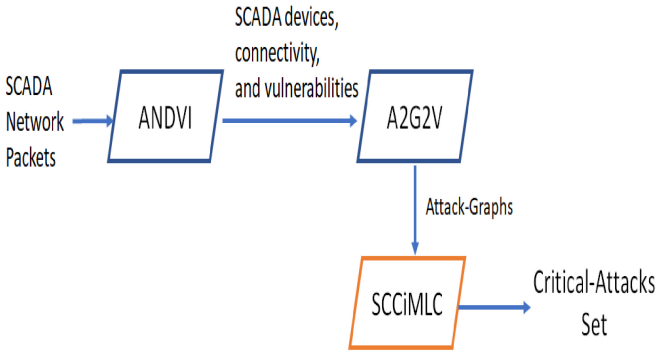

Fig. 1. Critical attacks set computation architecture.

The solutions put forth by other researchers also reduced the amount of time and money needed to implement security measures. Noel et al. [12] and Wang et al. [13] studied "minimum-cost network hardening" by exploring dependency among network components (as opposed to the more basic dependencies of atomic attacks as in the case of an attack graph). Cho et al. [14] used generalized stochastic Petri nets to quantitatively evaluate control network intrusion probability. Sawilla and Ou [15] and Sawilla and Burrell [16] proposed the use of Google's page rank-based approach for network security, in which the idea was to successively remove the highest page-ranked nodes to maximally decrease overall connectivity. Alhomidi and Reed [17] proposed a genetic algorithm for finding a minimum cut set in AND/OR dependency attack graphs. Hassin et al. [18] studied the related problem of finding a minimum label spanning tree and a minimum label path (but that does not necessarily induce a cut).

We leveraged our past work [19], [20], to generate and construct the attack graphs required to validate the proposed SCCiMLC. Our automated attack graph generation and visualization (A2G2V) is an algorithm for model-based automated generation of attack graphs that extends the initial work of [7]. A2G2V, in turn, relies on our method for the automated network device and vulnerability identification (ANDVI) tool [21] that passively observes network traffic to discover network connectivity and to fingerprint network devices, and next refers to vulnerability databases to enumerate device vulnerabilities as well as associated atomic attack actions. See the workflow in Fig. 1. A2G2V uses the output of ANDVI (device vulnerabilities/attacks and their connectivity) to enumerate all nesting of atomic attack actions to produce an attack graph that can compromise a system-level

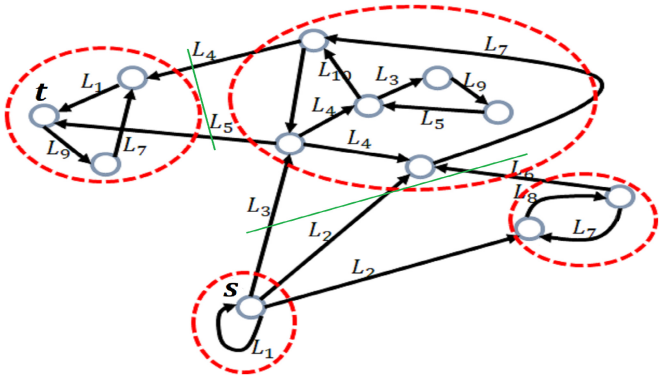

Fig. 2. Attack graph example.

security property, whereas the current paper proposes a new linear complexity SCCiMLC.

II. Critical Attacks Set Computation

Atomic attacks are the most basic form of attack that an attacker can carry out. (See examples in Section III.) An attack graph is an initialized and terminated graph capturing all possible sequences of atomic attacks that start from the source vertex and end at the terminal vertex. This is formalized in the following definition.

Definition 1: An attack graph is a tuple, $G = (V, E, s, t, L, \ell)$, where V is the set of vertices (each representing an attack status), $E \subseteq V \times V$ is the set of edges connecting one vertex to another (each representing an attack status change), $s \in V$ is the source vertex, $t \in V$ is the terminal vertex, L is a set of labels (each representing an atomic attack), and $\ell : E \to L$ is the function that labels each edge with an atomic attack.

Fig. 2 shows an example of an attack graph in which each edge sequence, beginning with the source vertex s and ending at terminal t, is an attack sequence that may jeopardize the system being attacked. Model-based approaches to computing an attack graph, given the atomic attack actions, system description, and security properties of interest, are reported in [7], [19], and [20].

Atomic attack actions can be eliminated by removing their root cause vulnerabilities (for example, by supplying their available software patches), which equates to the removal of their corresponding labeled edges from the attack graph. It is possible to mitigate a subset of the atomic attacks to protect the entire system, removing the corresponding labeled edges from the attack graph and disconnecting the terminal vertex from the source vertex. Such a subset of atomic attacks may therefore be viewed as causing a *labeled cut* in the graph.

Definition 2: Given a labeled attack graph $G = (V, E, s, t, L, \ell)$, a labeled cut is simply a set of labels $L' \subseteq L$ such that the removal of edges carrying those labels disconnects the source and terminal vertices. A set of attack labels whose disablement can disconnect a given attack graph is called a *critical attacks set*.

Finding a minimal critical attack set is an instance of an MLC problem [10]. MLC is known to be an NP-complete problem [10], [11].

A. Strongly Connected Components-Induced Min Label Cut

A solution to the MLC problem is approximated by the proposed SCCiMlC algorithm, which identifies a critical attack set in linear time in the following manner. It first generates an abstracted tree (i.e., an acyclic graph) version of the attack graph identified by the graph over the vertex set of its SCCs, found by using the depth-first algorithm of [22].

Definition 3: For a graph (V, E), a subgraph $(C \subseteq V, E_C \subseteq E)$ is an SCC if for all $u, v \in C$, there exists a path $v_1 = u \ldots v_n = v$, with $(v_i, v_{i+1}) \in E (1 \leq i \leq n-1)$, connecting u to v, such that $v_1, \ldots, v_n \in C$. The edge set E_C of the SCC C is simply defined to be $E_C \coloneqq E \cap (C \times C)$.

Fig. 2 depicts encirclements representing four different SCCs for its underlying attack graph. To obtain an SCCiMLC, our algorithm explores an abstracted acyclic graph defined over the vertex set of the SCCs of a given attack graph.

Definition 4: For the attack graph (V, E, s, t, L, ℓ), we obtain an abstracted graph over its set of SCCs, $(\mathcal{V}, \mathcal{E}, C_s, C_t, L, \ell)$, where $\mathcal{V}$ is the set of SCCs of (V, E); $\mathcal{E} \coloneqq \{(C, C') \in \mathcal{V} \times \mathcal{V} | \exists u \in C, v \in C', (u, v) \in E\}$ is the set of edges connecting any two SSCs; $C_s, C_t \in \mathcal{V}$ are the SCCs containing s, t, respectively (i.e., $s \in C_s, t \in C_t$); and for any edge $(C, C') \in \mathcal{E}$, $\ell((C, C')) = \{l \in L | \exists (u, v) \in E : u \in C, v \in C', l = \ell(u, v)\}$, i.e., it is the union of all labels of the edges connecting C to C'.

Note that the abstracted graph over the SCCs is acyclic, i.e., it is a tree graph. Finally, starting from the terminal node, SSCiMLC uses a backward search to iteratively define a set of nodes that can reach the terminal node in k or fewer steps ($k \geq 0$), as well as the associated cuts in the form of the outgoing edges of the k-step backward reachable nodes. The cut with the smallest number of labels then yields an SCCiMLC.

For example, in Fig. 2, starting from the terminal SCC, the source SCC can be reached in two hops. The label cut of the SCC edges at the first hop (shown by a green line) is of size 2 (consisting of labels $\{L_4, L_5\}$), while the label cut of the SCC edges at the second hop (also shown by another green line) is of size 3 (consisting of labels $\{L_2, L_3, L_6\}$). Thus, SCCiMLC for the attack graph of Fig. 2 is $\{L_4, L_5\}$.

An iterative backward search of one hop per iteration is performed over the abstracted tree graph over the SCCs starting from its terminal SCC C_t with an initial value of $\mathcal{V}_0 = C_t$, to identify the set of vertices $\mathcal{V}_k \subseteq \mathcal{V}$ that can reach the terminal one in k or fewer iterations. We also set the initial values of outgoing edges $\mathcal{E}_k \subseteq \mathcal{E}$ of $\mathcal{V}_k$ and their labels $L_k \subseteq L$, to be $\mathcal{E}_0 = L_0 = \emptyset$. These proposed SCCiMLC steps are formalized in Algorithm 1 that iterates over k until $\mathcal{V}_k = V$.

Remark 1: The algorithm picks i^* to be the index with the smallest number of labels ($i^* = \arg\min_{1 \leq i \leq k+1} |L_i|$); L_{i^*} is then the desired SCCiMLC that approximates the MLC. Identifying a minimum set of vulnerabilities as the optimum choice implicitly assumes that patching each such vulnerability has an identical cost, as is also the case with the cited prior related works. If the cost to patch different vulnerabilities are

Algorithm 1 SCCiMLC Algorithm

Input: $G = (V, E, s, t, L, \ell)$
Output: A critical attacks set, $CA \subseteq L$
main
Compute SCCs of G as in Definition 3
Generate the tree over the SCCs, $(\mathcal{V}, \mathcal{E}, C_s, C_t, L, \ell)$ as in Definition 4
Initialize $k = 0, \mathcal{V}_k = \{C_t\}, \mathcal{E}_k = \emptyset, L_k := \ell(\mathcal{E}_k)$;
While $\mathcal{V}_k \neq \mathcal{V}$:
  $\mathcal{V}_{k+1} := \{C \in \mathcal{V} | \exists C' \in \cup_{i \leq k} \mathcal{V}_i$ s.t. $(C, C') \in \mathcal{E}\}$;
  $\mathcal{E}_{k+1} := \{(C, C') \in \mathcal{E} | C \in \mathcal{V}_{k+1}, C' \in \cup_{i \leq k} \mathcal{V}_i\}$;
  $L_{k+1} := \ell(\mathcal{E}_{k+1}), k = k + 1$;
$i^* := \arg\min_{1 \leq i \leq k+1} |L_k|$;
$CA := L_{i^*}$.
end

Algorithm 2 MLC Exact Algorithm

Input: $G = (V, E, s, t, L, \ell)$
Output: A critical attack set, $CA \subseteq L$
main
Initalize $k = 0, L_k = \phi$;
while L_k does not induce a cut
  Pick any path $p \in G$ such that $\ell(p) \cap L_k = \emptyset$
  $L_{k+1} := L_k \cup \ell(p), k = k + 1$
Identify all subsets $\mathcal{A} := \{A \subseteq L_k\}$
  Pick a minimal subset $A \in \mathcal{A}$ that induces a cut
$CA := A$
end

different, then this variability of cost can be easily factored into our algorithm; in each iteration of the backward search over the graph of SCCs, we would compute the cost of the cut of that iteration by adding the cost of patching each vulnerability of that cut, and then we would pick the cut having the minimum cost. This can be achieved by simply replacing the definition of i^* in the "Terminate" step as: $i^* := \arg\min_{1 \leq i \leq k+1} \sum_{l \in L_i} c(l)$, where $c(l)$ is the cost of patching the vulnerability represented by $l \in L$.

Remark 2: In Algorithm 1, the generation of the SCCs, the construction of the tree over the SCCs, and the backward reachability over that tree to find an SCCiMLC, can all be performed in complexity that is linear with respect to the size of the given attack graph, i.e., in the number of its nodes plus edges. Hence, the proposed Algorithm 1 for finding SCCiMLC is also of linear complexity in terms of the number of nodes plus edges of the underlying attack graph.

We implemented the above algorithm for finding an SCCiMLC in the C language. To compare the results with respect to the state-of-the-art, we also implemented an exact MLC algorithm inspired by [9] and the approximation algorithm [10]; these two algorithms and their implementations are described below.

B. Implementing Exact MLC

In an attempt to provide a polynomial complexity algorithm for computing an exact MLC for a special case, Zhang and Fu [9] proved that when restricted to disjoint-path undirected graphs, MLC can be solved in polynomial time if the label-frequency (the number of times that a label appears in a path) is 2. For comparison with the proposed SCCiMLC, we extended the algorithm in [9] to a general directed-attack graph. Note that the time complexity of the extended algorithm is no longer polynomial but rather exponential in the size of the attack graph (as expected for an NP-complete problem).

Reference [9, Algorithm 3.4] is initialized by arbitrarily picking a path and all its edge labels, say $L_0 \subseteq L$ (those edge labels from a candidate set of labels to be removed as part of a label cut). Let L_k be the set of labels at the kth iteration (from past k selected paths). If cutting edges with labels L_k does not induce a cut, then there must exist a s–t path, none of whose labels are included in L_k. In the iterative step, such a path is found, and all its labels are added to L_k to obtain L_{k+1}. The iteration terminates when a set of labels L_k that induces a label cut is found. The size of this label cut can be further reduced by searching over all subsets of L_k to find a minimal subset $CA \subseteq L_k$ that also induces a cut. The exact algorithmic steps are formalized in Algorithm 2.

We implemented the above exact MLC algorithm in C and used the implementation to compare the performance against our own proposed algorithm in Section III.

C. Implementing State-of-the-Art Approximation Algorithm [10]

A few prior studies have proposed polynomial complexity algorithms for approximating a solution to the MLC problem. As noted above in Section I-A, some of those cannot be applied to general attack graphs, making them not suitable for comparison with our algorithm (for example, the algorithm in [7] requires an atomic attack to appear only once on an s–t path, which is generally not the case, as can be seen from our examples in Section III). Among the ones that are directly applicable to a general attack graph, we chose [10] that provides the best-known approximation factor (so it can be considered to be state-of-the-art) for implementation and comparison.

It is known from [7] that finding an MLC is an instance of a hitting-set problem: Given a set of labels, one for each s–t path, namely, $\mathcal{L} := \{L_p \subseteq L \mid p$ a s–t path in $G\}$, a hitting-set is a minimal subset $CA \subseteq L$ that intersects with each L_p, i.e., $\forall p \in G: L_p \cap CA \neq \emptyset$. Dutta et al. [10] proposed finding an approximation to the MLC solution by optimizing over a 0/1-polytope which corresponds to a hitting set for the set $\mathcal{L}$. Note a 0/1-polytope in d-dimensional space is the convex hull of a set of d-dimensional endpoints $Q \subseteq \{0, 1\}^d$, whose vertices only have 0/1-coordinates (i.e., it is a convex subset of the hypercube $\{0, 1\}^d$ with a constraint on its vertices being 0/1).

For formulating the hitting-set problem as a 0/1-polytope optimization problem, let $\mathcal{P}_G$ denote the set of all s–t paths in G. Define a set of endpoints in the $|L|$-dimensional unit cube

$$Q := \left\{ x \in \{0, 1\}^{|L|} \mid \forall p \in \mathcal{P}_G, \exists l_k \in \ell(p) : x(k) = 1 \right\}$$

where l_k denotes the kth label in the set L. In other words, each $x \in Q$ "selects" at least one label from each s–t path. Next, solve for

$$\min_{x \in Q} \sum_{k \in \{1,\ldots,|L|\}} x(k).$$

The solution to the above optimization problem provides an exact solution to MLC. To obtain an approximate solution, a relaxed version of the hitting-set problem is formulated in [10], where the end-points are allowed to be unit-interval valued numbers

$$\widehat{Q} := \left\{ x \in [0,1]^{|L|} \mid \forall p \in \mathcal{P}_G : |p| \leq |V|^{\frac{2}{3}}, \; \exists l_k \in \ell(p) : x(k) \geq \frac{1}{|V|^{\frac{2}{3}}} \right\}$$

and the following optimization is performed:

$$\min_{x \in \widehat{Q}} \sum_{k \in \{1,\ldots,|L|\}} x(k).$$

Let $x^* \in \widehat{Q}$ be the minimizer. Then, a label cut can be found as follows:

$$L' := \left\{ l_k \in L \mid x^*(k) \geq \frac{1}{|V|^{\frac{2}{3}}} \right\}$$
$$E' := \left\{ e \in E \mid \ell(e) \notin L' \right\}$$
$$L'' := \text{aMLC for } G' = \left(V, E', s, t, L - L', \ell\right)$$
$$CA := L' \cup L.''$$

For implementing the above hitting-set solution-based computation of an approximation to MLC, we used MATLAB code from [23], modifying the "separation oracle" function to match the separation hyperplane algorithm in [10].

III. Case Studies: Computer Networks and Water Treatment SCADA CPS

Realistic applications of a computer network example and a real-world water treatment testbed of the iTrust Lab [8] was chosen as case studies to demonstrate the applicability of the proposed SCCiMLC algorithm and to compare its performance to both the exact algorithm (our generalization of [9]) and the state-of-art approximation algorithm of [10].

A. Computer Network Case Studies

As a concrete example illustrating the problem of critical attacks set identification, we adapt the networked-system example from [6], [7], [19], and [20] shown in Fig. 3.

There are three hosts in the network: Host-0, where the attacker is located, and the two target hosts Host-1 and Host-2. Host-1 runs sshd and ftp, while Host-2 runs ftp and database. As a result, this system has four possible atomic attacks per host. There is also a firewall that separates the targets from the remainder of the network. The firewall places no restrictions on access control over network traffic flow. The network traffic between network hosts and external sources is supervised by an intrusion detection system (IDS). The IDS can monitor the flow of traffic between (Host-0; Host-1) and (Host-0; Host-2), but not between (Host-1; Host-2) because of its network positioning. The full system description and its implementation within our A2G2V tool can be found in [19] and [20]. Fig. 4 shows the automatically generated attack graph of this system using our tool A2G2V [19], [20], and that is also graphically displayed automatically by the tool. The A2G2V input includes network topology, defined in architecture analysis and development language (AADL) [24], the atomic attack actions for each network device (and their pre-/post-conditions) specified in the AADL AGREE annex [25].

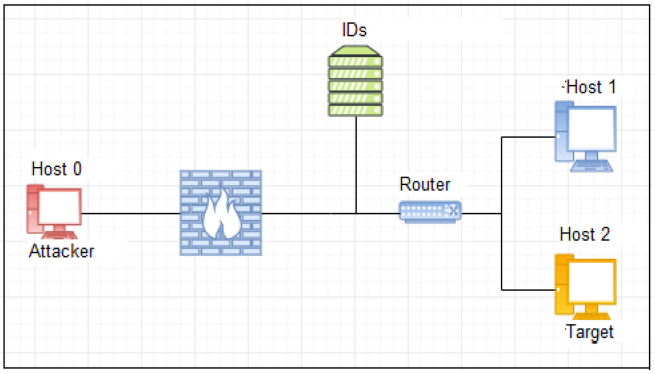

Fig. 3. 3-host networked-system example.

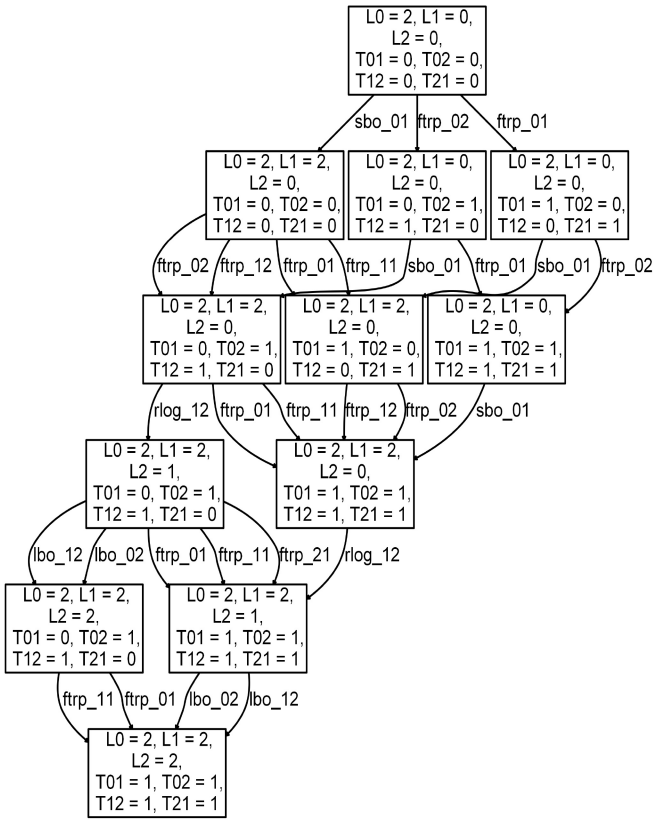

Fig. 4. 3-host networked-system generated attack graph.

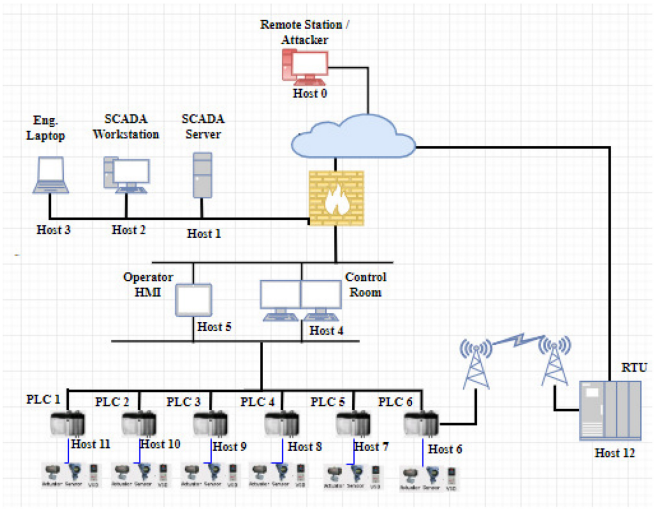

Fig. 5. SCADA architecture of water treatment system.

B. Water Treatment SCADA CPS Case Study

Fig. 5 shows a CPS water treatment testbed in the iTrust laboratory [8]. The system consists of thirteen hosts, Host-0 through Host-12: Host-0 is the remote station; Host-1 is the SCADA server; Host-2 is the SCADA workstation; Host-3 is the engineer's laptop; Host-4 is the control room; Host-5 is the operator's human machine interface (HMI); Host-6 to Host-11 are the programmable logic controllers (PLCs) that control the physical process through sensors and actuators; Host-12 is a remote terminal unit (RTU). We suppose that the attacker has been able to secure access to the remote station (Host-0). A firewall separates Host-0, Host-1, Host-2, Host-3, and the remainder of the system. The firewall restricts access to Host-1, Host-2, and Host-3 from the remote station. while allowing access to the remainder of the system from Host-1, Host-2, and Host-3. The communications with and among the PLCs go through Host-4 and Host-5. The PLCs communicate with Host-4 and Host-5 but not with each other. The remote station (Host-0) communicates with the RTU (Host-12). The RTU (Host-12) communicates with PLC-6.

The SCADA system's components and services are all Siemens products. In general, different SCADA devices may be programmed differently (e.g., to interact with different sensors or actuators). However, for the application, we considered the devices to have a large degree of homogeneity for the fact that their underlying software and OS have the same core. In the water treatment cyber–physical system of our case study, all the PLCs are Siemens PLC s7-1500, and although each has its own behavior, network traffic, and connectivity to the field devices (as they control different processes), they all use the same OS and have been programmed using the same software (Siemens TIA portal). This is also true for the SCADA monitoring devices, namely, the control room SCADA server.

Online databases [26], [27] list the following vulnerabilities in these devices.

1) The SCADA server is an SINEMA server, with these listed vulnerabilities.
   a) *iws:* An unauthenticated code can be remotely executed through the built-in Port 4999/TCP and Port 80/TCP Web server if the attacker has access to the server.
   b) *lmw:* If the affected products are not installed in their default file path (*C:\\ProgramFiles\\∗* or the localized equivalent), local users of Microsoft Windows can increase their privileges.
2) The SCADA workstation is an SIMATIC WinCC flexible runtime, with these listed vulnerabilities.
   a) *rmm:* The flexible remote control module in SIMATIC Wincc transmits weakly secured credentials over the network. Attackers who collect the module network packets can reconstruct the credentials.
   b) *inws:* Flexible Siemens Wincc allows attackers that have remote user access to insert arbitrary Web script or HTML through unspecified vectors.
3) The engineering Laptop runs a Wincc TIA portal, with these listed vulnerabilities.
   a) *bac:* When a user logs in, the program sets predictable authentication token/cookie values. This helps an attacker to avoid authentication checks.
   b) *inws:* The Wincc TIA portal allows remote attackers to insert arbitrary Web scripts or HTML using unknown vectors.
4) The HMI runs a Wincc runtime advanced, with these listed vulnerabilities.
   a) *rmm:* The flexible remote control module in SIMATIC Wincc transmits weakly secured credentials over the network. Attackers who collect the module network packets can reconstruct the credentials.
   b) *inws:* The Wincc TIA portal allows remote attackers to insert arbitrary Web scripts or HTML using unknown vectors.
5) The control center runs a Wincc runtime professional, with these listed vulnerabilities.
   a) *rmm:* The flexible remote control module in SIMATIC Wincc transmits weakly secured credentials over the network. Attackers who collect the module network packets can reconstruct the credentials.
   b) *inws:* The Wincc TIA portal allows remote attackers to insert arbitrary Web scripts or HTML using unknown vectors.
6) The PLCs are Siemens SIMATIC S7-1500 CPU PLC devices, with these listed vulnerabilities.
   a) *rng:* Inadequate entropy is present on Siemens Simatic S7-1500 CPU PLC systems in the random number generator leading to weak cryptographic security mechanisms that allow remote attackers to hijack sessions using unknown vectors.
7) The RTU (SICAM RTUs SM-2556 COM Modules), with these listed vulnerabilities:
   a) *osdi:* The built-in Internet server for affected devices (port 80/TCP) can allow remote attackers to access network-based sensitive device data.
   b) *uxac:* The integrated Web server (80/TCP port) could be used to execute arbitrary code on

the affected devices by unauthenticated remote attackers.

The above-listed SCADA system atomic vulnerabilities can be exploited, resulting in the following atomic attacks.

1) *Remote Code Execution (rce):* This attack exploits the vulnerability in the SINEMA embedded Web servers in Port 4999/TCP and Port 80/TCP, if an attacker has access to the server. This attack gives the attacked user access host.
2) *Unquoted Service Paths (usp):* This attack takes advantage of the vulnerability in the SINEMA server window operating system, enabling local users to increase their rights to get root access.
3) *User Credentials Construction (ucc):* The attack uses flexible SIMATIC Wincc, advanced SIMATIC Wincc, and SIMATIC Wincc professionals with weakly secured credentials. The attack allows users access to the host they have targeted.
4) *Cross-Site Scripting (xss):* In the Wincc TIA portal, SIMATIC Wincc flexible, SIMATIC Wincc advanced, and SIMATIC Wincc professional, this attack exploits the vulnerability of the Web script and HTML code injection. This attack provides root access to the host attacked by the attacker.
5) *Authentication Token/Cookie (atc):* This attack takes advantage of a Wincc TIA portal's predictable authentication token/cookie values. This attack provides root access to the host attacked by the attacker.
6) *Cryptographic Protection Mechanisms (cpm):* This attack takes advantage of the weakness in the random number generator of the SIMATIC S7-1500 CPU if the attacker has access to the PLCs. This attack provides root access to the host attacked by the attacker.
7) *Unauthorized Remote Sniffing (urs):* This attack exploits the vulnerability of embedded Web servers of SICAM RTUs SM-2556 at Port 80/TCP if the server is accessed by an adversary. This attack provides user access to the attacked host.
8) *Unauthorized Remote Execution (urx):* This attack exploits the vulnerability of embedded Web servers of SICAM RTUs SM-2556 at Port 80/TCP if the server is accessed by an adversary. This attack provides root access to the attacked host to the attacker.

The water treatment system can be formally specified as follows.

1) Set of hosts $H = \{0, 1, 2, \ldots, 12\}$; variable $i \in \{0, 1, 2, \ldots, 12\}$ (static parameters).
2) System connectivity, $C \subseteq H \times H$; Boolean; $c_{ij} = 1$ iff host i connected to host j (static parameters).
3) System services S; Boolean; $s_i = 1$ iff service $s \in$ {SCADA server is SINEMA, SIMATIC Wincc runtime flexible, Wincc runtime advanced, Wincc runtime professional, Wincc TIA portal, SIMATIC S7-1500 CPU, SICAM RTUs SM-2556} is running on host i (dynamic variables).
4) System vulnerabilities V; Boolean; $v_i = 1$ iff vulnerability $v \in \{$ *iws*, *lmw*, *rmm*, *inws*, *bac*, *rng*, *osdi*, *uxac*$\}$ exists on host i (static parameters).
5) Attack instances $A_I \subseteq A \times H \times H$; labeled $a_{ij} \equiv$ attack a from source i to target j, $a \in \{rce, usp, ucc, xss, atc, cpm, urs, urx\}$ (static parameters).
6) Trust relation $T \subseteq H \times H$; Boolean; $t_{ij} = 1$ iff i is trusted by j (dynamic variables).
7) Attacker level of privilege L on host i; variable $l_i \in \{none, user, root\}$ (dynamic variables).
8) *Attack Preconditions:* The following conditions must be met for each attack before the attacker can execute it.
   a) $Pre_{(rce_{ij})} \equiv c_{ij} = 1 \wedge (l_i \geq user) \wedge (l_j = none) \wedge iws = 1$.
   b) $Pre_{(usp_{ij})} \equiv c_{ij} = 1 \wedge (l_i \geq user) \wedge (l_j < root) \wedge lmw = 1$.
   c) $Pre_{(ucc_{ij})} \equiv c_{ij} = 1 \wedge (l_i \geq user) \wedge (l_j = none) \wedge rmm = 1$.
   d) $Pre_{(xss_{ij})} \equiv c_{ij} = 1 \wedge (l_i \geq user) \wedge (l_j < root) \wedge inws = 1$.
   e) $Pre_{(atc_{ij})} \equiv c_{ij} = 1 \wedge (l_i \geq user) \wedge (l_j = none) \wedge bac = 1$.
   f) $Pre_{(cpm_{ij})} \equiv c_{ij} = 1 \wedge (l_i \geq user) \wedge (l_j = none) \wedge rng = 1$.
   g) $Pre_{(urs_{ij})} \equiv c_{ij} = 1 \wedge (l_i \geq user) \wedge (l_j = none) \wedge osdi = 1$.
   h) $Pre_{(urx_{ij})} \equiv c_{ij} = 1 \wedge (l_i \geq user) \wedge (l_j = none) \wedge uxac = 1$.
9) *Attack Post-Conditions:* The following represents the attacker's status after each attack.
   a) $post_{(rce_{ij})} \equiv (l_j = user) \wedge (iws = 0)$.
   b) $post_{(usp_{ij})} \equiv (l_j = root) \wedge (lmw = 0)$.
   c) $post_{(ucc_{ij})} \equiv (l_j = user) \wedge (rmm = 0)$.
   d) $post_{(xss_{ij})} \equiv (l_j = root) \wedge (inws = 0)$.
   e) $post_{(atc_{ij})} \equiv (l_j = user) \wedge (bac = 0)$.
   f) $post_{(cpm_{ij})} \equiv (l_j = root) \wedge (rng = 0)$.
   g) $post_{(urs_{ij})} \equiv (l_j = user) \wedge (osdi = 0)$.
   h) $post_{(urx_{ij})} \equiv (l_j = root) \wedge (uxac = 0)$.
10) *Initial States:* $l_0 = root \wedge (l_1 = l_2 = \cdots = l_{11} = none) \wedge (\forall ij \in H \times H : t_{ij} = 0) \wedge (iws = lmw = rmm = inws = bac = rng = osdi = uxac = 1)$. (Initially, the attacker has root privilege on Host-0 and no privilege on other hosts, none of the hosts trust each other, and *iws*, *lmw*, *rmm*, *inws*, *bac*, *rng*, *osdi*, and *uxac* are running on the SCADA system components.)
11) The security property ϕ of interest is violated if the attacker has the root privilege level on Host-4, Host-5, or Host-6. This can then be described by a computational tree logic (CTL) formula

$$\neg\phi \equiv AG((l_4 = root) \vee (l_5 = root) \vee (l_6 = root)).$$

Note that the water treatment SCADA system has 12 devices with 23 connections among them. The devices face seven types of vulnerabilities and eight different types of atomic attacks per host (listed above). An atomic attack may be carried out on a device under certain preconditions to increase the attacker's rights on the linked device. The attack graph then maps the attacker's progressive privilege on all 12 devices as the atomic attacks take place by leveraging the vulnerabilities of the connected devices. These atomic attacks can be nested to form 150 different attack sequences (see Fig. 6) that an attacker may

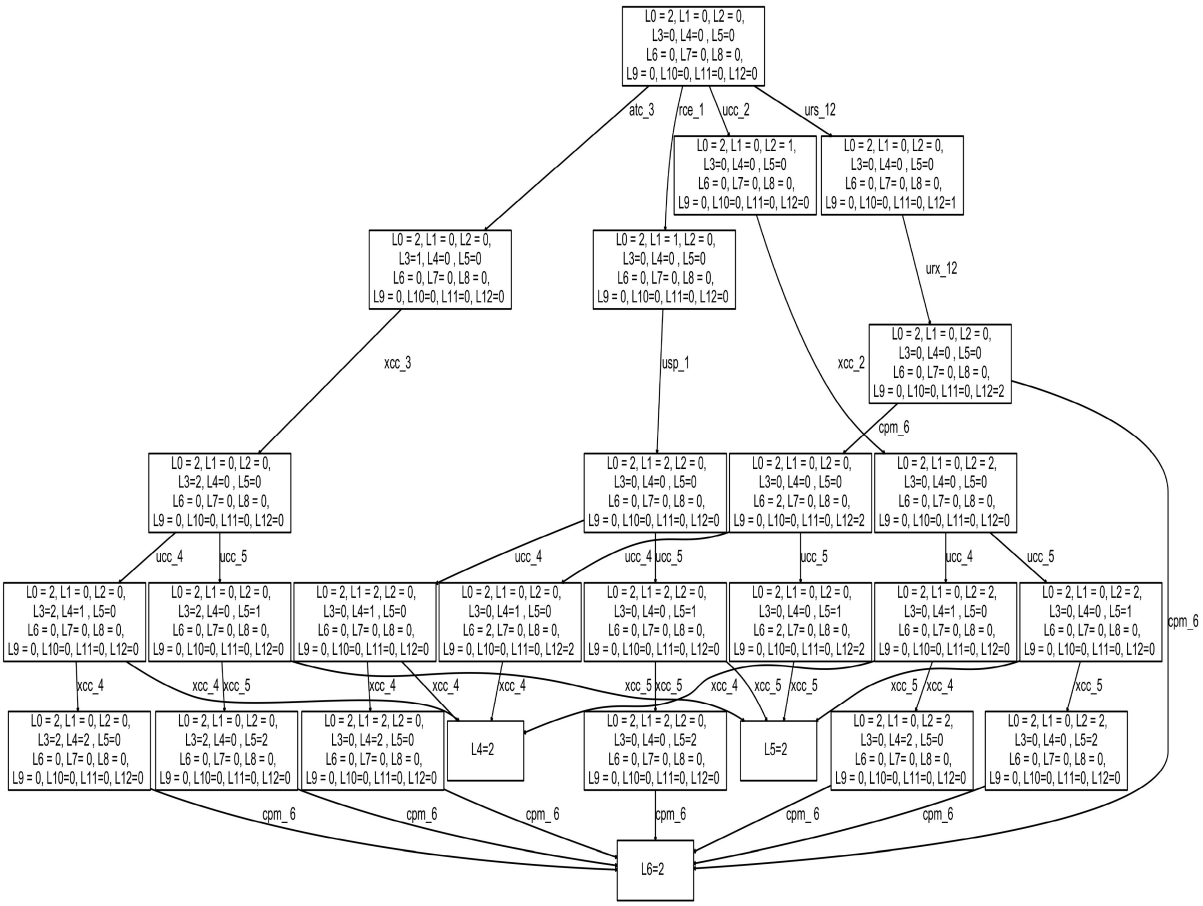

Fig. 6. Generated attack graph for water treatment SCADA CPS.

execute to violate system-level security properties (namely, gain root access on Host-4, Host-5, or Host-6), all of which are automatically computed using our A2G2V tool [19], [20], and also graphically displayed automatically by the tool.

IV. Results and Discussion

Our implementation of the proposed SCCiMLC computed the critical attacks set as $CA = \{rlog\_12\}$ for the 3-host system, and $CA = \{cpm\_6, xcc\_5, xcc\_4\}$ for the water treatment SCADA CPS, meaning that applying security measures to prevent an attacker from exploiting these attacks can guarantee the desired system-level security property of interest. A quick analysis has verified that removing a smaller set of attacks, i.e., a proper subset of CA would not disconnect the attack graph, so the SCCiMLC algorithm found a *minimal* critical attack set for the above-analyzed applications, which we find quite encouraging.

It can be seen that identifying the critical attacks set greatly reduces the effort required to secure the system: In the 3-host system, securing 1 atomic attack ($rlog\_12$) out of $4 \times 2 = 8$ atomic attack actions is required to secure the system; in the water treatment SCADA system, securing three atomic attacks (cpm_6, xcc_5, xcc_4) out of $8 \times 12 = 96$ atomic attack actions is required to secure the system.

To compare the performance of the proposed SCCiMLC, we computed the critical attacks set using an exact algorithm (our generalization of [9]) and a state-of-the-art approximation algorithm of [10] (both described in Section II). The exact algorithm identified the critical attacks set as $CA = \{rlog\_12\}$ for the 3-host system and $CA = \{cpm\_6, ucc\_5, ucc\_4\}$ for the water treatment SCADA CPS. The approximation algorithm identified the critical attacks set as $CA = \{sbo\_01, ftrp\_02, ftrp\_01\}$ for the 3-host system, and $CA = \{atc\_3, rce\_1, xcc\_2, urx\_12\}$ for the water treatment SCADA CPS. It can be seen that our proposed algorithm matches the exact one in terms of the size of the MLCs (although the solutions in the case of water treatment SCADA differ), while it supersedes the approximation algorithm that computed label cuts of larger size in all cases, compared to our SCCiMLC algorithm.

The complexity of our algorithm is linear in the number of attack-graph vertices and edges; the exact algorithm is NP-complete, and the complexity of the state-of-art approximation algorithm is that of a certain linear program (LP)—It is polynomial in the number of attack-graph labels (that equals the number of decision-variables of the LP) and also in the number of paths in the attack-graph which can, in general, be exponential in the number of vertices (that equals the number of constraints of the LP). On a standard computer, with Core i5/2.2 GHz/4-GB RAM running Win 10, our SCCiMLC algorithm took 9 s to compute the *minimal* critical attacks set for the 3-host system attack graph, and 15 s to compute the *minimal* critical attacks set from the water treatment SCADA

TABLE I
SCCiMLC VERSUS EXACT ALGORITHM AND STATE-OF-THE-ART APPROXIMATION ALGORITHM

System $\backslash Algorithm$	Exact Algorithm	Dutta, et al 2016 approx. Algorithm	SCCiMLC
3-Host system	CA={$rlog\_12$}	CA={sbo_01, $ftrp\_02$, $ftrp\_01$}	CA={$rlog\_12$}
Time	10 min	9 sec	9 sec
SCADA system	CA={cpm_6, ucc_5, ucc_4}	CA={atc_3, rce_1, xcc_2, urx_12}	CA={cpm_6, xcc_5, xcc_4}
Time	18 min	16 sec	15 sec

system attack graph. In contrast, the exact algorithm (our generalization of [9]) took 10 min to compute the *minimal* critical attacks set for both the 3-host system and 18 min to compute the *minimal* critical attacks set from the water treatment SCADA system attack graph. The approximation algorithm of [10] took 9 sec to compute the *minimal* critical attacks set for the 3-host system attack graph and 16 sec to compute the *minimal* critical attacks set from the water treatment SCADA system attack graph. These results are summarized in Table I.

Summarily, in the case studies examined, the proposed SCCiMLC achieved the exact algorithm's accuracy at more than 65 times the speedup and offered the same speed as the approximation algorithm but with superior accuracy. In the computer network case, the SCCiMLC algorithm solution matched that computed using the exact algorithm, whereas for the water treatment system, the identified labels differed, yet the size of the label cut reported was still 3 as in the case of the exact algorithm. But the speed of SCCiMLC is more than 65 times faster. Conversely, the size of the label cut computed by the approximation algorithm in [10] was larger than that computed by the SCCiMLC algorithm, although the speed was comparable.

V. CONCLUSION AND FUTURE WORK

An attack graph is a model that represents the ways in which an attacker can compromise a system. A critical attack set is a set of attacks that, if prevented, would make the system secure. However, finding the optimal critical attack set is generally a difficult problem known as the MLC problem, which is NP-complete. To address this challenge, we propose a novel approach called SCCs-induced min label cut (SCCiMLC). Our approach has linear complexity, making it *scalable* for practical systems. By using an abstracted attack graph over its SCCs, we are able to approximate the solution to the MLC problem with a linear complexity algorithm.

To demonstrate the effectiveness of our approach, we extended our tool-chain ANDVI [21] and A2G2V [21] (see Fig. 1) to analyze an automatically generated attack graph. This graph was constructed from passive observations of network packets to identify devices and their connectivity, mapping out device vulnerabilities utilizing existing vulnerability databases, and generating a system-level attack graph. We then used our SCCiMLC algorithm to obtain the critical attack set. To test our approach, we applied it to a realistic computer network and a real-world water treatment SCADA system testbed from the iTrust Lab. The results demonstrated the validity of our proposed approach, with only a fraction of the possible attacks forming a critical set (e.g., in the case of the water treatment SCADA, only 3 out of 96 attacks were critical). This implies that identifying a critical attack set can be hugely beneficial in securing complex networked systems, especially when faced with limited resources, such as budget and downtime for maintenance.

To evaluate the performance of our SCCiMLC algorithm, we compared it to both an exact algorithm and a state-of-the-art approximation algorithm from the literature. We implemented all three algorithms and compared their cut sizes and computation times. The SCCiMLC algorithm was able to compute a cut size that matched that of the exact algorithm. However, its computation time was comparable to the approximation algorithm, making it much faster than the exact algorithm. In contrast, the approximation algorithm always produced a larger cut size than the SCCiMLC (and the exact algorithm). In the examples we considered, we observed a speedup of greater than 65 times over the exact algorithm without any loss of accuracy. This demonstrates that our SCCiMLC algorithm is both efficient and accurate, providing a practical solution for identifying critical attack sets in complex networked systems. Overall, our extended toolchain and SCCiMLC algorithm provide a practical and effective way to secure complex networked systems. By automating the identification of critical attack sets, our approach can help prioritize limited resources for maximum security benefit.

Identifying the critical attacks set in SCADA/ICS/ Computer-networks systems is an essential first step toward improving overall cybersecurity by helping system administrators optimally allocate their resources for enhancing security defenses. A final step would be to integrate the proposed approach with a run-time security defense-patch implementation tool to achieve an optimized resource-based defense implementation for achieving security against potential cyber-attacks.

ACKNOWLEDGMENT

The authors would like to express their gratitude to the iTrust team for sharing the SCADA data for the Water Treatment System that we used to validate our proposed strategy.

REFERENCES

[1] C.-W. Ten, G. Manimaran, and C.-C. Liu, "Cybersecurity for critical infrastructures: Attack and defense modeling," *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 40, no. 4, pp. 853–865, Jul. 2010.

[2] A. F. AlEroud and G. Karabatis, "Queryable semantics to detect cyber-attacks: A flow-based detection approach," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 48, no. 2, pp. 207–223, Feb. 2018.

[3] "Year in review 2016 incident response pie charts." Kaspersky. 2017. Accessed: Oct. 17, 2022. [Online]. Available: https://us-cert.cisa.gov/sites/default/files/Annual_Reports/Year_in_Review_FY2016_IR_Pie_Chart_S508C.pdf

[4] "More than 40% of ICS computers were attacked in H1 2018." ICS-CERT. 2018. Accessed: Oct. 17, 2022. [Online]. Available: https://www.kaspersky.com/about/press-releases/2018_ics-computers-attacked-in-h1

[5] O. Sheyner and J. Wing, "Tools for generating and analyzing attack graphs," in *Proc. Int. Symp. Formal Methods Compon. Objects*, 2003, pp. 344–371.
[6] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. M. Wing, "Automated generation and analysis of attack graphs," in *Proc. IEEE Symp. Security Privacy*, 2002, pp. 273–284.
[7] S. Jha, O. Sheyner, and J. Wing, "Two formal analyses of attack graphs," in *Proc. 15th IEEE Comput. Security Found. Workshop (CSFW)*, 2002, pp. 49–63.
[8] "Secure water treatment tesetbed." iTrust. 2019. Accessed: Oct. 17, 2022. [Online]. Available: https://itrust.sutd.edu.sg/testbeds/secure-water-treatment-swat/
[9] P. Zhang and B. Fu, "The label cut problem with respect to path length and label frequency," *Theor. Comput. Sci.*, vol. 648, pp. 72–83, Oct. 2016.
[10] T. Dutta, L. S. Heath, V. S. A. Kumar, and M. V. Marathe, "Labeled cuts in graphs," *Theor. Comput. Sci.*, vol. 648, pp. 34–39, Oct. 2016.
[11] P. Zhang, J.-Y. Cai, L.-Q. Tang, and W.-B. Zhao, "Approximation and hardness results for label cut and related problems," *J. Comb. Optim.*, vol. 21, no. 2, pp. 192–208, 2011.
[12] S. Noel, S. Jajodia, B. O'Berry, and M. Jacobs, "Efficient minimum-cost network hardening via exploit dependency graphs," in *Proc. 19th Annu. Comput. Security Appl. Conf.*, 2003, pp. 86–95.
[13] L. Wang, S. Noel, and S. Jajodia, "Minimum-cost network hardening using attack graphs," *Comput. Commun.*, vol. 29, no. 18, pp. 3812–3824, 2006.
[14] C.-S. Cho, W.-H. Chung, and S.-Y. Kuo, "Cyberphysical security and dependability analysis of digital control systems in nuclear power plants," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 46, no. 3, pp. 356–369, Mar. 2016.
[15] R. E. Sawilla and X. Ou, "Identifying critical attack assets in dependency attack graphs," in *Proc. Eur. Symp. Res. Comput. Security*, 2008, pp. 18–34.
[16] R. Sawilla and C. Burrell, *Course of Action Recommendations for Practical Network Defence*, Defence R&D Canada, Ottawa, ON, Canada, 2009.
[17] M. Alhomidi and M. Reed, "Finding the minimum cut set in attack graphs using genetic algorithms," in *Proc. Int. Conf. Comput. Appl. Technol. (ICCAT)*, 2013, pp. 1–6.
[18] R. Hassin, J. Monnot, and D. Segev, "Approximation algorithms and hardness results for labeled connectivity problems," *J. Comb. Optim.*, vol. 14, no. 4, pp. 437–453, 2007.
[19] A. T. Al Ghazo, M. Ibrahim, H. Ren, and R. Kumar, "A2G2V: Automated attack graph generator and visualizer," in *Proc. 1st ACM MobiHoc Workshop Mobile IoT Sens. Security Privacy*, 2018, p. 3.
[20] A. T. Al Ghazo, M. Ibrahim, H. Ren, and R. Kumar, "A2G2V: Automatic attack graph generation and Visualization and its applications to computer and SCADA networks," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 50, no. 10, pp. 3488–3498, Oct. 2020.
[21] A. T. A. Ghazo and R. Kumar, "ICS/SCADA device recognition: A hybrid communication-patterns and passive-fingerprinting approach," in *Proc. IFIP/IEEE Symp. Integr. Netw. Serv. Manag. (IM)*, 2019, pp. 19–24.
[22] R. Tarjan, "Depth-first search and linear graph algorithms," *SIAM J. Comput.*, vol. 1, no. 2, pp. 146–160, 1972.
[23] F. Stallmann. "Masters-thesis-ellipsoid." 2015. Accessed: Oct. 17, 2022. [Online]. Available: https://github.com/mrflory/masters-thesis-ellipsoid
[24] "Open source AADL tool environment for the SAE architecture analysis and design language (AADL)." Carnegie-Mellon-University. 2016. Accessed: Oct. 17, 2022. [Online]. Available: http://osate.org/about-osate.html
[25] (Rockwell-Collins and Uof-Minnesota, Pittsburgh, PA, USA). *The Assume Guarantee Reasoning Environment*. (2016). Accessed: Oct. 17, 2022. [Online]. Available: http://loonwerks.com/tools/agree.html
[26] "Siemens common vulnerabilities." CVE. 2018. Accessed: Oct. 17, 2022. [Online]. Available: https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=siemens
[27] "ICS-CERT." Department of Homeland Security. 2018. Accessed: Oct. 17, 2022. [Online]. Available: https://search.usa.gov/search?utf%%%93&affiliate=us-cert-ics&sort_by=&query=siemens

Alaa T. Al Ghazo (Member, IEEE) received the B.S. degree from Yarmouk University, Irbid, Jordan, in 2009, the M.S. degree from Edinburgh Napier University, Edinburgh, U.K., in 2013, and the Ph.D. degree from Iowa State University, Ames, IA, USA, in 2020.

He is an Assistant Professor with the Department of Mechatronics Engineering, The Hashemite University, Zarqa, Jordan. He was with LG Electronics Jordan, Amman, Jordan, from 2009 to 2010, with Zoofi Tech, Riyadh, Saudi Arabia, from 2010 to 2012, and with Reborn Industries, Jordan, from 2014 to 2015. He was an Assistant Professor with The University of Hartford, West Hartford, CT, USA, from 2019 to 2021. His research interests include cyber–physical systems and cybersecurity, automation and control, artificial intelligence, and machine learning.

Ratnesh Kumar (Fellow, IEEE) received the B.Tech. degree in electrical engineering from IIT Kanpur, Kanpur, India, in 1987, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Texas at Austin, Austin, TX, USA, in 1989 and 1991, respectively.

He is a Palmer Professor with the Department of Electrical and Computer Engineering, Iowa State University, Austin, TX, USA, where he directs the Embedded Software, Sensors, Networks, Cyberphysical, and Energy Lab. Previously, he held faculty position with the University of Kentucky, Lexington, KY, USA, and various visiting positions with The University of Maryland, College Park, MD, USA; the Applied Research Laboratory, the Pennsylvania State University, State College, PA, USA; the NASA Ames, the Idaho National Laboratory, Falls, ID, USA; the United Technologies Research Center, Berkeley, CA, USA; and the Air Force Research Laboratory, Wright-Patterson Air Force Base, OH, USA.

Prof. Kumar is a recipient of the D. R. Boylan Eminent Faculty Award for Research and Award for Outstanding Achievement in Research from Iowa State University, and also the Distinguished Alumni Award from IIT Kanpur. He received Gold Medals for the Best EE Undergrad, the Best All Rounder, the Best EE Project from IIT Kanpur, and the Best Dissertation Award from UT Austin, the Best Paper Award from the IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING, and has been the Keynote Speaker and the Paper Award Recipient from multiple conferences. He is or has been an Editor of several journals (including of IEEE, SIAM, ACM, Springer, IET, and MDPI). He is a Fellow of AAAS and was a Distinguished Lecturer of the IEEE Control Systems Society.