Detection of Gray Hole Attack in Software Defined Networks

Yi-Ting Hsieh

Cheng-Yuan Ku

Hsieh, Y.T. & Ku, C.Y. (2018). Detection of gray hole attack in software defined network. In *Proceedings of The 18th International Conference on Electronic Business* (pp. 231-239). ICEB, Guilin, China, December 2-6.

Detection of Gray Hole Attack in Software Defined Networks

(*Full Paper*)

Yi-Ting Hsieh, National Chiao Tung University, Taiwan, andy19948@gmail.com
Cheng-Yuan Ku*, National Chiao Tung University, Taiwan, cooper.c.y.ku@gmail.com

ABSTRACT

Gray Hole Attack is an advanced transformation of black hole attack. Both of them are a common type of attack in Wireless Sensor Network (WSN). Malicious nodes may constantly or randomly drop packets and therefore reduce the efficiency of the networking system. Furthermore Software Define Network (SDN) has been highly developed in recent years. In this type of networks switch/router functionality is separated into the control plane and data plane. Network managers can select control policies and build operating rules according to their own preferences. In addition, network protocols and packet fields are also programmable. Because the switch/router only implements the data transmission and executes the switching/routing decisions based on commends coming from control plane. Compromised switches/routers themselves or malicious control instructions both can result in selectively dropped packets. This makes a gray hole attack possible in the infrastructure of SDN. Therefore, this paper would like to discuss time-base and random-base gray hole attack in SDN, and then propose a useful detection method based on weighted K-Nearest Neighbor (KNN) and Genetic Algorithm (GA). The simulation data collected from switches/routers indicate that our method does demonstrate pretty good performance.

Keywords: Gray hole attack, SDN, weighted KNN, GA, detection method.

*Corresponding author

INTRODUCTION

Software Define Network (SDN) decouples the control function from traditional data forwarding devices. The switches/routers become simpler because they separate the control unit and _functionality to controller. The controller can easily manage the network resource and traffic by sending the instruction to the switches/routers (data plane). The switches/routers can only focus on the packets transmission. The switches/routers typically ask the controller to obtain flow rules when the data plane receives new packets. The flow rules are stored in the Flow Table. Hence every decision and calculation are made by the controller. The centralized architecture can make it easier for the network manager to collect network information and implement the policies. Due to the fast growth of SDN, nowadays, it faces many types of threats. Mostly the purpose of these attacks may crash the network or reduce the network efficiency. Gray Hole Attack, also named selective forwarding attack, is one of the common attacks identified in the wireless networks, including mobile ad hoc network (MANET), WSN, etc. It is a variation of Black Hole Attack which has some compromised switches/routers swallow all of the received packets (Shanmuganathan & Anand, 2012). Gray Hole Attack is more difficult to detect due to its unpredictable behavior changing between the normal device and the black hole. Many prior research results regarding Gray Hole Attack focused on wireless networks. However, we believe it may also cause huge losses if initiated in SDN.

Machine learning algorithms have been very popularly used in recent years. The application of big data analytics to mitigate security attack problem has become more and more attractive (Cui et al., 2016). Because of the rapid growth of calculation capacity, modern computers or server farm can handle a lot more calculation jobs than ever before. As a consequence, the separation of data plane and control plane becomes more feasible for controller to collect huge data and then implement the switching/routing decision making and execute the machine learning algorithms to detect malicious attack as well.

In this paper, we study the Gray Hole Attack in SDN and then design an intelligent method to detect it. This method including three phases combines the weighted KNN and GA to recognize the abnormal switched/routers. The simulation results support the effectiveness and efficiency of our method.

LITERATURE REVIEW

Review of Software Defined Network

McKeown et al. (2008) had proposed the first concept of Software Define Network, the OpenFlow. OpenFlow switch consists of three parts: (1) Flow Table, (2) Secure Channel, (3) OpenFlow Protocol. The Flow Table is composed of many flow entries to inform the switch how to process the flow. The Secure Channel connects the switch to the controller. It allows transmitting instruction and packets between switch and controller. OpenFlow Protocol provides a standard for controller to communication with switches.

Flow Table plays an important role in OpenFlow switch. It manages the packets transmissions. When the switch receives a packet, it checks the flow table first. If the packet header matches the rule in the table, the switch will do the action as indicated in the rule. If the packet header doesn't match any rules in the tables (called table miss), the switch will check table-miss table. The actions in table-miss table include (1) send the packet to the controller (called packet-in), (2) forward to all interface, (3) drop the packet. After receiving the packet-in packet, the controller checks the network topology and add new rule about the packet in the switch. Next time the switch receives a packet with the same destination header, it will know how to handle this packet. Actually there are three parts in the Flow Table, i.e. packet header, action, and counter.

Gray Hole Attack

The gray hole attack is an extension of black hole attack in which a malicious node's behavior is exceptionally unpredictable (Jhaveri et al., 2012). It typically can be categorized into three types (Jhaveri et al., 2010). The first one is that a node may behave like black hole in a short time, but later on it acts like a normal node in the network. We call this type 'Gray hole attack'. It changes its status based on some situation and the drops the receiving packets accordingly. The time-based gray hole is that a node changes its status in regular period. For example, acts like a normal node in the first five seconds, and acts like a black hole node in the next five seconds and so on. The random-based gray hole is that it changes its status for no reason, usually by probability. For condition-based gray hole, it changes its status if some specific conditions happen. For example, the network speed reaches a specified value and then the switching node starts to drop packets. In the second type, a malicious node may drop packets in fixed percentage or drop packets of a specified data flow and forward all other packets. We call it selective forwarding attack (Bysani & Turuk, 2011). For this type, we classify them by defining how they drop packets, i.e. they drop 70% of all the received packets or 50% of flow 2 in queue. The third type is the hybrid of the previous two types and we can also name it as the selective forwarding gray hole. In this way, a malicious switch/router may drop packets randomly in the gray hole period.

We may consider that the gray hole is a kind of passive attack in general. It does not completely interrupt the normal operation of the network (Wazid et al., 2011). However, the black hole could be an active attack. The black hole attracts the flow by modifying the routing table. Although the gray hole attack is an extension of the black hole attack and it only drops packets without attracting the data flow, the gray hole attack is more difficult to detect than the black hole attack. That is because the malicious black hole switch /router drops the received packets with certainty (Hongmei et al., 2002).

Gray Hole Attack in SDN

Dhawan et al. (2015) proposed SPHINX: Detecting Security Attacks in Software-Define Networks. They presented a detect system to defend serval attacks in Software Define Network. One of these attacks is that there may be a black hole switch in the data plane. A malicious switch in the flow may drop or siphon off packets. They implemented it by the 'action' attribute of the OpenFlow. The 'action' of drop can make the switch drops specified packets. When it drops all packets from the in-port, a black hole switch occurs. Since the black hole switch may exist in the Software Define Network, the gray hole switch also may exist in the Software Define Network because a gray hole is the variation of black hole.

PROPOSED DESIGN

Our method is inspired by those results published by Sen et al. (2007). They proposed an operating architecture to detect the gray hole attack and the cooperative black hole attack in the Wireless Sensor Network. The detection approach is divided into three phases, i.e. anomaly detection phase, gray hole attack test and confirmation phase, and warming phase. The traffic monitor of SDN collects the traffic data and flow parameters of each switch/router constantly. After collecting these data and parameters, the controller uses the weighted KNN with GA to classify whether each switch/router is abnormal or not and then builds the suspicious list for the further test and confirmation in the next phase. In order to achieve the higher accuracy of detection, we should retest all the switches/routers in the suspicious list until we believe that their statuses are confirmed as clear or guilty. In this architecture, we may send test packets to go through some switches/routers in the suspicious list non-periodically. If the evidence is strong enough, the system will alert the manager and controller that which one should be a gray hole switch/router in the network.

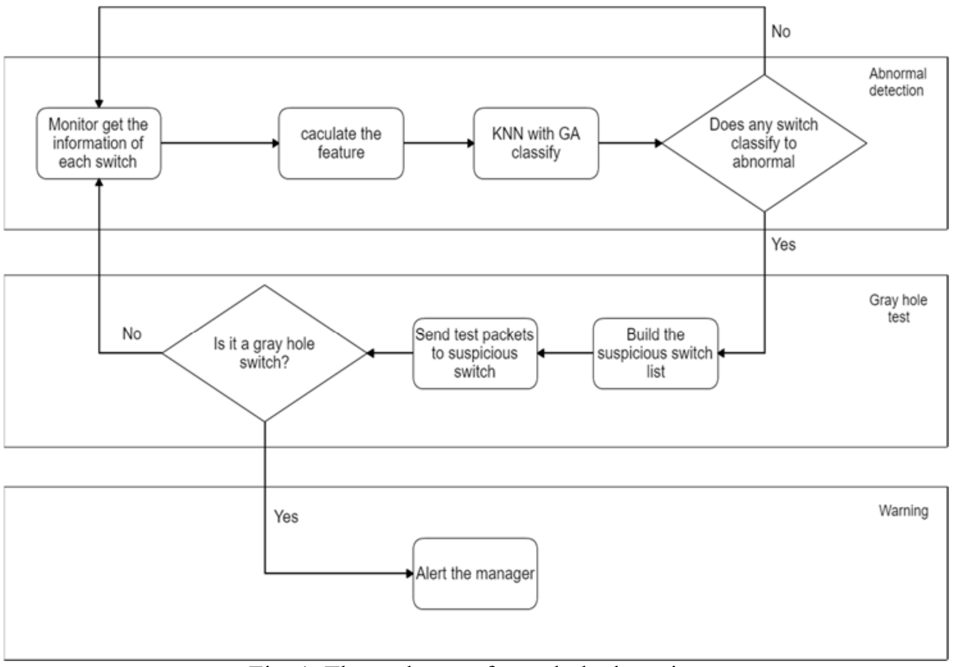
Abnormal detection
Monitor get the information of each switch
caculate the feature
KNN with GA classify
Does any switch classify to abnormal
No
Yes
Gray hole test
Build the suspicious switch list
Send test packets to suspicious switch
Is it a gray hole switch?
No
Yes
Warning
Alert the manager

Fig. 1. Three phases of gray hole detection

In the abnormal detection stage, the monitor evaluates each switch/router using weighted KNN with GA in which the weights of features of KNN are calculated according to GA since it is very difficult to decide which attribute is more important than the other in various situations and what number should be assigned as weight. Fig. 2 demonstrates the flow char of Genetic Algorithm that generates the weight set.

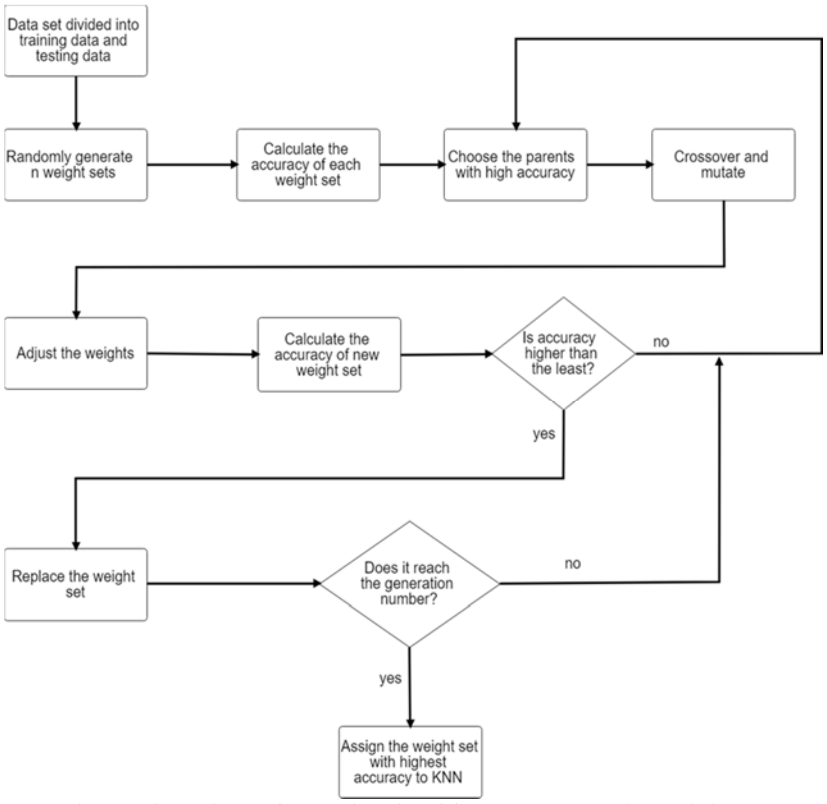
Data set divided into training data and testing data
Randomly generate n weight sets
Calculate the accuracy of each weight set
Choose the parents with high accuracy
Crossover and mutate
Adjust the weights
Calculate the accuracy of new weight set
Is accuracy higher than the least?
no
yes
Replace the weight set
Does it reach the generation number?
no
yes
Assign the weight set with highest accuracy to KNN

Fig. 2. Flow chart of Genetic Algorithm to generate the weight set

Anomaly Detection Phase

A gray hole drops the packet, and therefore decreases the networking efficiency (Tseng et al., 2011). In the anomaly detection phase, the SDN network monitor collects the detailed network information continuously. The monitor calculates the features of each switch/router, and these data will be used by the weighted K-Nearest Neighbor algorithm to classify them. If any switch/router is classified as the gray hole, the monitor should add it into the suspicious list for the next phase to implement the further test. The features we use in weighted KNN are listed in Table 1.

Table 1. Features in weighted KNN

Features	Description
pkt_rx	The number of packet received in the period
pkt_tx	The number of packet transferred in the period
speed	Bits transferred per second
change of speed	The difference of speeds
packet dropped	The number of packet dropped in the period

We choose weighted KNN features based on the transmission behavior of switch/router. The *pkt_rx* and *pkt_tx* are collected from the monitor module. The *speed* is calculated from *tx_byte* which is also collected from the monitor. The *change of speed* is calculated from the *speed* parameters. When the gray hole switch drops the packets, the *speed* of gray hole switch must drop to a low level. This should make the *change of speed* higher than a normal value. The *packet dropped* shows the number of packets dropped in some period. When there is a gray hole switch within the network, the *packet dropped* must be higher than a normal one. In the K-nearest neighbor algorithm, it usually calculates the distance and classifies items by using brute force concept. When the number of data and items we want to classify grows, it must do lots of calculation. Furthermore, the dimension of the features also affects the time complexity. Since the time complexity of K-nearest neighbor classification is O(D*n), where D is the dimension of feature and n is the number of data in data set, it is very hard to classify all the switches in a short period. To solve this problem, we use the ball tree data structure (Liu et al., 2006). The ball tree is a binary search tree. Every node defines a D-dimensional hypersphere. The concept is to encode the aggregate distance of the sample to reduce number of the calculation. The time complexity of ball tree in K-nearest neighbor is O(D*logn). Constructing the ball tree costs most of resource in CPU, but it is quite efficient to classify the dataset. With this data structure, we can construct the ball tree in the initial phase and the classification work can be done in time.

Gray Hole Attack Test and Confirmation Phase and Warming Phase

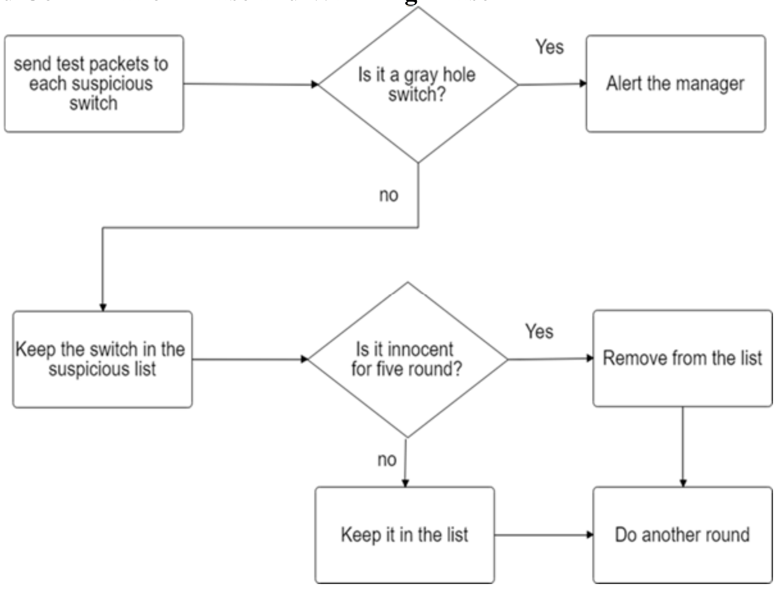
send test packets to each suspicious switch
Is it a gray hole switch?
Yes
Alert the manager
no
Keep the switch in the suspicious list
Is it innocent for five round?
Yes
Remove from the list
no
Keep it in the list
Do another round

Fig. 3. Flow chart of test and confirmation phase and warming phase

In the test and confirmation phase as shown in Fig. 3, the controller non-periodically sends test packets to go through suspicious switches/routers recorded in the suspicious list. These packets are filled with specific header. When one destination receives the packet, it sends back to controller immediately. If the controller doesn't receive the packet, then that switch/router may be gray hole with high probability. One switch/router will be identified as a gray hole after *n* consecutive losses of packets. If the controller receives all of *n* testing packets, then that switch/router should be released from the suspicious list. If not both, the monitor keeps the switches/routers in the suspicious list and waits for the results of next round test. Once a gray hole switch/router is detected, the controller alerts all the network by flooding this warming message as indicated in Fig. 3.

SIMULATION RESULTS AND DISCUSSION

Simulation Tools

- Mininet

Mininet (Oliveira et al., 2014; Mininet, 2018) is a network simulator. It runs hosts, switches, routers and links in Linux kernel. All the machines in Mininet are just like the machines in the real world, i.e. using ssh in it and running the programs. Sending packets in Mininet is also very much like sending in the real Ethernet interface. The user can write a program to input the speed and delay. The packets will get processed in the steps which are run in real Ethernet switches. All the virtual hosts, switches, links and controller are like the real things but created by software rather than hardware. Most importantly, its behavior is similar to hardware element in Ethernet. The user can create the necessary topologies, customized packet forwarding rules and the network application. Moreover, Mininet is widely used for simulating the Software Define Networks. Based on the above observation, we decide to use Mininet to create the network topology and simulate our proposed approach.

●Ryu controller

Ryu (Ryu controller, 2018) is one of the controllers used in the Software Define Network. The source code is available on the GitHub and is maintained by the open Ryu community in Japan. Ryu is a component-based SDN framework. It provides well-defined API for developers to create and manage the network easily. We decide to use Ryu controller in our simulation because it is written in Python, and we use scikit learn which is a famous machine learning library for Python, too.

●Scikit learn

Sciket learn (Sciket-learn, 2018) is a free software machine learning library for Python language. It contains many machine learning algorithms categorized as classification, regression, and clustering algorithms. It is designed to cooperate with the Python numerical and scientific libraries Numpy and Scipy. Every algorithm is designed in high efficiency, the time cost and memory cost are quite low. Since our monitor has to classify every switch/router in short period, it must finish the calculation of algorithms in time. That's why we use the KNN in scikit learn rather than create new one.

●D-ITG

D-ITG (Distributed internet traffic generator) is a traffic generator in internet (Botta et al., 2012; D-ITG, 2018). It supports many protocols, including TCP, UDP, ICMP, etc. It also supports both IPv4 and IPv6 traffic generation and it is capable to generate traffic on the network, transport, and application layers. Furthermore, the packet size and the distribution of generating traffic could be defined by the user. It makes the simulation environment closer to the real world situation. The distributions include uniform distribution, exponential distribution, normal distribution, Poisson distribution, and Pareto distribution. We use D-ITG to generate IPv4 traffic with UDP in our experiment.

Simulation Setup

● Topology

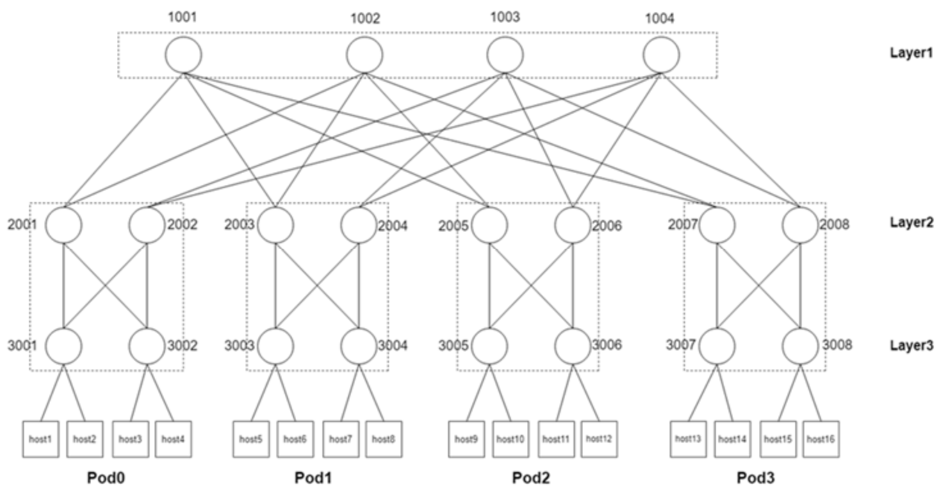

Fig. 4. Fat-tree topology

The fat-tree topology is a famous topology in Software Define Network. We choose this topology in our experiment because many studies use this topology to simulate their environments, too. Fat-tree topology is shown in Fig. 4. There are 20 switches and 16 hosts in the fat-tree topology. The 1001~1004 switches are the core switches based on definition and 2001~3008 are aggregation switches. Four switches compose a 'Pod', i.e. switch 2001, 2002, 3001 and 3002 compose Pod 0 as shown in Fig. 4. We define No. 1001~1004 switches as layer 1, No. 2001~2008 switches as layer 2 and No. 3001~3008 switches as layer 3. Each switch in layer 3 connects two hosts. We will discuss the detection rate and accuracy rate for gray hole switch in each layer.

● Traffic model

The concept of traffic model we used is commonly used in many experiments for Software Define Networks. Each host sends UDP packets to two hosts in every other Pods except its own Pod. The receiver is chosen randomly. Let's take the example in Fig. 4, host 1 may send six data flows to other six hosts, two in Pod 1, the other two in Pod 2, and the remaining two in Pod 3. Every host sends the data flow until the end of the experiment. D-ITG can generate packets based on many stochastic distributions. We use D-ITG to send UDP packets in normal distribution.

● Gray hole switch implementation

We use the attribute 'hard-timeout' in Flow Table to implement the gray hole switch in Software Define Network. The meaning of hard-timeout is that the flow rule will be eliminated after *n* seconds when this rule is created. We let the controller sends gray hole rule every *m* seconds and the hard-timeout of the rule is *n*. In this way, the gray hole switch drops packets for *n* seconds within every *m* period.

● The data set for KNN

We simulate the gray hole switch and then collect the data because there is no real data set available on Internet. In the simulation, we choose one switch as gray hole and run for 3 minutes. The monitor collects the features we need from each type of switch.

The simulation is run 60 times and each switch has been selected as the gray hole at least 2 times. We totally collect more than 150,000 sets of data for normal switches and gray hole switches. In the beginning of the experiment, we randomly choose 10000 data sets of normal switches and 10000 data sets of gray hole switches as the training data set for K-nearest neighbor.

- The K value

The K value for K-nearest neighbor is decided based on the accuracy rate of testing data. The initial data set contains 20000 data sets for each type of switches. The accuracy rate is the percentage of total testing data which are the same as the original ones. Each value are run 10 times and then we calculate the average accuracy rate. Finally we select K value with the highest accuracy rate. As shown in Fig. 5, 31 is the best K value for our experiment because the accuracy rate is nearly 75%.

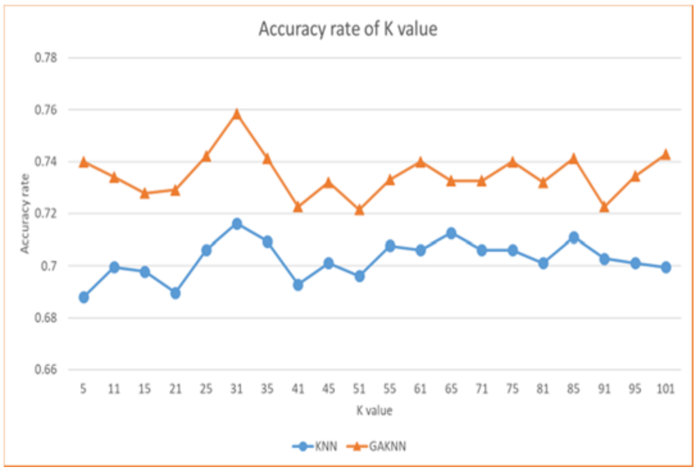

Fig. 5. Accuracy rate of K

Table 2. Experiment parameter

Emulation Parameter	Value
K value	31
Numbers of genes	20
Mutation rate	0.01
Generation number	30
Get topology period	20 sec
Initial time	90 sec
Detection time	180 sec
Total Emulation time	270 sec
Repeat times	Each layer for 100 times

Results of Time-based Gray Hole Attack

- Monitor period: 3 seconds

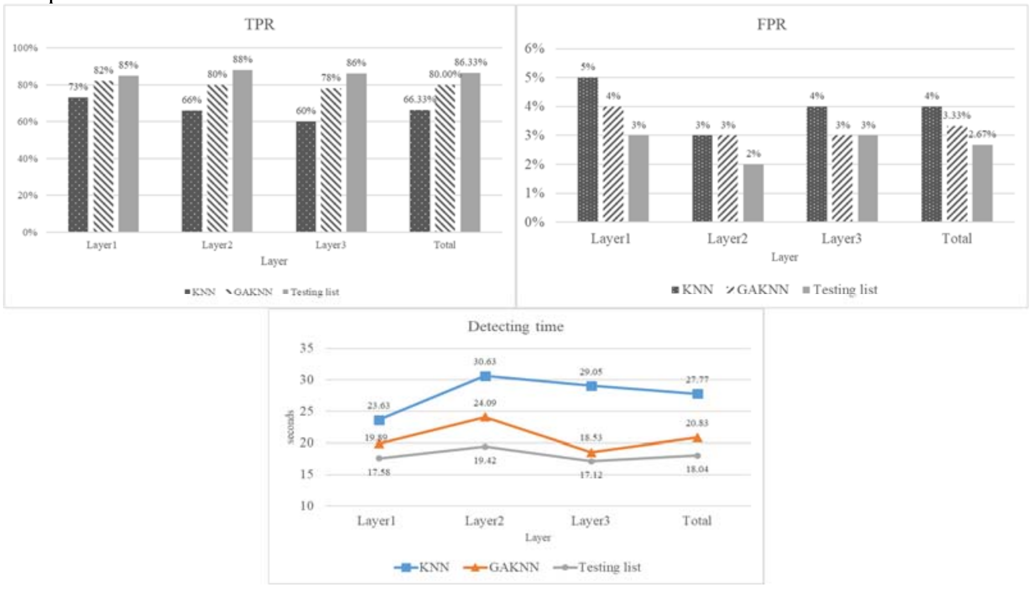

Fig. 6. Performance comparison for detection of time-based gray hole attack with shorter monitoring period

We can observe that our GAKNN (KNN with GA) outperforms the traditional KNN while detecting time-based gray hole attack with shorter monitoring period.

●Monitor period: 8 seconds

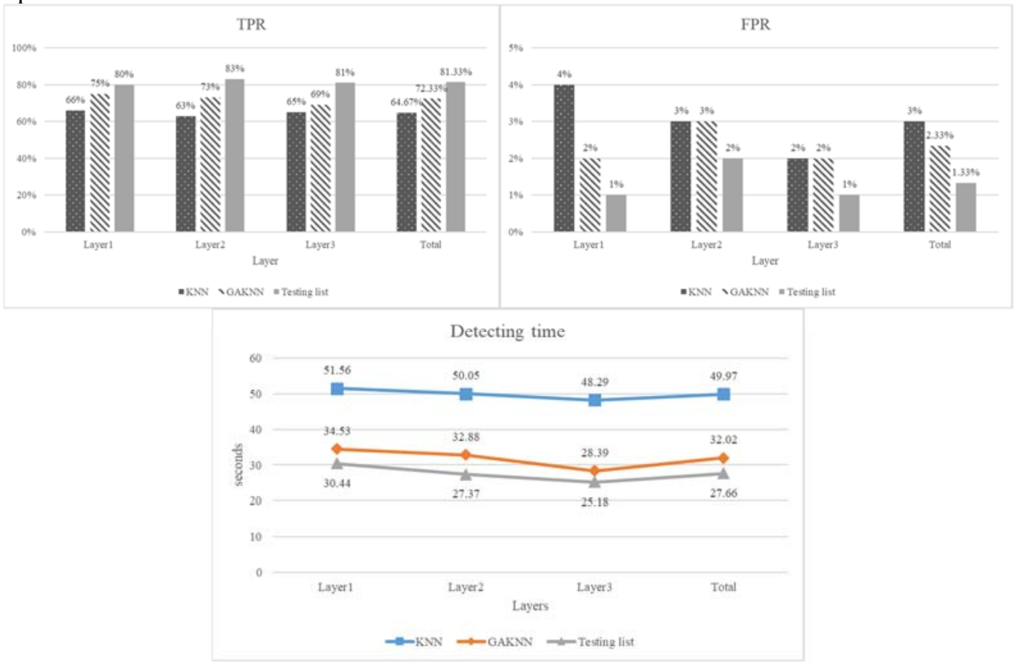

Fig. 7. Performance comparison for detection of time-based gray hole attack with longer monitoring period

We can observe that our GAKNN (KNN with GA) outperforms the traditional KNN while detecting time-based gray hole attack with longer monitoring period.

Results of Random-based Gray Hole Attack

●Monitor period: 3 seconds

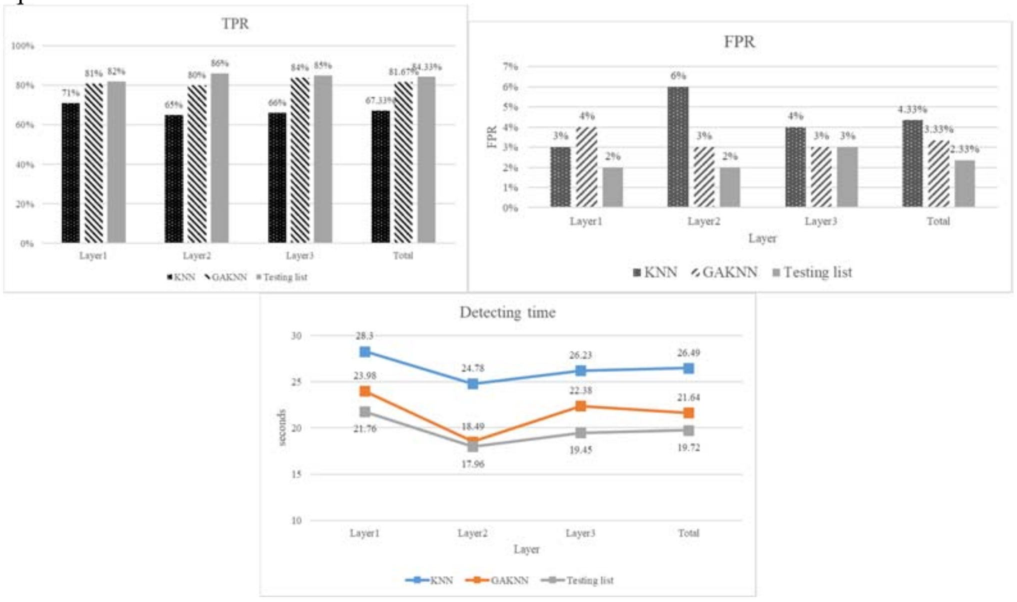

Fig. 8. Performance comparison for detection of random-based gray hole attack with shorter monitoring period

We can observe that our GAKNN (KNN with GA) outperforms the traditional KNN while detecting random-based gray hole attack with shorter monitoring period.

●Monitor period: 8 seconds

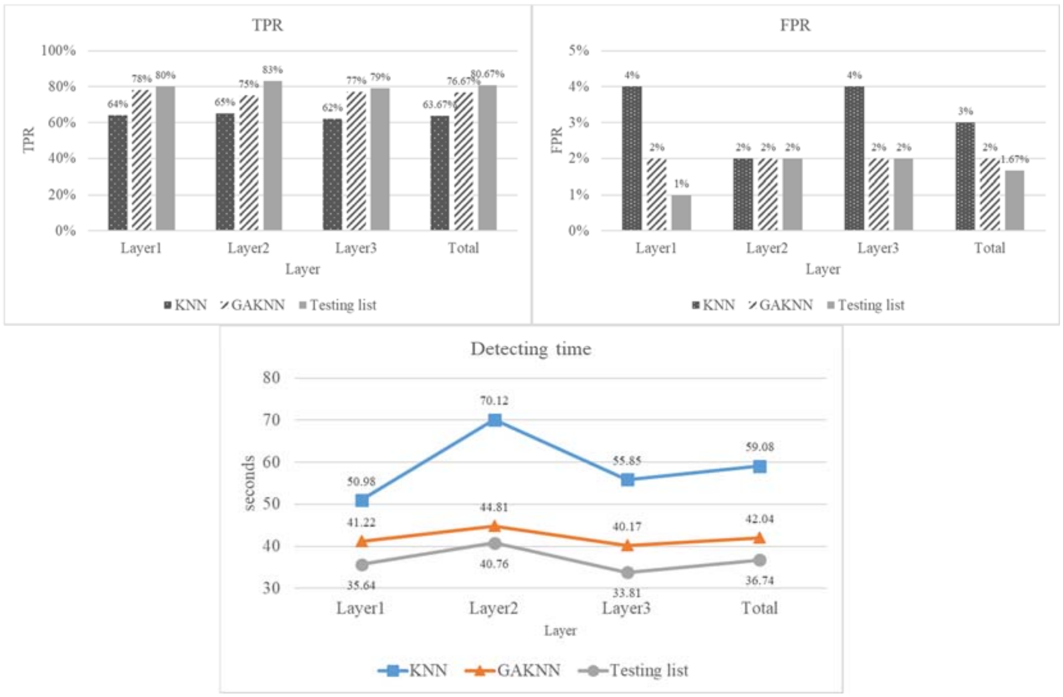

Fig. 9. Performance comparison for detection of random-based gray hole attack with longer monitoring period

We can observe that our GAKNN (KNN with GA) outperforms the traditional KNN while detecting random-based gray hole attack with longer monitoring period.

CONCLUSIONS

In this paper, we extend the gray hole attack study in Wireless Sensor Network to Software Define Network. Since the gray hole switch/router drops packets and reduces the efficiency of the Software Define Network, we propose a detection method and the corresponding architecture. The abnormal detecting phase is the key core to find the gray hole attack and GAKNN is suggested in this paper. The GAKNN classifies the switches/routers in a higher accuracy rate than the original KNN. Using GAKNN, the accuracy rate increases 10%~15%. The false positive rate reduces 1%~2%. The necessary detecting period reduces 2~4 rounds. In the future, we may study the complex type of gray hole attack. Furthermore, we should consider to simulate in a bigger topology to test the performance of our proposal. For the detection part, we can try other machine learning algorithms or statistic methods in the abnormal detection phase.

ACKNOWLEDGEMENT

This work is partially supported by Minster of Science and Technology in Taiwan, ROC under the project MOST 106-2410-H-009-025-MY3.

REFERENCES

[1] Botta, A., Dainotti, A., & Pescapé, A. (2012). A tool for the generation of realistic network workload for emerging networking scenarios. Computer Networks, 56, 3531-3547.
[2] Bysani, L.K., & Turuk, A.K. (2011). A survey on selective forwarding attack in wireless sensor networks. Proceedings of 2011 International Conference on Devices and Communications (ICDeCom), 1-5.
[3] Cui, L., Yu, F.R., & Yan, Q. (2016). When big data meets software-defined networking: SDN for big data and big data for SDN. IEEE Network, 30, 58-65.
[4] Dhawan, M., Poddar, R., Mahajan, K., & Mann, V. (2015). SPHINX: Detecting Security Attacks in Software-Defined Networks. Available: https://rishabhpoddar.com/publications/Sphinx.pdf.
[5] D-ITG (2018), Distributed internet traffic generator. Available: http://www.grid.unina.it/software/ITG/.
[6] Hongmei, D., Wei, L., & Agrawal, D.P. (2002). Routing security in wireless ad hoc networks. IEEE Communications Magazine, 40, 70-75.
[7] Jhaveri, R.H., Patel, A.D., Parmar, J.D., & Shah, B.I. (2010). MANET routing protocols and wormhole attack against AODV, International Journal of Computer Science and Network Security, 10, 12-18.
[8] Jhaveri, R.H., Patel, S.J., & Jinwala, D.C. (2012). DoS attacks in mobile Ad Hoc networks: A survey. Proceedings of 2012 Second International Conference on Advanced Computing & Communication Technologies, 535-541.
[9] Liu, T., Moore, A.W., & Gray, A. (2006). New algorithms for efficient high-dimensional nonparametric classification. Journal of Mach. Learn. Res., 7, 1135-1158.
[10] McKeown, N. et al. (2008). OpenFlow: enabling innovation in campus networks. SIGCOMM Comput. Commun. Rev., 38, 69-74.
[11] Mininet (2018). An Instane Virtual Network on your Laptop (or other PC). Available: http://mininet.org/.

[12] Oliveira, R.L.S.d., Schweitzer, C.M., Shinoda, A.A., & Rodrigues, P.L. (2014). Using Mininet for emulation and prototyping Software-Defined Networks. Proceedings of 2014 IEEE Colombian Conference on Communications and Computing (COLCOM), 1-6.

[13] Ryu controller (2018). Available: https://osrg.github.io/ryu/.

[14] Sciket-learn (2018). Machine learning in Python. Available: http://scikit-learn.org/stable/index.html.

[15] Sen, J., Chandra, M.G., Harihara, S.G., Reddy, H., & Balamuralidhar, P. (2007). A mechanism for detection of gray hole attack in mobile Ad Hoc networks. Proceedings of 6th International Conference on Information, Communications & Signal Processing, 1-5.

[16] Shanmuganathan, V. & Anand, T. (2012). A survey on gray hole attack in manet. International Journal of Computer Networks and Wireless Communications, 2, 647-650.

[17] Tseng, F.H., Chou, L.D., and Chao, H.C. (2011). A survey of black hole attacks in wireless mobile ad hoc networks. Human-centric Computing and Information Sciences, 1:4. Available: https://doi.org/10.1186/2192-1962-1-4.

[18] Wazid, M., Singh, R.K., & Goudar, R. H. (2011). A survey of attacks happened at different layers of mobile ad-hoc network & some available detection techniques. Proceedings of 2011 International Conference on Computer Communication and Networks.