Measuring and Improving the Effectiveness of Defense-in-Depth Postures

Peter Mell
National Institute of Standards and Technology
100 Bureau Drive, Stop 8930
Gaithersburg, MD 20871
01-301-540-0061
peter.mell@nist.gov

James Shook
National Institute of Standards and Technology
100 Bureau Drive, Stop 8930
Gaithersburg, MD 20871
01-301-975-5264
james.shook@nist.gov

Richard Harang
Army Research Laboratory
2800 Powder Mill Road
Adelphi, MD 20783
01-301-394-2444
richard.e.harang.civ@mail.mil

ABSTRACT
Defense-in-depth is an important security architecture principle that has significant application to industrial control systems (ICS), cloud services, storehouses of sensitive data, and many other areas. We claim that an ideal defense-in-depth posture is 'deep', containing many layers of security, and 'narrow', the number of node independent attack paths is minimized. Unfortunately, accurately calculating both depth and width is difficult using standard graph algorithms because of a lack of independence between multiple vulnerability instances (i.e., if an attacker can penetrate a particular vulnerability on one host then they can likely penetrate the same vulnerability on another host). To address this, we represent known weaknesses and vulnerabilities as a type of colored attack graph. We measure depth and width through solving the shortest color path and minimum color cut problems. We prove both of these to be NP-Hard and thus for our solution we provide a suite of greedy heuristics. We then empirically apply our approach to large randomly generated networks as well as to ICS networks generated from a published ICS attack template. Lastly, we discuss how to use these results to help guide improvements to defense-in-depth postures.

Categories and Subject Descriptors
C.2.1 [**Computer-Communication Networks**]: General – *security and protection.*

General Terms
Algorithms, Measurement, Design, Experimentation, Security.

Keywords
attack graph, defense in depth, measurement, security.

1. INTRODUCTION
For network security, defense-in-depth (DID) is the security architectural practice of putting multiple defensive barriers between potential attackers and their desired targets [1]. These barriers often take the form of security devices or application portals/gateways; this then forms a layered security architecture. In this work, we measure the effectiveness of DID postures through evaluating residual weaknesses that, if exploited, can provide avenues for an attacker to breach the layers of security. We model the available avenues of attack both at the level of individual software vulnerabilities (e.g., Common Vulnerabilities and Exposures (CVE) names [2]) and at higher levels of abstraction (e.g., general threats that apply to classes of devices).

DID is a concept that is generally applicable in a wide variety of security domains. For example, DID has important application in industrial control systems (ICS) where computer controlled physical components are often separated from the Internet through a series of high level controllers and isolation devices [3]. It also has important application in cloud computing application architectures where, for example, a software-as-a-service (SaaS) application may have a series of layers that separate users from the raw application data [4]. It is also often important for storehouses of sensitive data where users must be limited in their access to the data (such as in federal government systems [5]).

We claim that an ideal DID architecture has three important features. It must be 'deep', meaning that there are many independent layers of security (we define independence later). It must be 'narrow', meaning that the number of node independent attack paths is minimized. In this work we focus on the equivalent but less intuitive idea of minimizing the smallest set of independent nodes whose removal could cut all attack paths. And it must be 'strong', meaning that each layer must provide the greatest possible deterrent to an attacker. In this work, we focus on measuring only depth and width because the strength of a layer is difficult to accurately assign. While strength is important to consider in one's layered defense, we focus here only on metrics that are quantitative and repeatable.

The basic definition of depth could be solved easily and quickly. However, the version of depth we need is computationally expensive to calculate. To evaluate it, we represent the DID security architecture as a type of attack graph. Each path in the graph represents a possible avenue of attack and each node represents a specific vulnerability or weakness on a host. The basic definition of depth asks for the shortest path from the initial node (point of expected hostile presence, often the Internet) to the closest target node. The target nodes vary by domain but represent the crown jewels of the network (e.g., the physical devices in an ICS or the main data store for a SaaS).

Complicating the basic definition of depth is that if an attacker can penetrate a particular vulnerability on one host then they can likely penetrate the same vulnerability on another host (given the same provisions for logical access). To account for this, we color each node corresponding to the vulnerability it represents and we then consider differently colored nodes to be independent. Thus,

calculating depth is no longer a shortest path calculation on a graph (e.g., Dikjstra or Floyd's [6]), but instead a calculation of the fewest number of colors needed to enable a path from the source to one of the targets. We show that this small change to calculating shortest color paths (SCPs) converts the problem from a well-studied polynomial realm to being NP-Hard.

With respect to width, we use the same colored attack graph representation and calculate the smallest set of colors such that at least one color in the set will occur on each possible attack path. This is equivalent to finding the smallest set of colors where removal of all nodes of the chosen colors will break all attack paths. Without consideration for color, the attack graph could be transformed into a flow graph using published transformations and the minimum cut calculated in polynomial time [6]. However, we show that incorporating the colors and calculating minimum color cuts (MCCs) again converts the problem to being NP-Hard.

Since SCP and MCC are NP-Hard we developed a suite of approximation algorithms for them both. For some, we use variants on the standard polynomial time shortest path and minimum cut algorithms to attempt to greedily use many nodes of the same color (thereby minimizing the overall number of colors used). We also use genetic algorithms and another local minimum search approach. Lastly, we developed algorithms that provide exact answers, with limited scalability. These can be used on small problems and to test the accuracy of the approximation heuristics on such problems.

To empirically test our algorithms, we primarily use a set of randomly generated graphs to test effectiveness and scalability. We used another set of graphs to test applicability to an important DID domain area: ICS. For the effectiveness and scalability tests, we use layered colored attack graphs with 1000 nodes 10 layers. For the ICS domain testing, we generate ICS colored attack graphs by leveraging a published attack template of an ICS secured according to best security practices [7]. This attack template shows the security layers and attack paths which enable us to generate corresponding colored attack graphs.

From the random graph experiments, we compare the effectiveness and scalability of different heuristics. For the SCP, we find that our greedy breadth first search (BFS) algorithm provides the closest overall approximation to the actual SCPs. However, for less than 50 distinct vulnerabilities in the network, our exact algorithm can provide the true answer. For reduced runtime in this range, our genetic algorithm can be used and it empirically provides almost the exact answer. Our greedy BFS algorithm executes with linear time complexity and thus scales to massive networks. For the MCC, our greedy color-aware minimum node cut ensemble approach was the most effective overall. For greater accuracy with less than 100 vulnerabilities in the network, our genetic algorithm provides better approximations. For less than 30 vulnerabilities, our exact algorithm can provide the true answer.

From the ICS graphs generated from the ICS attack template, we performed a proof of concept of the approach on more realistic graphs. We determined that real ICS networks are layered (the attack template itself had 8 layers) and that our algorithms were able to determine the SCP and MCC which varied depending upon how sets of specific hosts were instantiated conformant with the attack template.

Our main conclusion is that he SCP and MCC metrics represent a novel way in which to measure characteristics of a DID posture. They are both quantitative and repeatable. While exponential to calculate exactly, we provide several scalable approximation approaches. While not intended to be a complete measure of security, SCP and MCC can contribute to an overall security metrics suite. In particular, they provide measurement of two particular features of the important principle of DID.

Our contributions in this paper include the following:

1. a novel and general method to express the DID of a network using layered colored attack graphs,
2. the novel metrics of SCP and MCC to measure DID effectiveness,
3. a proof that exact SCP and MCC measurements are NP-Hard,
4. and computational efficient and effective approximation algorithms for determining SCP and MCC.

The rest of the paper is organized as follows. Section 2 describes our colored attack graph representation for the DID architectures. Section 3 presents our security metrics of depth and width while section 4 describes our related algorithms. Section 5 discussed our test data: random layered security graphs and the ICS graphs. Section 6 provides the results of our empirical trials. Section 7 discusses how to use this data to optimize network security. Section 8 reviews related network security metrics and section 9 concludes.

2. REPRESENTING DID ARCHITECTURES AS COLORED ATTACK GRAPHS

To evaluate the DID posture of a network, we construct colored attack graphs that show available avenues of attack. These avenues can be general threats to specific types of devices, actual known vulnerabilities (e.g., CVEs), or a combination of both. In this section, we provide a novel attack graph construction using colored nodes that will enable us to evaluate DID characteristics.

Expected sources of attack are represented by a set of nodes, S, in our attack graph, G. A special 'super source' node, s, is added to G that has a directed edge to each node in S. The presence of s is for algorithmic convenience in reaching all expected sources of attack.

The high value targets in the network are represented by a set of nodes, T, in G. A special 'super sink' node, t, is added to G by adding a directed edge from each node in T to node t. As with s, t is added for algorithm convenience.

We then add a node to G for each vulnerability instance (or more generalized threat) in the network that could enable the creation of an attack path. Thus, each node represents a distinct vulnerability at a particular location (usually on some network host). For example, a node a may represent the vulnerability host pairing $[v_1,h_1]$ where v_1 is a vulnerability that exists on host h_1.

Edges are now added to G using the following formula (note, no additional edges are added to or from s and t). For each ordered pair of nodes, $a=[v_i,h_j]$ and $b=[v_k,h_l]$, a directed edge is created from a to b iff v_i on h_j provides an attacker sufficient privileges to exploit v_k on h_l and h_l is logically accessible from h_j. This accessibility will usually be network connectivity but may generalize to other forms of access (e.g., insertion of physical media in a device). For local attacks internal to a host, h_i and h_j may represent the same host.

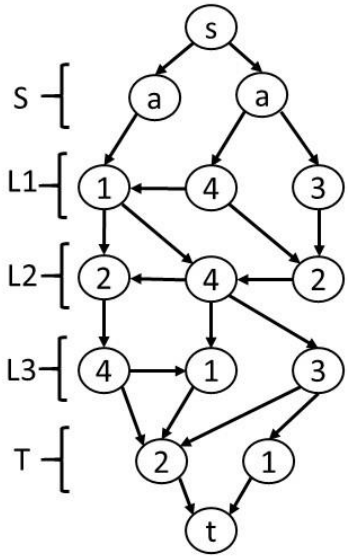

Figure 1. Example Colored Attack Graph Representing a Layered Security Architecture

An example attack graph is shown in Figure 1. At the top is the super-source and at the bottom is the super-sink, labelled with 's' and 't' respectively. On the left side one can see the groupings of nodes for S, T, and the individual security layers (denoted L1-L3) in the DID architecture. The layers are automatically generated by calculating the shortest path distance (e.g., Dikstra distance) minus 1 of each node from *s* and using that as the layer number. Thus, layer *i* contains all the nodes that are distance *i* from the set of attacking nodes.

The numbers within each node represent the color of the node. Note that the nodes in layer S do not have colors as they are the sources of attack and so they are labelled with an 'a'. The hostnames associated with each node are not shown. Note that multiple nodes may represent distinct vulnerabilities within the same host or between different hosts. Also note that actual layered security architectures can have edges traversing layers in both directions; we exclude such edges from our example for simplicity. However, edges may never traverse multiple layers as that would contradict our method of layer construction (such an edge would indicate a vulnerability that was put at the incorrect layer number based on its distance from *s*).

Our layered and colored attack graph representation where nodes represent host/vulnerability pairings is thus a novel, general, and flexible representation that can be applied to most DID architectures. One limitation is that our approach does not cover cases where an attacker must have privileges on two different hosts in order to execute an attack or where a combination of two or more exploited vulnerabilities is necessary to compromise another vulnerability. These are not the normal cases for vulnerabilities in public repositories (e.g., the National Vulnerability Database [8]).

While we have emphasized the novelty of our attack graph approach (with the colorings and layering), note that we leverage the node and edge semantics of the attack graph 'vulnerability oriented' model presented [9].

3. SECURITY METRICS

We can now use our colored attack graph representation to measure the DID posture of a network. We focus on measurements of depth and width. Depth measures the number of independent layers of security and width measures the smallest set of independent nodes whose removal could cut all attack paths. An increasing depth forces an attacker to penetrate more distinct vulnerabilities and a decreased width gives an attacker less flexibility in choosing which vulnerabilities to exploit. Thus, an ideal DID has both a large depth and small width.

Note that in this work, we do not combine these two measurements into a single DID strength score because such efforts tend to be ad hoc, even if operationally useful.

3.1 Depth – Evaluating Shortest Color Paths

To calculate depth, we must find a path from s to t that uses that fewest number of colors. We count only distinct colors instead of nodes because we assume that if an attacker can exploit a vulnerability on a host *a* then the attacker can exploit the same vulnerability on some host *b* (provided logical connectivity is available). The requirement for logical connectivity is easy to take into account because it is modelled by the edges in G. We refer to the entire operation of measuring the depth as finding the shortest color path (SCP).

Using the same example attack graph from Figure 1, Figure 2 shows the SCP by bolding, dotting, and making red the applicable directed edges.

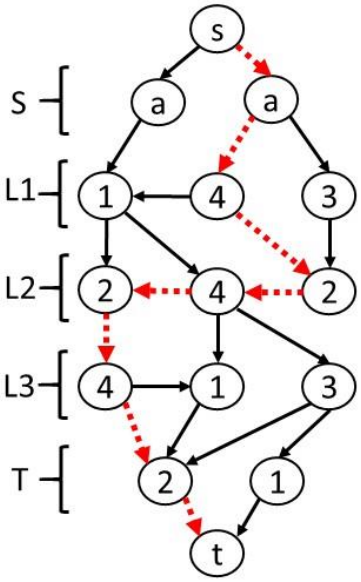

Figure 2. Example Attack Graph with the Shortest Color Path Highlighted

Note how the SCP is not necessarily the shortest path (as defined by number of nodes used). Here, the SCP is 2 (using colors 2 and 4) even though it traverses 8 nodes total. Remember that the nodes in S do not have colors since they are sources of attack. For comparison, the shortest path from s to t as traditionally defined is 6.

3.2 Width – Evaluating Minimum Color Cuts

To calculate width, we must find the smallest set of colors such that at least one color in the set will occur on every possible attack path. Another way to view this is to find a cut set of nodes with the fewest colors that eliminates all paths from s to t. We thus refer measuring the width as finding the minimum color cut (MCC). Note that the nodes in S are not candidates for the MCC as they represent the

attackers (they are not vulnerability options for the attackers to exploit). However, the nodes in T are eligible as they represent exploitation opportunities for the attackers on critical targets. Nodes s and t are not eligible as they were added for algorithmic convenience.

Using the same example attack graph from Figure 1, Figure 3 shows the MCC by bolding, dotting, and making red the applicable nodes.

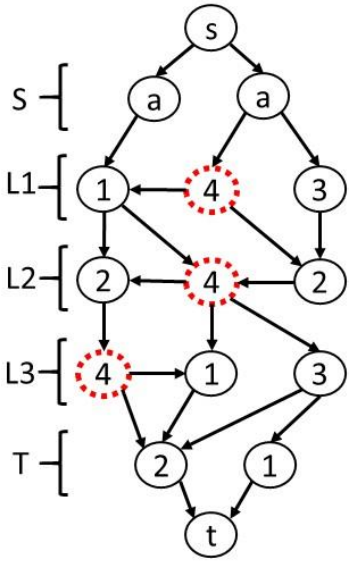

Figure 3. Example Attack Graph with the Minimum Color Cut Highlighted

Removal of the MCC nodes will disconnect s from t. Note how the MCC consists of all nodes of any chosen color (not just particular nodes that disconnect the graph). Thus, calculating the MCC is different from calculating the minimum cut (which counts the number of nodes used irrespective of color). The minimum cut for this graph is 2 while the MCC is 1 (even though it uses 3 nodes).

4. ALGORITHMS FOR DEPTH AND WIDTH

In this section, we present a variety of heuristics to calculate DID depth and width as well as high complexity exact approaches. Our best heuristic for SCP is a greedy shortest color paths algorithm. Our best heuristic for MCC is a color aware node cut algorithm. We also test a local minimization approach and a genetic algorithm for both SCP and MCC. The heuristics and genetic algorithms enable our measurements to scale to large networks while the exact approaches can be used for small networks.

4.1 Greedy Approaches

We used two greedy heuristics for our DID measurements. For depth, we use a greedy color-aware modified breath first search approach. For width, we make color aware use of standard minimum node cut algorithms.

4.1.1 Shortest Path Based Depth

To calculate depth we execute a breadth first search (BFS) algorithm [6] starting at s and terminating upon arrival at t. As typical with BFS algorithms, whenever a node is reached, it is labelled with its predecessor so that the shortest path can be enumerated upon reaching t by following the path in reverse. However, our BFS is modified to take into account color. Not only is each node labelled with its predecessor, but it is labelled with all colors used to get to that node.

Whenever a node is visited, the algorithm starts a secondary BFS from that node that is limited to visiting nodes colored with the previously used colors (those colors listed at the start node of the secondary BFS). This secondary BFS performs a greedy expansion but is not allowed to visit any nodes previously visited by any BFS instance (primary or secondary).

The secondary BFS capability enables a particular path being explored by the primary BFS to greedily include as many nodes as possible that use colors already used by the node currently being processed by the primary BFS.

Standard BFS is a linear time algorithm as is our variant (since each node will be processed by one and only one BFS instance) and so the computational complexity is $O(n+m)$ where n is the number of vertices and m is the number of edges.

4.1.2 Minimum Node Cut Based Width

To calculate width, we iteratively calculate the minimum node cut that separates the nodes in set S from t in the colored attack graph being evaluated. After each iteration, we remove all nodes of some chosen color. We repeat this until there does not exist a path from nodes in S to t.

There are several ways in which we choose the color to remove after each iteration:

1. Choose the color that is most frequent within the discovered node cut.
2. Restricted to the colors found in the standard node cut, choose the color that occurs most frequently in the entire attack graph.
3. Use method 1 above unless there is no color with an occurrence within the cut of greater than 1. In that case, use method 2.

Finding a minimum node cut (using the Edmonds Karp method) is $O(nm^2)$. We iteratively run this algorithm with the loop limited by the number of colors, c. Thus, our algorithms runs in $O(cnm^2)$.

We can assist the minimum node cut algorithm in identifying cuts with important colors by pre-processing the graph. Our pre-processor looks for nodes of the same color (e.g., a and b) that have edges to a common node (e.g., c) and adds a new node (e.g., d) with the color of a and b that is put in front of c. More specifically, the preprocessor creates a new node d, then adds edges a->d, b->d, and d->c, and then deletes edges a->c and b->c. This adds a single node that the node cut algorithm can find that allows both a and b to be cut from c. It makes the node cut algorithm somewhat color aware without modifying the algorithms itself. This preprocessor take $O(m+n)$ time.

4.1.3 Minimization Based Depth and Width

We also greedily calculate both depth and width through calculating which colors are necessary, taking us to a minimal solution. To do this, we rank the colors by their popularity (number of nodes with that color) and place them in a list. Then we iteratively remove the most popular color. We also create an, initially empty, set of 'necessary' colors which will be the output of the algorithm.

For each removed color, we construct a candidate color set of the remaining unprocessed colors combined with the necessary set. If the candidate set does not enable an (s,t) path (for SCP) or does enable an (s,t) paths (for MCC) then we know that the removed color is necessary and we add it to the necessary set. Processed

colors not added to the necessary set are dropped because they are not required for a color path (for SCP) or for a color cut (for MCC).

We can also execute this same algorithm for both SCP and MCC by providing an initial set of colors (colors not in the initial set are not used). This enables one to refine an existing SCP or MCC answer to find a minimal solution. Since the minimization routine executes quickly, we apply this optimization to the output of all our other approximation algorithms.

4.2 Genetic Algorithm Approaches

Our genetic algorithm (GA) approach represents all available colors as a bit string. It uses a fitness function that is the inverse of the sum of the number of bits set to 1 (i.e., number of colors used) multiplied by a binary value indicating whether or not there exists an s to t path using the allowed colors. We use a population size of 1000 and the initial population contains bit strings allowing all colors, all but one color, and bit strings where each color is turned on with a probability of .8. We thus bias our initial population towards using an excessive number of colors and allow the algorithms to trim out the unnecessary colors. To generate new populations, we leverage only bit strings from the previous generation that had a non-zero fitness function. We keep unchanged the top 10% scorers of the previous generation. The next 20% of the top scorers from the previous generation are kept with one of their colors randomly removed. Most of the remaining population for the new generation is created by using a random mask to crossover a random choice of the top 30% scoring bit strings with a random choice from all bit strings from the previous generation with a non-zero fitness. This process may generate candidate sets with duplicates (which we remove). We then ensure a new population size of 1000 by creating random bit strings (we always generate at least 10 random bit strings for each new population). To generate a set of random bit strings, we pick a threshold value for the probability of a bit being 1 (chosen uniformly from 0 to 1) and then use that value to generate all bit strings for that population.

Note that this algorithm takes a very pessimistic view of the usefulness of the graph since it only uses the graph to verify the correctness of proposed solutions. To verify a SCP solution, our linear time greedy approach is used. Because of this we can use the same algorithm to calculate both SCP and MCC with only a slight modification to the fitness function. For SCP, we force to 0 the fitness if an s to t path exists and for MCC we force it to 0 if an s to t path does not exist.

4.3 Exact Approaches

Since both SCP and MCC are minimization problems we can use a simple exhaustive search of the subsets of colors to find an exact solution. We use our previous approximation algorithms to determine an upper bound on the number of colors needed for SCP and MCC and then we iteratively try different sized sets of colors. At each chosen size, we exhaustively try all color combinations. We stop processing a particular sized color set if we find a set of colors that enable an (s,t) path (for SCP) or we find a set of colors that enable an (s,t) cut (for MCC).

We implemented three approaches. The first is to start with the upper bound and decrement the color set size to be searched by one in each iteration. The second is to use the upper bound to perform a binary search on the remaining possible color set sizes (reducing the possible set of sizes in half for each iteration). The third is to start at color set sizes of 1 and work upwards until we find the first working solution. The first approach empirically worked the fastest and so we use it exclusively in reporting results.

Note that these algorithms depend on the number of colors. If there are k colors, then the algorithms have to check up to 2^k subsets and then traverse up to n vertices for each subset. Thus, they are fixed parameter tractable problems. If the algorithms can be fed a good upper bound solution (e.g., from our approximation algorithms) then the exact algorithm can be sped up considerably which explains why our first approach outperformed the binary search approach in all of our experiments.

4.4 Complexity Evaluation

In this section we provide a proof sketch showing that both the SCP and MCC problems are NP-Hard (by using set cover reductions). Given that no polynomial algorithms exist to provide exact solutions for the set of NP-Hard problems, this explains our motivation to search for effective approximation algorithms.

We show that any instance of set cover problem can be reduced to an instance of an MCC problem in polynomial time. We let $U = \{u_1,\ldots,u_m\}$ and $S = \{S_1,\ldots,S_n\}$ be an arbitrary instance of a set cover problem where U is the set to be 'covered' and S is the set of sets that can be chosen to cover U. Assume that each S_i has cost 1. For an example problem for which we will show reductions, let U={a, b, c} and S={ac, ab, b}.

Given any set cover instance, we can create an auxiliary digraph. Let s and t be vertices. For each u_i, create a vertex disjoint (s,t)-path that has a vertex colored S_k for each S_k that contains u_i. Note that vertices s and t are not colored. This provides an auxiliary graph with m vertex disjoint (s,t)-paths. A minimum MCC calculation on the auxiliary graph will then yield a set of colors that correspond to the minimum set within S that covers U. Figure 4 shows the MCC auxiliary digraph for our example problem.

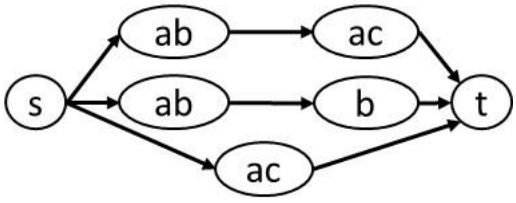

Figure 4. Auxiliary Digraph for Minimum Color Cut

The reduction of set cover to an SCP problem is as follows. Create nodes s and t. For each u_i, create a graph layer that contains a S_k colored node for each S_k that contains u_i. Including a layer for both s and t, this gives us m+2 layers (note that m=|U|) with each layer being an independent set. Create a complete set of edges from node s to the nodes in layer 1. For each layer $i< m$, create a complete set of edges from the nodes in layer i to layer i+1. Lastly, create a complete set of edges from the nodes in layer m to node t. The result is an auxiliary graph that is a layered graph as described in section 2. A minimum SCP calculation on the auxiliary graph will then yield a set of colors that correspond to the minimum set within S that covers U. Figure 5 shows the SCP auxiliary digraph for our example problem.

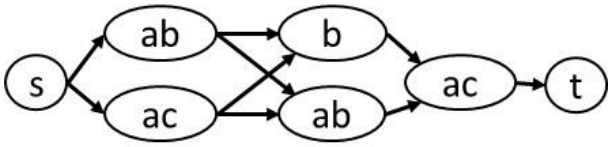

Figure 5. Auxiliary Digraph for Shortest Color Path

Thus, any arbitrary set cover problem can be reduced into both an SCP problem and an MCC problem such that a solution to either

problem yields a solution to the set cover problem. These are sufficient transformations to prove that the SCP and MCC problems are NP-Hard (the proofs themselves are not included due to space constraints).

5. DATA SETS

For our empirical analysis, we use two different data sets. The first is a random layered attack graph used to assess the effectiveness and scalability of our algorithms. The second is a layered attack graph based on an attack template for secured industrial control systems.

5.1 Random Layered Graphs

Given some enterprise network and starting from a set of possible attack sources (e.g., the Internet, employee desktops, or hosts that provide a SaaS interface), one can perform a breadth first search (BFS) of all possible attack paths from the attack sources to all other nodes in a network. In each iteration of the BFS, we consider the nodes reached to be at a new layer (representing increasing distance from the attack sources).

For our experiments we use this concept of layering to generate random attack graphs. We construct a graph by choosing the number of layers, the number of nodes per layer, the number of attack sources, and the number of high value targets. We add in the supersource 's' and supersink 't' as described in section 3. For edges, we decide on the probabilities for edges existing between layers of varying distances (including edges within a layer). The number of layers chosen will effect (but not decide) the depth of the graph and the number of nodes per layer will effect (but not decide) the width of the graph. By varying the number of layers and nodes per layer, we can scale the generated graphs.

5.2 Industrial Control System Data Set

An attack template for a hypothetical ICS that is secured with 'best practice' methods was provided by [7]. The template provides nodes for different actor types within the ICS (e.g., controller types, server types, and human entities). The edges representing possible attacks that can be launched from the vantage point of one actor in order to compromise another actor.

We use this attack template to generate randomized ICS attack graphs (colored and layered as described in section 3). The randomization comes in as we instantiate a particular actor type into actual instances of the actor. To perform this 'node explosion', we copy a single actor type node many times (retaining the same neighbors for each copy) in order to represent an instantiation of the actor type into actual actors. Each node copy is assigned a random color (unique from any color assigned to other actor types). We then take all node copies and connect them with edges to form a clique. Lastly we randomly remove edges incident with the copied nodes (both those edges used for the copied node clique and those edges with the rest of the attack graph).

This approach allows us to generate colored layered attack graphs conformant with the attack template but that randomly instantiate actor types into actors instances and randomly enable available attack edges from the template.

6. EMPIRICAL RESULTS

In this section we empirically evaluate the effectiveness of our proposed SCP and MCC algorithms as we scale the network in various dimensions. We use random layered attack graphs for the majority of the evaluation but then also apply our results to our instantiation of the ICS template.

We run our experiments in a virtualized Ubuntu operating system provisioned with a two Intel i7 cores and 10 GB of RAM. Our experimental system thus represents lower end commodity hardware.

6.1 Random Layered Graphs

We used random layered graphs to test the effectiveness and execution time as we scaled the number of colors, number of layers, and number of nodes per layer for both measuring SCP and MCC.

6.1.1 Shortest Color Path Measurements

We constructed random layered graphs with 10 layers, 100 layers per node, 100 attack sources (at layer 1), and 100 attack targets (at layer 10). The edge probability within a layer and between adjacent layers for each pair of nodes was 0.1. The attack sources in layer 1 represent a network where compromises are likely (e.g., where hosts communicate regularly with the Internet). The attack targets represent a layer of high value targets (e.g., hardware controllers in an ICS).

Using this model, we varied the size of the set of colors available. Each node was randomly assigned a color from the color set using a uniform distribution. This enables us to compare the relative effectiveness of our heuristics as the number of colors in a large graph increase.

We also ran experiments where we used the same setup but varied the number of layers and separately then number of nodes per layer (holding the number of colors constant at 50). There were no change in the relative effectiveness of the algorithms when varying the input along these dimensions.

Figure 6 shows the results for the Dijkstra 'naïve' approach, our SCP GA, and our SCP greedy BFS algorithm (our SCP algorithms with polynomial running times). Each data point represents the mean of 50 trials using different randomly generated graphs.

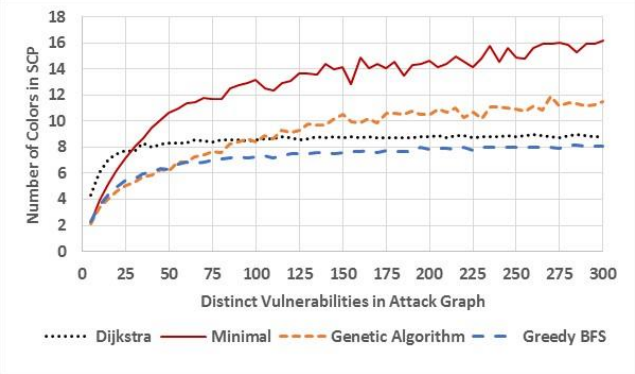

Figure 6. Relative Comparison of SCP Cardinality

The GA approach performs the best for up to 55 colors, after which the SCP greedy BFS provides the best solution. This is not unexpected because the GA is performing a multi-dimensional search on the bit string of color values. As more colors are added, the dimensionality of the problem increases making it more difficult for the GA to find effective local minimums. The naïve Dijkstra approach never provides the best solution although it converges toward the optimal as the number of colors increase. This is because once the number of colors equals the number of node in the network the problem converts into a standard shortest path problem. Likewise, the greedy BFS behaves more and more like the Dijkstra approach as the number of colors increases and thus also similarly converges to the optimal.

We now move from comparing relative performance to a comparison against the exact SCP using the same experiment. Our algorithm to calculate the exact SCP has a combinatorial term which results in an exponential increase in execution time. Thus, Figure 7 shows the results only for small numbers of colors.

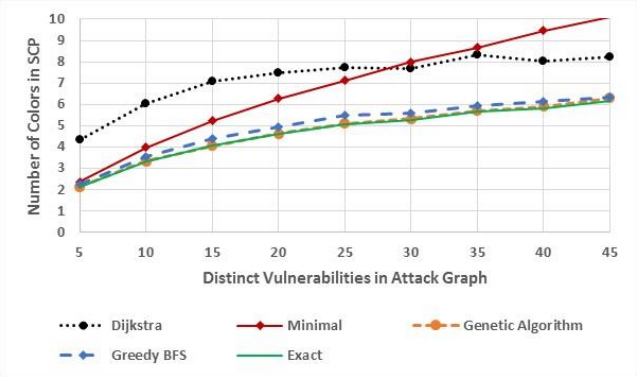

Figure 7. Exact Comparison SCP Cardinality

We see that the GA and greedy BFS approaches both provide close approximations to the exact SCP cardinality for the number of colors evaluated. The GA provided slightly better approximations in this color range. The naïve Dijkstra approach performed noticeably worse.

We next evaluate how the execution time for the algorithms changes as the number of colors increase, shown in Figure 8.

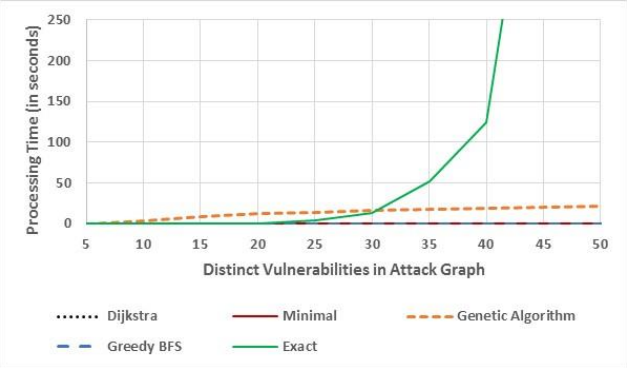

Figure 8. Growth of SCP Execution Time as the Total Number of Colors Increase

Here we see that the Dijkstra's algorithm and the greedy BFS operate essentially instantaneously. Both of them hug the x-axis and never take more than 0.01 seconds (even for 300 colors, not shown). The GA takes 41 seconds at 300 colors. The exact approach can clearly be seen to take exponential time making it tractable only for graphs with less than around 50 colors on our experimental setup.

6.1.2 Minimum Color Cut Measurements

We used the same experimental sets as with the SCP measurements and we apply it to compare the relative effectiveness of the MCC algorithms with an increasing number of colors.

As with the SCP measurements, we also ran experiments where we used the same setup but varied the number of layers and separately then number of nodes per layer (holding the number of colors constant at 50). There were no changes in the relative effectiveness of the algorithms when varying the input along these dimensions.

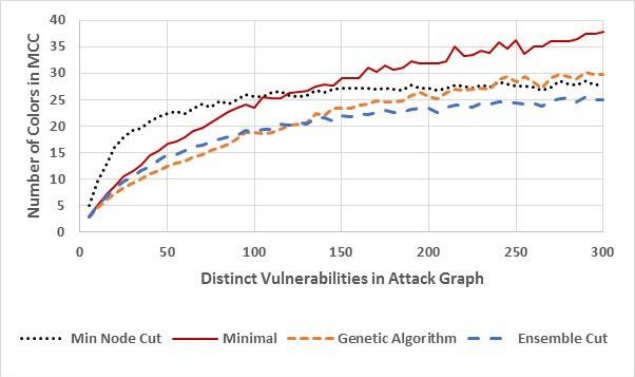

Figure 9. Relative Comparison of MCC Cardinality

As with the SCP GA, the MCC GA performs the best for a small number of colors but then loses performance as the dimensionality of the problem increases. For MCC measurements, this crossover happens at 130 colors. Our ensemble approach of color aware minimum node cuts performs the best after 130 colors but is reasonably close to the GA up to that point. The naïve minimum node cut approach always does much worse than our ensemble approach.

Next we compare the algorithms against exact answers, limited to a small number of colors since our algorithms to find the exact answers are exponential. Figure 10 provides the comparison up to 20 colors.

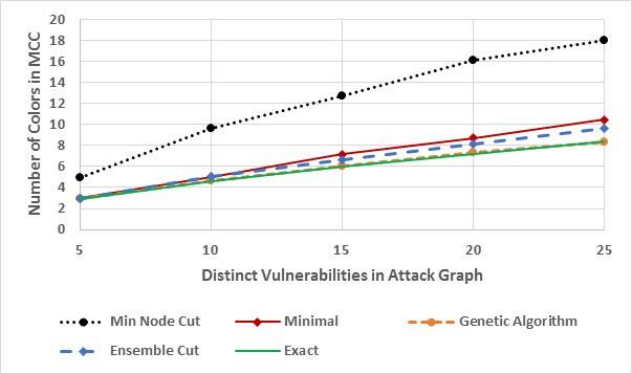

Figure 10. Exact Comparison of MCC Cardinality

We see that the naïve minimum node cut approach provides the worst performance. The GA so closely matches the exact answer that the lines are almost not distinguishable on the graph. The GA is at most 0.08 from the exact answer while. Our ensemble approach is never more than 1.25 away.

We next evaluate how the execution time for the algorithms changes as the number of colors increase, shown in Figure 11.

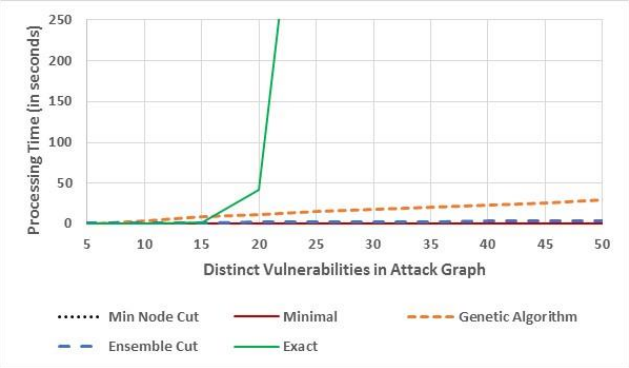

Figure 11. Growth of MCC Execution Time as the Number of Colors Increase

The exact solution clearly grows exponentially. The naïve minimum node cut approach takes at most 0.11 seconds and our color aware ensemble variant takes at most 6.8 seconds at 300 colors. The GA takes 80 seconds at 300 colors.

6.2 Industrial Control System Data Set

The ICS graphs were primarily used as a proof-of-concept test for both our SCP and MCC metrics as well as the algorithms used to implement them. These graphs were instantiated from an attack template representing an ICS secured to industry standards. Thus, each link in our instantiated attack graphs represented actual attack types that occur between the modeled services/entities.

From this exercise, we obtained evidence that real ICS networks are layered (the attack template itself had 8 layers) confirming our overall approach of processing attack graphs using a layered analysis.

Note that the number of layers in the attack template did not reveal the SCP of the instantiated graphs. Depending upon which of the instantiated entities has the vulnerabilities and which actually communicated, the SCP varied significantly. With fewer vulnerable nodes and less communication, the SCP tended to be longer while with a high presence of vulnerable nodes the SCP often was smaller than the number of layers in the attack template. This occurred because some of the attack segments crossed multiple layers.

The MCC of the instantiated graphs might at first appear to be equal to the MCC of the attack template. While a correlation exists, they are not necessarily equal. The MCCs of the instantiated graphs vary based on which of the attack paths from the template are actually available in the instantiated ICS graphs. Note the nodes must be both vulnerable and have the ability to communicate in order for them to be used in an attack path.

7. DISCUSSION

We envision our SCP and MCC metrics being used to provide an understanding of a network's DID posture with respect to expected sources of attack (be it external or internal). This will be most useful for networks with entry points that are threatened and that have 'crown jewel' servers to be protected.

A single network can be measured over time to determine the relative trending of the DID posture. The 'colors' returned from our algorithms can be used to highlight vulnerability types that, if fixed, could significantly improve the DID posture.

The SCP and MCC metrics can also be used to compare the DID posture between multiple networks of the same enterprise. This could be used for accountability purposes. It might be more impactful to use the comparison metrics to determine which networks need investment in patch/remediation technologies or in DID architectures.

8. RELATED NETWORK SECURITY MEASUREMENTS

A significant amount of work has been done examining the role of diversity in security. A large subfield of this research focuses on inducing diversity on the level of a single system [10] [11] [12]; this is beyond the scope of what we consider, however it could be considered as a method by which to generate additional 'colors' to introduce to our model.

On the network level, which is more directly applicable to our current problem, the work of [13] considers a simple Boolean measure of "survivability" that simply indicates whether an attacker possesses sufficient attacks to compromise all components of a particular network or not, without considering the possibility of multiple paths to a particular goal. A more sophisticated reduction of network security properties derived from diversity to coloring algorithms is explored by O'Donnell and Sethu [14]. They focus on the case of a fixed network topology, and consider the allocation of different version of software to different nodes within that network to impede the ability of an attacker to find a reliable attack path, in effect finding either a perfect coloring of the graph or a minimally defective coloring. They examine standard heuristics to obtain these goals, as well as the impact of adversarial activity during the coloring process on finding such colorings.

Within the context of ICSs, DID is typically presented as layering of independent security mechanisms, including 'soft' ones such as corporate policy and proper training. Depth of network topology, while occasionally mentioned [15], is rarely discussed in detail beyond simple approaches such as use of a "demilitarized zone" and firewalls between different segments of a network that includes ICS services [15] [16]. Our examination of heterogeneous devices as imposing additional burdens on an attacker has not yet been considered in this context, to the best of our knowledge.

Our work bears some relationship to traditional attack graph work, as does much other work in risk estimation and mitigation in ICS and SCADA systems [17]. Indeed, much work has been done on various minimization problems associated with attack graphs. In their study of properties of attack graphs, Jha et al. [18] consider a question related to our MCC property in examining the smallest subset of some set of defensive measures whose implementation makes the network under consideration secure; by reduction to minimum hitting set, they show that this property too is NP-hard to satisfy. Wang et al. [19], focus on the preconditions required to trigger the various exploits or vulnerabilities within a network, and examine ways to minimize the cost of removing these pre-requisites and thus denying access to attackers. While many results are analogous, their attack graphs differ significantly from our approach in two significant ways. First, they typically consider only known attacks, and do not attempt to model risk inherent in novel attacks, as we do by considering device heterogeneity (although see the work of Ingols et al. [20] which briefly considers services and programs that could be targeted by a "zero day" attack). Second, they generally do not directly examine questions of depth in any manner analogous to our SCP approach, in which the adversary must transit through multiple heterogeneous devices to reach their goal (although see [21], which does consider the impact of traffic filtering through multiple layers of a network as a component of defense in depth).

9. CONCLUSIONS

DID is an important security paradigm for many types of networks and is one component among many that is important for measuring overall security. We have identified depth, width, and strength as three important characteristics of DID security.

In this work, we provided a novel and general method to express the DID of a network in terms of security depth and width. We modeled a network's DID characteristics using a novel colored attack graph approach that exposed the residual risks in a layered security architecture. The colors represented vulnerability types and enabled us to model the ability of an attacker to exploit a particular vulnerability if they have already successfully exploited that same vulnerability elsewhere in the network.

To measure depth and width, we provided the novel metrics of SCP and MCC along with a proof sketch showing that exact SCP and MCC measurements are NP-Hard. We leave evaluations of strength (the third characteristic) for future work given the difficulty of measuring it rigorously and defensibly. Finally, we provided computationally efficient and effective approximation algorithms for determining SCP and MCC.

We empirically tested our approaches on large randomly generated layered security architectures to show scalability and functionality for large networks. We also tested against ICS networks using a published ICS attack template to randomly generate conformant layered security architectures. This demonstrated functionality and applicability to one important DID domain.

Overall, the most effectiveness algorithms were the greedy shortest path algorithm for SCP and our ensemble of color aware node cut algorithms for MCC.

Performing these measurements can enable one to compare the relative DID security of a network over time or compare two different networks. This analysis can also guide enhancements to further strengthen a DID posture and thus increase overall attack resistance in critical and important infrastructure.

10. ACKNOWLEDGMENTS

This research was sponsored by the U.S. Army Research Labs and the National Institute of Standards and Technology (NIST). It was partially accomplished under Army Contract Number W911QX-07-F-0023. The views and conclusions contained in this document are those of the authors, and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory, NIST, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes, notwithstanding any copyright notation hereon.

11. REFERENCES

[1] U.S. National Security Agency, "Defense in Depth," [Online]. Available: https://www.nsa.gov/ia/_files/support/defenseindepth.pdf. [Accessed 25 06 2015].

[2] MITRE, "Common Vulnerabilities and Exposures," [Online]. Available: https://cve.mitre.org/.

[3] U.S. Department of Homeland Security, "Recommended Practice: Improving Industrial Control Systems Cybersecurity with Defense-In-Depth Strategies," 10 2009. [Online]. Available: https://ics-cert.us-cert.gov/sites/default/files/recommended_practices/Defense_in_Depth_Oct09.pdf.

[4] B. Lyons, "Applying a Holistic Defense-in-Depth Approach to the Cloud," 25 07 2011. [Online]. Available: https://www.niksun.com/presentations/day1/NIKSUN_WWSMC_July25_BarryLyons.pdf.

[5] S. Jordan, "Defense in depth: Employing a layered approach for protecting federal government information systems," 16 11 2012. [Online]. Available: http://www.sans.org/reading-room/whitepapers/bestprac/defense-depth-employing-layered-approach-protecting-federal-government-information-system-34047.

[6] S. Even and G. Even, Graph Algorithms, Cambridge, 2011.

[7] E. Byres, A. Ginter and J. Langill, "How Stuxnet Spreads - A Study of Infection Paths in Best Practice Systems," 2011. [Online]. Available: http://www.controlglobal.com/assets/11WPpdf/110228_Tofino_Stuxnet.pdf.

[8] "National Vulnerability Database," National Institute of Standards and Technology, [Online]. Available: http://nvd.nist.gov.

[9] P. Mell, R. Harang, "Minimizing Attack Graph Data Structures," in *Tenth International Conference on Software Engineering Advances*, Barcelona, 2015.

[10] G. S. Kc, A. D. Keromytis and V. Prevelakis, "Countering code-injection attacks with instruction-set randomization," in *Proceedings of the 10th ACM conference on Computer and communications security*, 2003.

[11] R. C. Linger, "Systematic generation of stochastic diversity as an intrusion barrier in survivable systems software," in *Proceedings of the 32nd Annual Hawaii International Conference on Systems Sciences*, 1999.

[12] B. Cox and D. Evans, "N-variant systems: a secretless framework for security through diversity," *Usenix Security*, 2006.

[13] Y. Zhang, H. Vin, L. Alvisi, W. Lee and S. K. Dao, "Heterogeneous networking: a new survivability paradigm," in *Proceedings of the 2001 workshop on New security paradigms*, 2001.

[14] A. J. O'Donnell and H. Sethu, "On achieving software diversity for improved network security using distributed coloring algorithms," in *Proceedings of the 11th ACM conference on Computer and communications security*, 2004.

[15] K. Stouffer, J. Falco and K. Scarfone, "Guide to industrial control systems (ICS) security," NIST, 2011.

[16] D. Kuipers and M. Fabro, "Control systems cyber security: Defense in depth strategies," United States Department of Energy, 2006.

[17] P. A. Ralston, J. H. Graham and J. L. Hieb, "Cyber security risk assessment for SCADA and DCS networks," *ISA transactions*, vol. 46, no. 4, pp. 583-594, 2007.

[18] S. Jha, O. Sheyner and J. Wing., "Two formal analyses of attack graphs," in *IEEE Computer Security Foundations Workshop*, 2002.

[19] L. Wang, S. Noel and S. Jajodia, "Minimum-cost network hardening using attack graphs," *Computer Communications* , pp. 3812-3824, 2006.

[20] K. Ingols, M. Chu, R. Lippmann, S. Webster and S. Boyer, "Modeling modern network attacks and countermeasures using attack graphs," in *Annual Computer Security Applications Conference*, 2009.

[21] S. Jajodia, S. Noel, P. Kalapa, M. Albanese and J. Williams, "Cauldron mission-centric cyber situational awareness with defense in depth," in *Military Communications Conference*, 2011.

[22] P. Mell, K. Scarfone and S. Romanosky, "NIST IR 7435: The Common Vulnerability Scoring System (CVSS) and Its Applicability to Federal Agency Systems," National Institute of Standards and Technology, 2007.