Abstract—The supervisory control and data acquisition (SCADA) network in a smart grid requires to be reliable and efficient to transmit real-time data to the controller. Introducing SDN into a SCADA network helps in deploying novel grid control operations, as well as, their management. As the overall network cannot be transformed to have only SDN-enabled devices overnight because of budget constraints, a systematic deployment methodology is needed. In this work, we present a framework, named SDNSynth, that can design a hybrid network consisting of both legacy forwarding devices and programmable SDN-enabled switches. The design satisfies the resiliency requirements of the SCADA network, which are specified with respect to a set of identified threat vectors. The deployment plan primarily includes the best placements of the SDN-enabled switches. The plan may include one or more links to be installed newly. We model and implement the SDNSynth framework that includes the satisfaction of several requirements and constraints involved in resilient operation of the SCADA. It uses satisfiability modulo theories (SMT) for encoding the synthesis model and solving it. We demonstrate SDNSynth on a case study and evaluate its performance on different synthetic SCADA systems.

Index Terms—SDN architecture; incremental deployment; smart grid; SCADA; formal modeling; network synthesis

I. INTRODUCTION

Smart grids are large, heterogeneous, and distributed in nature, which present a high complexity of maintenance for a large number of intelligent end devices. The SCADA network infrastructure of smart grid needs to be reliable and efficient to transmit large amount of real-time data. The observability of a grid bus system is determined by the successful delivery of critical measurements collected by the end devices (e.g., intelligent electronic devices (IED), remote terminal units (RTU), etc.). The overall network should be resilient to cyberattacks to ensure seamless transmission of control and measurement data from the devices for system observability.

Network infrastructures in SCADA systems use diverse protocols and heterogeneous forwarding devices, which make the management, maintenance, and integration of new devices difficult [1]. Software defined networking (SDN) has great potential to be used in SCADA systems [2]. It not only provides flexibility to implement novel networking solutions and quality of service (QoS) optimization but also provides greater resiliency to cyberthreats [3], [4]. SDN can provide flexible rerouting in case of congestion or compromise, prioritizing certain network components, and isolate compromised sections of the network more effectively [5], [6]. However, network upgrades in SCADA networks are budget and resource-constrained. It is impractical to substitute all the legacy switches with SDN switches overnight. The process of simultaneous deployment of legacy and SDN-enabled switches remains one of the greatest challenges in incorporating SDN.

In this work, we use the threats to the SCADA system, which are vulnerable sets of electronic devices discovered in [7]. The network can be resilient to attacks and the grid will be observable if these devices can be ensured proper communication with the control center. This research places the available SDN switches and links properly so that rerouting of control and data traffic, as well as setting up virtual networks, whenever needed, are possible.

The problem of deployment of SDN satisfying grid observability constraints within a limited budget, is a recent topic and is generally an NP-hard one [8]. Utilizing the available limited budget (e.g., a limited number of SDN-enabled switches), while perceiving the benefits of SDN, is challenging. We propose to formally model the constraints and requirements into a constraint satisfaction problem (CSP) and solve it using a CSP solver to generate the SDN-enabled network architecture. We present an automated framework, SDNSynth, which solves this problem using formal verification. It takes the existing network topology, security requirements, and resources as inputs and formulates the deployment problem. The problem is solved by encoding the model into first order logic. We use SMT for encoding which also provides a solution, if there is any, in the form of the deployment plan for SDN switches.

II. BACKGROUND AND OBJECTIVE

Several manual heuristic based algorithms have been devised in the literature to determine the locations of the limited number of SDN switches [8]. Hong et al. systematically studied the incremental SDN deployment problem by formulating it as an optimization problem and proposed effective heuristics for selecting a small set of existing devices for upgrading [8]. Levin et al. present the design and implementation of an architecture called Panopticon for operating networks that combine legacy and SDN switches [9]. Panopticon formalizes an optimal cost-aware upgrade algorithm based on mathematical programming [10]. A solution for seamless peering between SDN and existing IP networks is studied by Jonathan et al.
The major features of the SDNSynth framework are as follows:

1. **Formally models the network topology**, required configurability (i.e., SDN features) of switches by the controller, and resource constraints.
2. **Formalizes the incremental SDN design synthesis problem** as the determination of deployment decision of SDN switches, new links, and their placements that satisfy the given requirements and constraints.
3. **Encodes the synthesis problem into SMT logics and provides a feasible solution using an SMT solver.**

The tool takes input from a user using an input file. The output of the tool indicates the best possible candidates for switch replacements, as well as the new links that should be deployed in the network according to the budget.

### B. Priority Management

In a SCADA environment, we want to ensure that any critical end device must be able to communicate with others, as long as they are not compromised. First, we determine the criticality/priority of the field devices. To do this, we systematically assign ranks to all the IEDs. The rank increases if it has some attached sensors or actuators. The rank also increases if it is an element of one or more sets of threat vectors. The rank is used in calculating the priorities of the IEDs. For example, we can define three levels of priorities for the IEDs: high, medium, and low. When modeling the resiliency requirements of the devices, the devices having higher priority will be ensured more alternate paths and other resiliency features before others.

### C. Resiliency Management

Here we present our model according to several SDN benefits in the management of security and resiliency of SCADA.

**Alternate Paths:** First, we define alternative paths for the IEDs. In calculating alternate paths from an IED to an MTU, we consider \( \alpha \)-Alternative paths, where \( \alpha \) means the percentage of overlapped/shared links on the paths. If a path contains less than \( \alpha \% \) of common links with another path, it can be considered as an alternate to the other one.

We consider all possible forwarding paths from an IED \( i \) to the MTU, through one or more RTUs, as \( P_i \). A path \( p_{i,y} \) is defined as the \( y^{th} \) path among all possible paths. \( p_{i,y} \) is a set of links, while each link \( l \) represents a pair of nodes that belong to the set \( L \subseteq N \times N \), assuming that \( L \) is the set of links and \( N \) is the set of all nodes.

Let \( \mathbb{I} \) be the set of all IEDs in the threat vectors. If \( \text{AltPath}_{p_{i,y},p_{i,y}'} \) denotes that path \( p_{i,y} \) is an alternate path for \( p_{i,y}' \), then:

\[
\forall i \in \mathbb{I}, \forall p_{i,y}, p_{i,y}' \in \mathbb{P}_i, \text{AltPath}_{p_{i,y},p_{i,y}'} \rightarrow \left( \sum_{l \in p_{i,y}'} l \in p_{i,y} \right) \leq \left( |p_{i,y}'| \times \alpha \right) \quad (1)
\]

Next, we find the switches that split a path of an IED to create multiple alternate paths. If \( S \) is the set of all candidate switches to be replaced by SDN-enabled ones and \( L_s \) is the set of all links connected to switch \( s \), then \( \text{SwitchOnAltPathBranch}_s \) ensures that switch \( s \) is positioned where two or more alternate paths for IED \( i \) branches.

\[
\text{SwitchOnAltPathBranch}_s \rightarrow \text{AltPath}_{p_{i,y},p_{i,y}'} \wedge \sum_{l,l' \in L_s} \left( (l \in p_{i,y}) \land (l' \in p_{i,y}') \right) \geq 1 \quad (2)
\]
SDN-Enabled Switches: SDN-enabled switches should be deployed intelligently on the network branches. We want the SDN-enabled switches to be deployed on the alternate paths for the IEDs. Also, they should be deployed at the forks of the paths, so that SDN controller is able to route the data and command packets to and from IEDs efficiently, and according to priority. We define such alternate paths as software-defined alternate paths, SDAltPath_{p_i,y,p_i,y'}. If \text{SwitchIsSDN}_s denotes whether switch s is SDN-enabled or not, d_{i,y} is the set of all switches on the y^{th} path from IED i to the MTU, and d_{i,y'} is the set of all switches on path y', the following should hold:

\[
\forall i \in I \forall (s) \in d_{i,y}, \text{SDAltPath}_{p_i,y,p_i,y'} \land \text{SwitchIsSDN}_s \land \text{SwitchOnAltPathBranch}_s
\]

\[
\forall i \in I \text{AssuredMinAltPath}_i \to (\text{Priority}_i = m) \land \forall p_{i,y}, (1 + \sum_{y'} \text{SDAltPath}_{p_i,y,p_i,y'} \geq \text{minAltPath}_m)
\]

If m is the priority of IED i, in the Equation 4, minAltPath_m is a constant specifying the minimum number of alternate paths that need to exist between IED i and the MTU. Each path p_{i,y} for an IED i with priority m, should have at least minAltPath_m alternate paths p_{i,y}'.

Different number of alternate paths may be specified for different levels of priorities. For example, at least five alternative paths should be deployed for high priority communications, while low priority devices may require at least two alternate paths, for each of its paths to the MTU.

All the links in the communication paths and their alternate paths need to be deployed and up. If Link_i is a boolean variable denoting whether a link is up or not,

\[
\forall i \in I \text{AssuredLinksOnAltPath}_i \to \text{SDAltPath}_{p_i,y,p_i,y'} \land \forall l_{i,y}, \text{Link}_i \land \forall l_{i,y'}, \text{Link}_{i,y'}
\]

AssuredMinAltPath_i and AssuredLinksOnAltPath_i ensure the existence of alternate paths from IED i to the MTU.

\[
\forall i \in I \text{AssuredMinAltPath}_i \to \text{AssuredMinAltPath}_i \land \text{AssuredLinksOnAltPath}_i
\]

If altPathExp denotes the expected percentage of IEDs to have assured alternate paths, then the following should hold:

\[
\sum_i \text{AssuredAltPath}_i / I \geq \text{altPathExp}
\]

D. Resources and Budget Constraints

It is important in an SDN environment that any communication is supervised by the SDN controller. This means that any packet from a source must traverse through at least one SDN switch on its way to the destination. Let ESwitchIsSDN_s denote an already existing SDN switch s and DSwitchIsSDN_s be the deployable new SDN switch. The following should hold about the already existing switches or routers and the newly deployable SDN-enabled switches.

\[
\forall i \in I \text{AssuredMinAltPath}_i \to \text{ESwitchIsSDN}_s \land \text{DSwitchIsSDN}_s
\]

\[
\text{SwitchIsSDN}_s \to \text{ESwitchIsSDN}_s \lor \text{DSwitchIsSDN}_s
\]

\[
(\text{ESwitchIsSDN}_s \to \neg \text{DSwitchIsSDN}_s) \land (\text{DSwitchIsSDN}_s \to \neg \text{ESwitchIsSDN}_s)
\]

Similarly, if DLink_i denotes a newly deployed link that does not exist and need to be set up, then,

\[
\text{Link}_i \to \text{ELink}_i \lor \text{DLink}_i
\]

\[
(\text{ELink}_i \to \neg \text{DLink}_i) \land (\text{DLink}_i \to \neg \text{ELink}_i)
\]

The total budget, TOT_AVAIL_BUDGET, hence the number of available SDN-enabled switches and links are limited. It is not possible to replace more number of switches than the total available SDN switches. That is, the sum of all deployed SDN switches and links must be less than what we have in budget. This can be represented by the following constraint, given c_{Link_i} represents the deployment cost of new DLink_i and c_{SDN} denotes the cost of each SDN switch:

\[
\left( \sum_i \text{DLink}_i \times c_{\text{Link}_i} + \left( \sum_s \text{DSwitchIsSDN}_s \times c_{\text{SDN}} \right) \right) \leq \text{TOT_AVAIL_BUDGET}
\]

IV. IMPLEMENTATION AND A CASE STUDY

A. Target Variables in the Model

We implement our model by encoding the system configuration and the constraints into SMT logics [15]. In this encoding purpose, we use Z3, an efficient SMT solver [16]. The solver provides a satisfiable (sat) result if all the constraints are satisfied. The sat result provides the value assignments to the required Boolean parameters of the model, e.g., SwitchIsSDN_s, DLink_i, etc., which represent the new network topology.

B. A Synthetic Case Study

We present a small network of a 14-bus SCADA system for our case study. It consists of 26 IEDs, 13 RTUs, and 1 MTU. There are 18 traditional routers connecting these intelligent devices. The input file consists of the network topology, the number of possible new links, etc. It also includes 8 threat vectors where each is a set of IEDs, the expected percentage of threat mitigation (75%), average cost of new SDN switches.
First, we determine the $k$-resiliency of the SCADA network, which means the network is resilient to less than $k$ IED failures [7]. In other words, the system is still observable if less than $k$ IEDs fail to deliver their measurement data to the MTU. The value of $k$ in our network is 4, which means that it is 3-device-failure resilient. If the attacker is capable of compromising up to 4 devices, this scenario yields 1 threat vector consisting of 4 devices; whereas, for a 5-device failure we get 8 different threat vectors. We use the 8-threat vector scenario for our case study.

We have a budget of $35,000. The SDN-enabled switches should replace some of the traditional routers. We would like to mitigate at least 75% of the threat vectors through the placement of SDN-enabled switches. We have a limited budget and 6 possible new links. Given all the constraints and requirements, SDNSynth generates a satisfiable solution that tells the user to replace the routers 40, 42, 43, 44, and so on. This is demonstrated in Fig. 2. Also, new links between switch 44 and 55; 45 and 47; 52 and 57; and so on, are suggested. The proposed network topology ensures alternate paths for the critical IEDs according to the percentage of expected IEDs having alternate paths. The alternate paths enable the SDN controller to reroute any critical data to and from the IEDs in the case of any link or device failure.

V. Evaluation

To evaluate SDNSynth, we run experiments on different synthetic SCADA network topology for different sizes of IEEE test bus system in smart grids, i.e., 14-bus, 30-bus, 57-bus, and 118-bus. We ran the program on a machine with Windows 10 OS, an Intel Core i7 processor and 16 GB memory.

A. Relationships of Deployment Parameters

In this analysis, the resultant numbers of SDN-enabled switches and links are the minimum numbers of switches and links possible with the tightest possible constraints.

In Fig. 3(a), we demonstrate the number of deployed SDN-enabled switches, as well as the number of newly deployed links, with respect to the total available budget. The threat mitigation requirement was set to a value low enough to utilize all possible resources. As the available budget increases, the number of newly deployed SDN switches and links also increases slowly. The resource constraints are responsible for this increase, as SDNSynth tries to find a solution utilizing the available budget. Proper positioning of the SDN switches and links allow more flexible rerouting.

B. Scalability

We evaluate the scalability of SDNSynth by analyzing the time required to synthesize the network topology by varying the problem size and other parameters.

Impact of Bus Size: The model synthesis time with respect to the incoming bus size is shown in Fig. 3(b). Two scenarios, one for satisfiable results and another for unsatisfiable results, are presented in the graph for 8 threat vectors. We observe that the required time increases in somewhere between linear and quadratic orders with the increment of bus size. The execution time differs for sat and unsat results for a specific bus size. The unsat results usually take more time than sat ones. As the bus size increases, the number of constraints and requirements increase rapidly. For this reason, we observe such timing (almost quadratic) for obtaining a result.

Impact of Budget: Fig. 3(c) shows the impact of budget on the network synthesis time. All the results are taken for sat solution for the lowest budget. We can observe that if the bus and SCADA size, threat vectors, as well as all other requirements are kept constant, the tool requires almost similar times for synthesizing the network. The network size is larger for the 57 bus than the 30 bus. As a result, there are more constraints to solve, hence it takes more time to provide a satisfiable result.

VI. Conclusion

SDNSynth is a tool for synthesizing a resilient SDN topology in smart grid SCADA systems. We protect the critical IEDs, with the use of SDN-enabled switches, which allow fast rerouting, prioritization of packet flows, novel application-based routing, etc. The technique successfully generates a solution depicting the SDN switch and new link placements, while satisfying the resiliency requirements and budget constraints. We evaluate the tool by scalability analysis and satisfactory relationship between different deployment parameters.
REFERENCES


