BAGUETTE: Towards a Secure and Cost-effective Switch Upgrade in Hybrid Software-Defined Networks

Wendi Feng*, Zehua Guo†, Chuanchang Liu*, Yueming Zheng‡, Meng Wang* Bo Cheng* and Junliang Chen*
*Beijing University of Posts and Telecommunications
†Beijing Institute of Technology
‡ Trinity College Dublin

***Abstract*—Software-Defined Networking (SDN), providing flexible controlling and monitoring mechanisms that simplifies network management, is becoming prevalent in recent years. However, replacing all legacy network devices with SDN-capable devices is cost-prohibitive. One practical approach for the SDN deployment is to incrementally upgrade a few legacy devices to SDN devices. The network, which consists of legacy and SDN devices, is called a hybrid SDN. Existing hybrid SDN deployment schemes do not consider the security impact of device deployment. They use the same type of devices to upgrade, and upgraded devices could be compromised if an attacker controls one SDN device by leveraging its vulnerabilities.**

In this paper, we consider this security issue in the hybrid SDN deployment and present the Secure and Cost-effective Switch Upgrade (SCESU) problem. The SCESU problem aims to upgrade a few network devices to satisfy the security requirement by using multiple SDN switch types with a minimum upgrade cost. The complexity of the SCESU problem comes from *common vulnerabilities* shared among different types of SDN devices and *attack propagations* among network nodes. To efficiently solve the problem, we propose the BAGUETTE algorithm to judiciously choose and upgrade critical legacy switches with selected SDN devices. Simulation results show that BAGUETTE achieves up to 12.6x security enhancement compared with legacy network and reduces to 11.1% cost of the securest deployment.

***Index Terms*—SDN, Hybrid SDN deployment, security, attack mitigation.**

I. INTRODUCTION

Software-defined Networking (SDN) [1] is a prevalent networking technology, which decouples the control plane and data plane of network devices (i.e., SDN switches) and allows innovations to be easily applied. It also simplifies network management with the fine-grained network controlling and monitoring mechanisms. Owning to the advantages, SDN plays an important role in Cloud Computing [2], Edge Computing [3], 5G [4], and Fog Computing [5]. Many industrial companies, such as Google [6] and Facebook [7], have started deploying SDN in their production environments. AT&T, as one of the biggest Internet Service Providers (ISP), also plans to increase the SDN deployment to 75% by 2020 [8].

SDN deployment is a long way to go because replacing all legacy devices with SDN devices is cost-prohibitive. One practical way to deploy SDN is to incrementally upgrade a few legacy devices to SDN devices, which makes the network a hybrid SDN [9], [10], [11], [12].

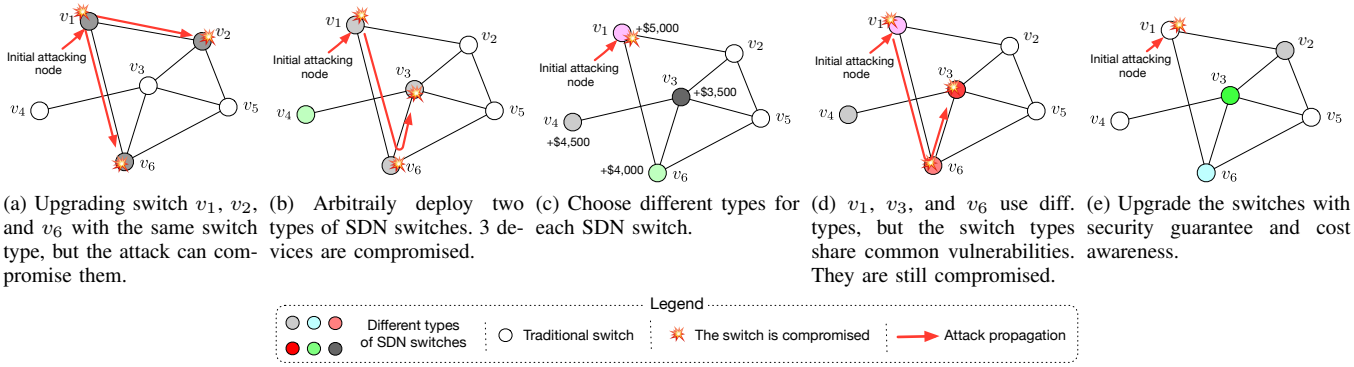
Typically, existing works consider many constraints (e.g., upgrade budget [11], flow programmability [13], [14]) for the SDN device upgrade. However, they use only one type of SDN devices and do not consider the security impact of device deployment. If the attacker controls one device by leveraging its vulnerabilities, other devices with the same type can also be compromised. For example, a zero-day attack can exploit a vulnerability of a switch type and construct a special packet that can overflow its memory buffer and controls the switch [15]. When controlling one switch, the attacker can then broadcast the attack packet to all its neighbors. If neighbors use the same switch type, they also have the vulnerability and can thus be leveraged by the attacker and compromised.

The question is whether we can mitigate attacks and make the network resilient to attacks by using devices with different vulnerabilities in the hybrid SDN deployment? The answer is yes. Devices have defects[1], but each attack may only leverage one or a few specific defects. If using a device whose defects that cannot be leveraged by the attack, the attack cannot compromise the device.

Simply using different switch types at each network node can be either unnecessary or insufficient in the hybrid SDN deployment. The upgrade cost will be exorbitant since this often upgrades more switches than needed and increases the expense when the security level has already been satisfied (detailed in Section II-C). Security may fail to be provided because common vulnerabilities reside in multiple switch types. Thus, the attack can still propagate and compromise the network.

Inspired by the above observation, we present the Secure and Cost-effective Switch Upgrade (SCESU) problem in this paper, which jointly considers security and upgrade cost in the hybrid SDN deployment. In a nutshell, SCESU aims to use a few SDN devices to upgrade legacy devices to satisfy the security requirement with a minimum upgrade cost. The complexity of the problem is resulting from *common vulnerabilities* of switch types and *attack propagations* among network nodes. We propose an efficient heuristic algorithm called BAGUETTE to solve the problem. It judiciously chooses

[1]Including those undisclosed. We interchangeably use defects, flaws, vulnerabilities throughout this paper.



(a) Upgrading switch v_1, v_2, and v_6 with the same switch type, but the attack can compromise them.

(b) Arbitrarily deploy two types of SDN switches. 3 devices are compromised.

(c) Choose different types for each SDN switch.

(d) v_1, v_3, and v_6 use diff. types, but the switch types share common vulnerabilities. They are still compromised.

(e) Upgrade the switches with security guarantee and cost awareness.

Fig. 1: Motivation examples demonstration. Unsuccessful attack propagation is omitted for a clearer illustration.

a small portion of the legacy devices and upgrades them with smartly selected device types. Simulation results show that the proposed BAGUETTE algorithm achieves a near-optimal performance and achieves up to 12.6x security enhancement and reduces to 11.1% cost of the most secure method.

In summary, our contribution is three-fold, as follows.

- We consider the security impact of SDN device deployment in hybrid SDN, and mathematically formulate the SCESU problem.
- We propose BAGUETTE to efficiently solve the SCESU problem, which satisfies the security requirement with a minimum upgrade cost.
- We evaluate BAGUETTE under various real-world topologies and compare them with baseline algorithms. Simulation results show that BAGUETTE achieves near-optimal performance.

The rest of this paper is organized as follows. Section II introduces the attack model and motivates the SCESU problem with examples. Section III mathematically formulates the SCESU problem, and in Section IV, we propose BAGUETTE to efficiently solve the SCESU problem. Section V describes the simulation setup and compares BAGUETTE with baseline algorithms. Section VI introduces the most relevant related works, and Section VII concludes the paper.

II. Attack Model and Motivation

This section introduces our attack model and motivates the SCESU problem with examples. In the examples, we show that the single SDN switch type upgrade fails to mitigate attack propagations, and thus needs multiple types to upgrade the network. However, merely considering the security factor results in exorbitant cost, while concerning about the cost alone fails to guarantee the security requirement. Thus, we need an intelligent hybrid SDN deployment mechanism by jointly considering security and cost to achieve the Secure and Cost-effective Switch Upgrade (SCESU) in the hybrid SDN switch upgrade.

A. Attack Model

In this subsection, we introduce our attack model. It is notoriously hard to develop a computer system without defects. We consider the attack has the following 4 characteristics. i) The attack compromises the network devices (e.g., switches, routers) by leveraging their defects. ii) The attack can propagate to adjacent nodes if the current node is compromised, and iii) if the device defends the attack, the attack is cleared and terminated. iv) The attacker can only use known attack methods to conduct attacks. Many attacks in the real world have these characteristics (e.g., packet overflow attack, DoS (Denial of Service) attack). Note that detailed attack processes are out of the scope of this paper.

B. Insecure Single SDN Switch Type Upgrade

Fig. 1 shows examples of a simple network with 6 nodes, and each node has a switch. At first, the network is a traditional network, and all nodes are deployed with legacy switches. Fig. 1a shows that the network needs controlling and management benefits provided by SDN and upgrades nodes v_1, v_2, and v_6 to SDN switches with the same type of switches. However, using the same type of switches to upgrade also leads to vulnerabilities shared among v_1, v_2, and v_6. Thus, if v_1 is compromised by an attack, then the attack would propagate to v_2 and v_6. Thus, the attack can further compromise v_1, v_2, and v_6. Consequently, using only one switch type to upgrade switches is not secure, and thus, multiple types should be used to improve security.

C. Curse of Arbitrary Upgrade with Multiple Types

Generally speaking, multiple switch types deployment usually enhances security because the vulnerabilities can be different on different types of switches. However, arbitrarily upgrading switches with multiple types would also fail to protect the network. In Fig. 1b, if nodes v_1, v_3, v_4, and v_6 are arbitrarily selected to upgrade to SDN switches, where v_1, v_3, and v_6 use one switch type (shown in gray color), and v_4 uses another type (shown in light green color), and the gray and light green types do not share common vulnerabilities. Suppose the attack arrives at v_1 and the attack can propagate to switch v_6, then to v_3, and finally to v_4. Thus v_1, v_3, and v_6 are compromised. When the attack propagates to v_4, it cannot compromise v_4, and the attack stops. However, half of the network devices are compromised.

TABLE I: Notation Definitions.

Notation	Description
$\mathcal{V}$	The network node set. $\mathcal{V} = \{v_1, v_2, \dots\}$.
$\mathcal{S}$	The switch type set. $S = \{s_1, s_2, \dots\}$.
$\mathcal{C}$	The cost set. Costs of upgrading a node to each switch type. $C = \{c_1, c_2, \dots\}$.
x_{ij}	Node v_i is uses type s_j.
$\mathcal{F}_j$	Switch type s_j's vulnerabilities. $F_j = \left\{f_1^j, f_2^j, \dots, f_{l_j}^j\right\}$.
$\mathcal{A}_j$	The attack set of the switch type s_j. $\mathcal{A} = \{a_1, a_2, \dots\}$.
P_j	The attack probability of switch type s_j.
X	The switch type mapping scheme. $x_{ij} \in X, \forall v_i \in \mathcal{V}, \forall s_j \in \mathcal{S}$.
$e(X)$	The compromising expectation of the SDN network under upgrade scheme X.
E^{max}	The maximum compromising expectation provided by network operator.
$\mathcal{V}_i^{*a_k}$	The compromising sub-graph nodes set.
$r_i^{a_k}(X)$	The compromised ratio under the upgrade scheme X and attacked by attack a_k when network node v_i is the entry node.

As depicted in Fig. 1c, using different types for each network node may mitigate the issue. However, the overall cost would be exorbitant since the security requirement can be satisfied by upgrading a few number of the network nodes. Since each type of switch has different costs [16], one possible way to lower the cost would be simply sorting the switch types based on their cost from low to high and upgrade the nodes using the sorted types one by one to minimize the cost. However, even different switch types may contain *common vulnerabilities* due to the use of shared libraries and hardware components, and attacks can leverage these common vulnerabilities to compromise the network. Moreover, production environments require optimized switches for specific businesses, which demands experts who proficient in a specific type of switch. Thus, using more types of SDN devices increases the budget level. As shown in Fig. 1d, even v_1, v_3, and v_6 use different types, the attack can still propagate from v_1 to v_6 and v_3 and compromise all of them. Consequently, arbitrarily upgrading the switches is not reliable.

D. Security and Cost-effective Switch Upgrade

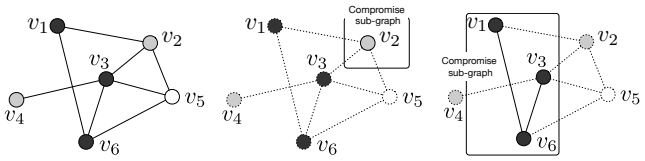
By selecting vulnerability-distinct switch types next to the attack entry node, the attack can be mitigated at the initial stage. In Fig. 1e, the upgraded switch can efficiently mitigate the attack on its propagation path and achieve the minimum overall cost through the SCESU intelligent SDN type upgrade scheme. This paper presents the design of our SCESU scheme that guarantees the security requirements at a minimum cost.

III. Problem Formulation

In this section, we first mathematically present the network system description and then propose security metrics to describe the possibility of compromising the network. We further introduce constraints and the objective function of the SCESU problem. Finally, we formulate it as an optimization problem.

A. System Description

We mathematically formulate the network in this subsection. Notation definitions can be found in Table I.



(a) The network consists of 3 switch types (shown as black, gray, and white).

(b) An attack compromises gray type nodes and enters at v_2.

(c) An attack compromises black type nodes, and enters v_1.

Fig. 2: Compromising sub-graph demonstration. In Fig. 2b, the compromising sub-graph only contains v_2, and the compromised ratio is $\frac{1}{6}$. In Fig. 2c, the compromising sub-graph contains v_1, v_3, and v_6, and the compromised ratio is $\frac{3}{6} = \frac{1}{2}$.

A network can be represented as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, v_2, \dots\}$ is the set of network nodes, and v_i represents the i^{th} node in the network, and $\mathcal{E}$ is the links set. Each node in $\mathcal{V}$ can choose multiple switch types to upgrade. Let $\mathcal{S} = \{s_1, s_2, \dots\}$ represents the switch type set. In the default setting, all nodes are deployed with a legacy switch. Different switch types have different upgrade costs, and the cost of legacy switches are 0. Let $\mathcal{C} = \{c_1, c_2, \dots, \}$ represents the cost of upgrading to each switch type. We use $x_{ij} = 1$ to indicate network node v_i is deployed with switch type s_j, and otherwise $x_{ij} = 0$.

B. Attack Metrics

In this subsection, We introduce probability-based security metrics to measure the security of the network. Specifically, we use *attack probability* to measure the possibility of compromising a specific switch type by an attack. We then consider the influence of attack propagation with the *switch compromised ratio*. Finally, we present the *compromising expectation* to evaluate the overall compromising expectation of the network.

1) Attack Probability: Each SDN switch type has multiple vulnerabilities. Let $\mathcal{F}_j = \left\{f_1^j, f_2^j, \dots, f_{l_j}^j\right\}$ be the set of all vulnerabilities of type s_j, where l_j is the number of vulnerabilities of type s_j. Let $\mathcal{F} = \bigcup_j \mathcal{F}_j$ be the total vulnerability set, and let $\mathcal{A} = \{a_1, a_2, \dots\}$ be the attack set. The attack set contains all possible public known attacks and unpublished attacks based on the network operator's previous experiences. Each attack can exploit one or more vulnerabilities, so each attack a_k is a subset of the vulnerability set $\mathcal{F}$ (exclude $\emptyset$), and thus, the total number of attacks is $2^{|\mathcal{F}|} - 1$. If attack a_k can exploit the vulnerability that switch type s_j has, then type s_j is compromised by attack a_k.

The attack probability of switch type s_j is the ratio of the number of attacks that can compromise the type to the total number of attacks. It is formulated as

$$P_j = \frac{\left| \bigcup_{a_k \cap \mathcal{F}_j \neq \emptyset} \{a_k\} \right|}{|\mathcal{A}|}. \tag{1}$$

2) Switches Compromised Ratio: An attack can traverse from one node to another, if these nodes are adjacent to each

other and have the same type of device or share common vulnerabilities. We present the concept of the *compromising sub-graph* to identify the possible compromised switches of given entry network node v_i and attack a_k. The compromising sub-graph is generated by 2 steps. i) Adjusting the adjacency matrix of the network graph, by removing the nodes in the adjacency matrix whose type's vulnerabilities have no intersection with attack a_k. ii) Traversing the graph with the adjusted adjacency matrix by using breadth-first search (BFS) to get all the switches that can be reached in the graph traversal. Then, we get $\mathcal{G}'^{a_k}_i = (\mathcal{V}'^{a_k}_i, \mathcal{E}'^{a_k}_i)$, which is the generated compromising sub-graph. Fig. 2a depicts a network with 6 switches where v_1, v_3, v_6 use the "black" type; and v_2, v_4 use the "gray" type; and v_5 use the "white" type. In Fig. 2b, v_2 is the only element in $\mathcal{V}'^{a_1}_2$. And in Fig. 2c, $\mathcal{V}'^{a_2}_1$ contains v_1, v_3, v_6. The switch compromised ratio is formulated as

$$r^{a_k}_i(X) = \frac{\left|\mathcal{V}'^{a_k}_i\right|}{|\mathcal{V}|}. \tag{2}$$

3) Compromising Expectation: In this subsection, we propose the *compromising expectation* to measure the overall compromising possibility of the network. The compromising expectation is formulated as a mathematical expectation shown below

$$e(X) = \frac{1}{|\mathcal{V}|} \sum_{v_i \in \mathcal{V}} \sum_{a_k \in \mathcal{A}} \sum_{s_j \in \mathcal{S}} r^{a_k}_i(X) P_j x_{ij}, \tag{3}$$

where $r^{a_k}_i(X)$ is the switches compromise ratio of switch v_i under attack a_k, and P_j is the attack probability of switch type s_j under all attacks.

C. Constraints

1) Maximum Compromising Possibility: Security requirements may vary between network to network based on their business. Therefore, a minimum security level, or put another way, a maximum compromising possibility can be set by the network operator to guarantee the least security requirement of the network. Thus we have

$$\frac{1}{|\mathcal{V}|} \sum_{v_i \in \mathcal{V}} \sum_{a \in \mathcal{A}} \sum_{s_j \in \mathcal{S}} r^{a}_i(X) P_j x_{ij} \leq E^{max}, \tag{4}$$

where E^{max} is the maximum required compromising possibility of the network.

2) Single Switch Type Constraint: Only one switch type can be used for each node in the network. The constraint is written as

$$\sum_{s_j \in S} x_{ij} = 1, \forall v_i \in \mathcal{V}. \tag{5}$$

D. Objective Function

Our objective is to minimize the overall cost of switch type mapping for each node in the network. Therefore, the objective function is written as follows

$$obj = \sum_{v_i \in \mathcal{V}} \sum_{s_j \in S} x_{ij} c_j. \tag{6}$$

E. Problem Formulation

The goal of the SCESU problem is to find an optimal switch type mapping scheme between network nodes in $\mathcal{V}$ and types in $\mathcal{S}$ by judiciously placing the *suitable* type to the switch, thus reaching the target of minimizing the overall cost under the security requirement. Consequently, we formulate the SCESU problem as follows:

$$\begin{aligned} \min_{x} & \sum_{v_i \in \mathcal{V}} \sum_{s_j \in \mathcal{S}} x_{ij} c_j \\ \text{s.t.} & (4)(5), \\ & x_{ij} \in \{0, 1\}, \\ & v_i \in \mathcal{V}, s_j \in \mathcal{S} \end{aligned} \tag{P}$$

where $\{c_j\}$ are constants, and $\{x_{ij}\}$ are designed variables. In the SCESU problem, the objective function is linear, and variables are binary integers. Thus, this problem is an Integer Linear Programming (ILP) problem.

IV. Problem Solution

The complexity of the SCESU problem comes from both the attack can propagate among network nodes, and vulnerabilities can share between switch types. This section presents an efficient heuristic algorithm called BAGUETTE to solve it.

The idea behind the BAGUETTE algorithm is by upgrading *critical nodes* in the network to SDN switches with different types to mitigate attacks. If critical switches are adjacent to each other, they should use different switch types with the highest vulnerability *differentiation*, which prevents the attack from propagating. BAGUETTE follows three steps to solve the problem as follows.

(1) Preparing critical network nodes. BAGUETTE identifies critical network nodes based on their degrees and stores the critical network nodes in an array in the order of each node should be processed. This is achieved by sorting the network nodes based on their degrees in descendent order.

(2) Preparing switch type candidates. One target of the SCESU problem is to minimize the upgrade cost. To this end, BAGUETTE sorts the types based on their costs from low to high, which ensures cheaper types can be prioritized considered in the mapping procedure.

(3) Mapping switch types. BAGUETTE repeatably picks a critical node from the sorted critical network node array and selects a switch type for it until the whole network satisfies the security requirement. BAGUETTE maintains a globally current minimum security mapping variable and compares its compromising expectation with each new mapping's expectation in the process. This is helpful to get the minimum compromise expectation of all tested mappings when BAGUETTE cannot satisfy the security requirement. The switch type selection procedure has the following six subroutines. i) Get all types of the current selected node's adjacent nodes. ii) Calculate a new array of switch types, by removing the adjacent switch types, represented as $\mathcal{S}'$. iii) Get the common vulnerabilities of the adjacent switches as $\mathcal{F}'$. iv) For each type s_j in $\mathcal{S}'$, calculate the *vulnerability variation* that is the percentage of

Algorithm 1: The BAGUETTE algorithm.

Input: $\mathcal{G}$, $\mathcal{S}$
Output: X

```
/* Preparing the critical switch
   candidates                                   */
```
1 $N \leftarrow |\mathcal{V}|$;
2 $Matrix \leftarrow$ `getAdjacencyMatrix(`$\mathcal{G}$`);`
3 $D \leftarrow \emptyset$;
4 $X_{min} \leftarrow X$, $expect_{min} \leftarrow \infty$;
5 **for** $i \leftarrow 1$ **to** N **do**
6 $\quad D \leftarrow D \cup (\sum_{i_1}^{N} Matrix_{i,i_1})$;
7 Generating vector $\mathcal{V}' = \{v_i, i \in [1, N]\}$ by sorting the degree of each switch D_i in descending order;
```
/* Preparing switch types                       */
```
8 $C \leftarrow$`getSwitchCost(`$\mathcal{V}$`);`
9 Generating vector $\mathcal{S}' = \{s_j, j \in [1, |\mathcal{S}|]\}$ by sorting the cost of each switch type C_j in ascending order;
```
/* Mapping switch types                         */
```
10 **for** $v_i \in \mathcal{V}''$ **do**
11 $\quad Adj \leftarrow$ `getAdjacent(`v_i`);`
12 $\quad \mathcal{F}' \leftarrow \emptyset$, $S_1 \leftarrow \emptyset$;
13 $\quad$ **for** $v_{i_1} \in Adj$ **do**
14 $\quad\quad \mathcal{F}' \leftarrow \mathcal{F}' \cup$ `getVulnerability(`v_{i_1}`);`
15 $\quad\quad S_1 \leftarrow S_1 \cup$ `getSwitchType(`v_{i_1}`);`
16 $\quad \mathcal{S}'' \leftarrow \mathcal{S}' \setminus S_1$;
17 $\quad min\_ratio \leftarrow 1$, $j_{min} \leftarrow 0$;
18 $\quad$ **for** $s_j \in \mathcal{S}''$ **do**
19 $\quad\quad ratio \leftarrow \frac{\mathcal{F}_j \cap \mathcal{F}'}{\mathcal{F}'}$;
20 $\quad\quad$ **if** $min\_ratio > ratio$ **then**
21 $\quad\quad\quad min\_ratio \leftarrow ratio$, $j_{min} \leftarrow j$;
22 $\quad x_{i,j_{min}} \leftarrow 1$;
23 $\quad expect \leftarrow$`compromiseExpectation(`X`);`
24 $\quad$ **if** $expect \leq E^{max}$ **then**
25 $\quad\quad$ **break**;
26 $\quad$ **else**
27 $\quad\quad$ **if** $expect_{min} > expect$ **then**
28 $\quad\quad\quad X_{min} \leftarrow X$;
29 $\quad\quad\quad expect_{min} \leftarrow expect$;
30 **return** X_{min}

the number of common vulnerabilities between $\mathcal{F}_j$ and $\mathcal{F}'$, over the number of vulnerabilities of $\mathcal{F}_j$. v) Map the type with the lowest vulnerability variation calculated in Step iv) to the critical node. vi) Calculate the compromising expectation of the network, if the value satisfies the requirement, stop; otherwise, pick the next critical switch and go to Step i).

The BAGUETTE algorithm is detailed in Algorithm 1. Lines 1-7 generate the order of switch to be upgraded to SDN switches. In Lines 2-6, it first generates the degree vector D based on the adjacency matrix of the network, and it then sorts the nodes based on the degree in descending order. Lines 8-9 prepare the different switch types by sorting the types based on the cost of each type. We use quicksort to sort the types' costs. Lines 10-29 map types to switches. The key idea is to choose the type that has the biggest vulnerability differentiation of all adjacent switches. The cost of the upgrade is considered when multiple type candidates have the same vulnerability variation, and we choose the one with a smaller cost. Lines 11-15 get all the types and vulnerabilities of adjacent switches. Lines 16-21 calculate the ratio of common vulnerabilities of current type and adjacent switches' types to the vulnerabilities of adjacent switches' types. We choose the type with the minimum ratio. Finally, we calculate the compromise expectation based on the current mapping X. If the mapping satisfies the requirement, the algorithm stops; otherwise, it keeps mapping the next switch until all candidates are processed.

V. SIMULATION RESULTS

This section presents the simulation of BAGUETTE. First, we introduce the simulation setup information. Then, we present the comparison algorithms. Finally, we compare the performances of different algorithms under various real-world topologies. We find our BAGUETTE algorithm performs a near-optimal performance under non-full mesh toplogies.

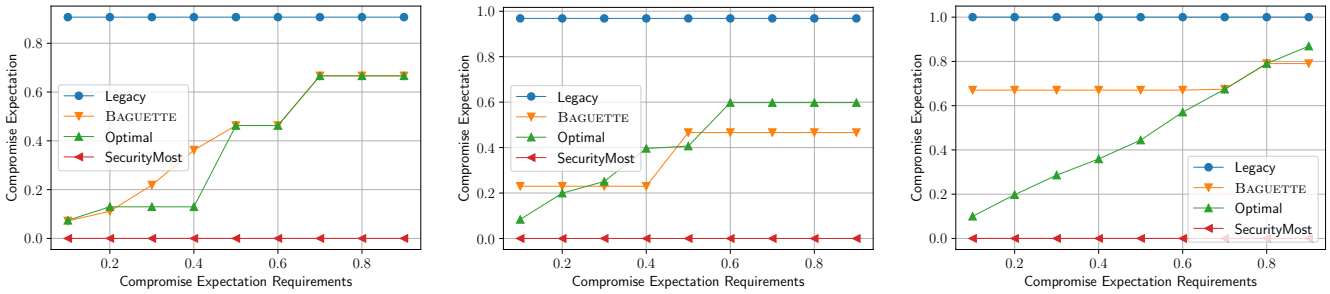
A. Simulation Setup

We use Arpanet [17], GlobalCenter [18], and HEAnet [19] from Topology Zoo [20] to conduct the simulation. Topology Zoo is a collection of 262 real-world backbone network topologies, and each topology is provided with a `gml` file. We use a popular python graph library `python-igraph` [21] to read `gml` files. Arpanet and GlobalCenter both have 9 nodes, and GlobalCenter is a full-meshed network with 36 links. HEAnet has 7 nodes. In the simulation, each node in the topology has only one switch, it can be either a traditional switch or an SDN switch. There are 4 types in total. We treat the traditional switch as a special switch type s_1. Each switch type contains multiple vulnerabilities in the simulation, we randomly generate vulnerabilities for each switch type, and the number of vulnerabilities is random in the range of $(0, 6)$. Besides, we have surveyed many SDN switches from [16], and the prices of popular SDN switches range from \$1000 to \$5000. Thus, we randomly generate a cost for every switch type in the range of $(1000, 5000)$. We generate attacks by calculating the subset of total vulnerability set $\mathcal{F}$, and the total number of attacks is $2^6 - 1$ (empty set is removed). We use Python to implement the simulation.

B. Compared Algorithms

We compare the following algorithms.

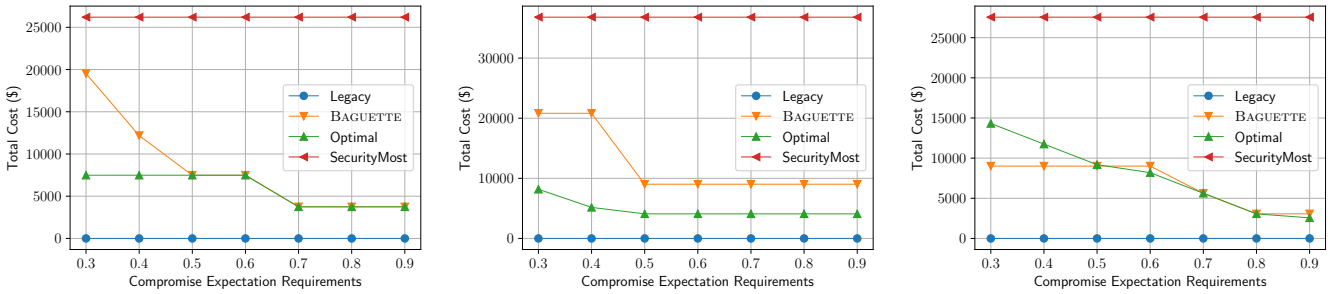
- Legacy: this is the default switch type deployment where all network nodes are deployed with legacy switches.
- Optimal: this is the optimal solution of the SCESU problem which minimizes the overall switch upgrade cost under certain network security requirements. Since Equation 2 requires graph traversal. Common ILP solvers like GUROBI [22] cannot help. We use brutal force method to solve the problem.



(a) The security comparison of 4 algorithms under Heanet (non-full mesh topology). The lower the better.

(b) The security comparison of 4 algorithms under Arpanet (non-full mesh topology). The lower the better.

(c) The security comparison of 4 algorithms under GlobalCenter (full mesh topology). The lower the better.



(d) The cost comparison of 4 algorithms under Heanet (non-full mesh topology). The lower the better.

(e) The cost comparison of 4 algorithms under Arpanet (non-full mesh topology). The lower the better.

(f) The cost comparison of 4 algorithms under GlobalCenter (full mesh topology). The lower the better.

Fig. 3: Legacy, Optimal, BAGUETTE, and SecurityMost performances under different security requirements. BAGUETTE achieves near-optimal performance under non-full mesh topologies.

- SecurityMost: this solution calculates the switch upgrade scheme with the minimum compromise expectation. We also use brutal force method to solve it due to the aforementioned reason.
- BAGUETTE: this algorithm is shown in Algorithm 1.

C. Simulation Results

We compare the security and overall cost performances of Legacy, Optimal, BAGUETTE, and SecurityMost algorithms under different security (compromise expectation) requirements ranging from 0.1 to 0.9. Due to the complexity of Optimal and SecurityMost, we do not use larger topology since they are time-consuming in large topologies. While BAGUETTE is tolerable to all scale of topologies because its time complexity is a polynomial.

Figure 3 shows the security and cost performances of the 4 algorithms under the 3 topologies. In a nutshell, BAGUETTE achieves up to **12.6x security enhancement** compared with the legacy setup and **as low as 11.1% cost of SecurityMost**. Legacy has no security guarantee, and SecurityMost can stop all attacks while introduces an exorbitant cost. In Figures 3a, 3b, and 3c, Optimal satisfies all the security requirements in each tested topology. **BAGUETTE satisfies 100% tests on HEAnet and approximately 80% tests on Arpanet**. However, it does not perform well on GlobalCenter when the security requirement is strict (low compromise expectation). Fig. 3c shows that BAGUETTE fails to satisfy the security requirement when the required compromise expectation is below 0.7. This is because BAGUETTE prefers to choose nodes with the most number of degrees to upgrade, but GlobalCenter network is a full-meshed network, and the number of degrees of each node is the same. To this end, BAGUETTE can only sequentially upgrade nodes one by one until all 3 SDN switch types are used.

To better understand the performance between Optimal and BAGUETTE, we define *Performance Likelihood* (PL) as the absolute difference between two algorithms over the difference of the minimum and maximum performances.

$$\text{PL} = \frac{|p_{A_1} - p_{A_2}|}{|p_{\max} - p_{\min}|}, \tag{7}$$

where, p_{A_1} and p_{A_2} are the performances of algorithms A_1 and A_2, and $p_{\max}$ and $p_{\min}$ are the maximum and minimum performances. The performances are retrieved at the same x-scale value in a experimental result figure. For example, the security PL of BAGUETTE and Optimal is represented as the absolute difference of compromise expectations of BAGUETTE and Optimal over the absolute difference of compromise expectations of Legacy and SecurityMost. A PL is a floating number between 0 and 1. If the performances are similar, the value approaches to 0.

Fig. 3 shows that the average security PL is 0.12, and BAGUETTE and Optimal have the same security performance in 46.7% of all the tests. In Figures 3e, 3f, and 3d, the cost PL between Optimal and BAGUETTE is 0.11, and BAGUETTE and Optimal have the same cost performance in 42.8% of all the tests. We can conclude that **BAGUETTE achieves near-optimal performance** under non-full mesh topologies.

VI. Related Works

SDN switch upgrade has attracted a plethora of research studies in recent years, too numerous to be discussed fully here, and thus we will focus on those that are most relevant. Jin et al. [9] propose a unified SDN controller that exerts SDN-like, fine-grained path control over both SDN and legacy switches. Hong et al. [11] present a systematic incremental SDN deployment methodology and hybrid operation model to decide which device to upgrade to SDN and how legacy and SDN devices cooperate in the hybrid environment. Guo et al. [23] propose a novel incremental SDN switch upgrade scheme, which aims to lower the latency of controller processes and the propagation delay of flow requests due to improper multi-controller deployment. Jia et al. [12] advocate considering how to maximize the network control ability with a given upgrading budget constraint, and minimize the upgrading cost to achieve the best network control ability. Levin et al. [10] propose an optimization framework to determine the partial SDN upgrade and assumes that at least one SDN switch is traversed by each network flow. However, none of the aforementioned works considers the security impact when upgrading switches to SDN switches. BAGUETTE satisfies the security requirement by mitigating the attack at the very initial node after entering the network and achieves the minimum overall cost with smartly selected switch types.

VII. Conclusion and Future Works

In this paper, we have identified the SCESU problem in the hybrid SDN deployment, which aims to achieve the security requirement with a minimum cost. We have proposed the BAGUETTE algorithm, which solves the problem by judiciously choosing critical switches and upgrading them with selected SDN devices to mitigate the attack propagations. We have conducted simulations with real-world topologies, and experimental results have shown that BAGUETTE achieves a near-optimal performance under non-full mesh topologies with up to 12.6x security enhancement and down to 11.1% cost of the most secure algorithm. By presenting BAGUETTE, we hope it can bring benefits to both industry and academia world when deploying SDN and inspire researchers to take advantage of the hybrid SDN. We will further improve BAGUETTE to achieve better performance on full-mesh topologies and spine-leaf topologies whose nodes have the same degree.

References

[1] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, Jan 2015.

[2] B. Hayes, "Cloud computing," *Communications of the ACM*, vol. 51, no. 7, pp. 9–11, 2008.

[3] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[4] F. Boccardi, R. W. Heath, A. Lozano, T. L. Marzetta, and P. Popovski, "Five disruptive technology directions for 5g," *IEEE communications magazine*, vol. 52, no. 2, pp. 74–80, 2014.

[5] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. ACM, 2012, pp. 13–16.

[6] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, and A. Vahdat, "B4: Experience with a globally-deployed software defined wan," in *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, ser. SIGCOMM '13. New York, NY, USA: ACM, 2013, pp. 3–14.

[7] S. Choi, B. Burkov, A. Eckert, T. Fang, S. Kazemkhani, R. Sherwood, Y. Zhang, and H. Zeng, "Fboss: Building switch software at scale," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '18. New York, NY, USA: ACM, 2018, pp. 342–356.

[8] AT&T, "First in the U.S. to Mobile 5G – What's Next? Defining AT&T's Network Path in 2019 and Beyond," *https://about.att.com/story/2019/2019_and_beyond.html*, 2019.

[9] C. Jin, C. Lumezanu, Q. Xu, H. Mekky, Z.-L. Zhang, and G. Jiang, "Magneto: Unified fine-grained path control in legacy and openflow hybrid networks," in *Proceedings of the Symposium on SDN Research*, ser. SOSR '17. Santa Clara, CA, USA: ACM, 2017, pp. 75–87. [Online]. Available: http://doi.acm.org/10.1145/3050220.3050229

[10] D. Levin, M. Canini, S. Schmid, F. Schaffert, and A. Feldmann, "Panopticon: Reaping the benefits of incremental SDN deployment in enterprise networks," in *2014 USENIX Annual Technical Conference (USENIX ATC 14)*. Philadelphia, PA: USENIX Association, Jun. 2014, pp. 333–345.

[11] D. K. Hong, Y. Ma, S. Banerjee, and Z. M. Mao, "Incremental deployment of sdn in hybrid enterprise and isp networks," in *Proceedings of the Symposium on SDN Research*, ser. SOSR '16. Santa Clara, CA, USA: ACM, 2016, pp. 1:1–1:7.

[12] X. Jia, Y. Jiang, and Z. Guo, "Incremental switch deployment for hybrid software-defined networks," in *2016 IEEE 41st Conference on Local Computer Networks (LCN)*, Nov 2016, pp. 571–574.

[13] K. Poularakis, G. Iosifidis, G. Smaragdakis, and L. Tassiulas, "Optimizing gradual sdn upgrades in isp networks," *IEEE/ACM Transactions on Networking (TON)*, vol. 27, no. 1, pp. 288–301, 2019.

[14] Z. Guo, W. Feng, S. Liu, W. Jiang, Y. Xu, and Z. Zhang, "Retroflow: maintaining control resiliency and flow programmability for software-defined wans," in *Proceedings of the International Symposium on Quality of Service, IWQoS 2019, Phoenix, AZ, USA, June 24-25, 2019.*, 2019, pp. 1:1–1:10.

[15] P. J. Schweitzer and S. S. Lam, "Buffer overflow in a store-and-forward network node," *IBM Journal of Research and Development*, vol. 20, no. 6, pp. 542–550, 1976.

[16] "Router-Switch.com, leading network hardware supplier." *https://www.router-switch.com*, accessed: 2019-08-20.

[17] Topology Zoo, "Arpanet 1970-6," *http://topology-zoo.org/maps/Arpanet19706.jpg*, 2019.

[18] The Center, LLC, "The Global Center for Nonprofit Excellence," *https://www.theglobalcenter.net*, 2019.

[19] HEAnet, "HEAnet - Ireland's National Research & Education Network," *https://www.heanet.ie*, 2019.

[20] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *IEEE Journal on Selected Areas in Communications (JSAC)*, vol. 29, no. 9, pp. 1765–1775, October 2011.

[21] G. Csardi, T. Nepusz *et al.*, "The igraph software package for complex network research," *InterJournal, Complex Systems*, vol. 1695, no. 5, pp. 1–9, 2006.

[22] Gurobi, "Gurobi optimizer," *Gurobi: http://www.gurobi.com*, 2019.

[23] Z. Guo, W. Chen, Y. Liu, Y. Xu, and Z. Zhang, "Joint switch upgrade and controller deployment in hybrid software-defined networks," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 5, pp. 1012–1028, May 2019.