A Fuzzy Logic-Based Information Security Management for Software-Defined Networks

Sergei Dotcenko[*], Andrei Vladyko[*], Ivan Letenko[*]
[*] The Bonch-Bruevich Saint-Petersburg State University of Telecommunications, Russian Federation
vicerector.sc@sut.ru, vladyko@bk.ru, letenko@gmail.com

Abstract - **In terms of network security, software-defined networks (SDN) offer researchers unprecedented control over network infrastructure and define a single point of control over the data flows routing of all network infrastructure. OpenFlow protocol is an embodiment of the software-defined networking paradigm. OpenFlow network security applications can implement more complex logic processing flows than their permission or prohibition. Such applications can implement logic to provide complex quarantine procedures, or redirect malicious network flows for their special treatment. Security detection and intrusion prevention algorithms can be implemented as OpenFlow security applications, however, their implementation is often more concise and effective. In this paper we considered the algorithm of the information security management system based on soft computing, and implemented a prototype of the intrusion detection system (IDS) for software-defined network, which consisting of statistic collection and processing module and decision-making module. These modules were implemented in the form of application for the Beacon controller in Java. Evaluation of the system was carried out on one of the main problems of network security - identification of hosts engaged in malicious network scanning. For evaluation of the modules work we used mininet environment, which provides rapid prototyping for OpenFlow network. The proposed algorithm combined with the decision making based on fuzzy rules has shown better results than the security algorithms used separately. In addition the number of code lines decreased by 20-30%, as well as the opportunity to easily integrate the various external modules and libraries, thus greatly simplifies the implementation of the algorithms and decision-making system.**

Keywords - **Information security; Software-Defined Networks; OpenFlow; Fuzzy Logic; Port scan;**

I. Introduction

Software-defined networks (SDN) differ from the traditional network infrastructure due to the substantial rethinking the relationship between data plane and control plane of the network device. For now, the OpenFlow (OF) is the mostly commonly used SDN communication protocol. For today's networks, which are increasingly facing the task of virtualization and transferable network applications, OpenFlow protocol can offer the flexibility needed for dynamic network management that goes beyond traditional networks. In OpenFlow control plane rules define the basic instructions for flows that specify forwarding, changing, or dropping packets that enter the OF-switch. The control level in the switch is simplified; the switch only sends statistics and applies to the new flow rules to the external OpenFlow controller.

OpenFlow controller contains the logic for defining, update and adapt of flow rules. The controller can communicate with multiple switches simultaneously; it can distribute a set of consistent flow rules through switches for direct routing or optimize tunneling, which can significantly improve the efficiency of traffic management. The controller also provides an application programming interface (API) to develop OpenFlow applications that implement the logic required to produce flow rules. In terms of network security OpenFlow offers researchers unprecedented control over network and defines a single point of control over the data flows routing of all network infrastructure. OpenFlow network security applications can implement more complex logic processing flows than their permission or prohibition. Such applications can implement logic to provide complex quarantine procedures, or redirect malicious network flows for their special treatment. Security detection and intrusion prevention algorithms can be implemented as OpenFlow security applications, however, their implementation is often more concise and effective.

This paper is arranged as follows. Section 2 introduces an algorithm for an information security management system based on fuzzy-logic. Section 3 explains the implementation of anomaly detection algorithms. Section 4 contains prototype architecture and related simulations. Section 5 gives conclusions and future work.

II. Development of SDN Security Algorithm

To develop complex information security management solution for software-defined networks we use, proposed in [1], algorithm for an information security management system based on soft computing. This system includes: assessment the security level of information systems, information security risk management and intrusion detection and prevention.

The algorithm and the actions performed on each of its steps are shown in Figure 1.

A. Aggregation of parameters and statistics collection

An important feature of the developed approach is possibility of statistics collection directly on the switch; this gives the opportunity to analyze the traffic and perform actions closer to the source of malicious activity or destination host. In this case, the aggregation algorithms must be adapted for use in embedded systems and have the following characteristics: to be undemanding to the processor and have very low and predictable memory requirements.

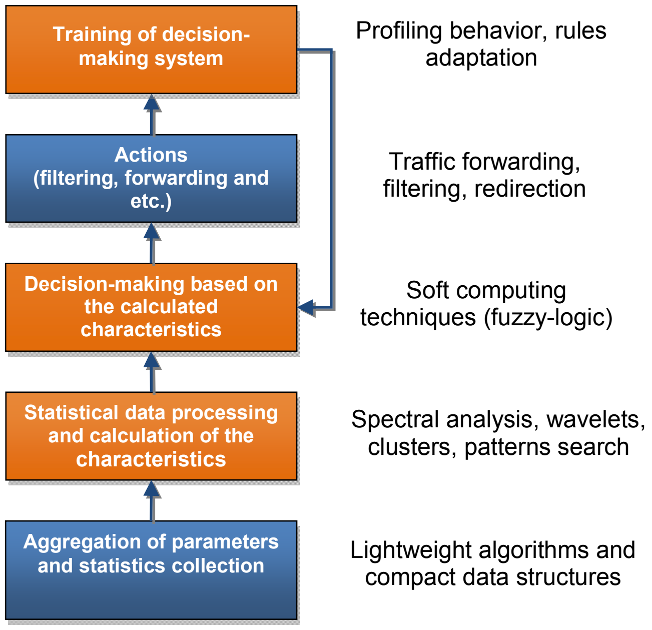
Training of decision-making system
Profiling behavior, rules adaptation
Actions (filtering, forwarding and etc.)
Traffic forwarding, filtering, redirection
Decision-making based on the calculated characteristics
Soft computing techniques (fuzzy-logic)
Statistical data processing and calculation of the characteristics
Spectral analysis, wavelets, clusters, patterns search
Aggregation of parameters and statistics collection
Lightweight algorithms and compact data structures

Figure 1. Proposed algorithm

B. Actions

The main possible actions based on decisions are: pass traffic, traffic filtering by specific signatures, active response or checking host. These actions are performed by the switch in accordance with the tables of flows specified by the controller. To implement the active response (interaction) with suspicious or attacking hosts the network can include the device, as discussed in [2]. Especially effective these devices can be used in the SDN in consequence of capability dynamically reroute traffic. Such devices can effectively detect attacks on web applications, prevent overflow of the application resources, and reduce the intensity of the attacks on denial of service.

C. Adapted network operating system software interface

A network operating system of SDN controller should provide a specialized set of functions for efficient network security application operation.

Firstly, a set of functions to work with the flow rules [3]:

- Identifying the source of the rules (for example, an administrator, a regular application or the security application), and provide a method for signing rules;
- Detection of conflicts between rules, for example between the rules issued by the various applications;
- Conflict resolution based on the priorities of the sources of the rules and their signatures;

Secondly, provide functions to simplify the collection and analysis of the traffic statistics, for example, information about the state of the TCP - session. And also features to simplify development of rule sets for traffic filtering and redirecting.

D. Statistical data processing and calculation of the characteristics

For processing collected statistical data is proposed to use time-series analysis methods like wavelet, spectrum analysis and etc. These methods facilitate analysis of time-frequency traffic characteristics, and have proved to be effective at detecting volume anomalies. Another advantage of time-series analysis methods is that they are well studied, comparatively computationally simple, and there are many examples of their implementation.

E. Decision-making based on the calculated characteristics

The main objective of the information security management system is to detect malicious activity on the basis of a set of input variables. Usually, obtained characteristics have a simple distribution and can be well approximated by fuzzy membership-functions, so the fuzzy logic decision-making is well suited for this type of problem.

F. Training of decision-making system

Since the information security management system operates in a constantly changing environment, it is required adaptation of the algorithms, and training of the decision-making systems for a current situation.

Our implementation of the training subsystem includes short-term and long-term learning modules. Typically, short-term learning module is implemented directly in the controller. Depending on the expected performance of the controller, long-term learning module can be implemented either directly in the controller, or in an external device, for example, in the management server in which the database is located.

Statistical data processing, decision-making and training algorithms are software level tasks and can be implemented as applications for the network operating system or as modules for specialized framework.

Consider how the algorithm may be implemented in software-defined networks. Figure 2 shows the logic diagram (architecture) of a protected SDN.

III. Anomaly Detection Algorithms

In this work we constructed a prototype implementation of the proposed architecture of protected SDN that includes statistic collection and processing module and decision-making module. These modules have been implemented as applications for the Beacon controller in Java.

Evaluation of the system was carried out on one of the main problems of network security - identification of hosts engaged in malicious network scanning. Compare possibility and complexity of the algorithm implementation for SDN and traditional networks.

We chose recently implemented in [3], [4] popular network anomaly detection algorithms TRW-CB [5] and Rate Limit [6].

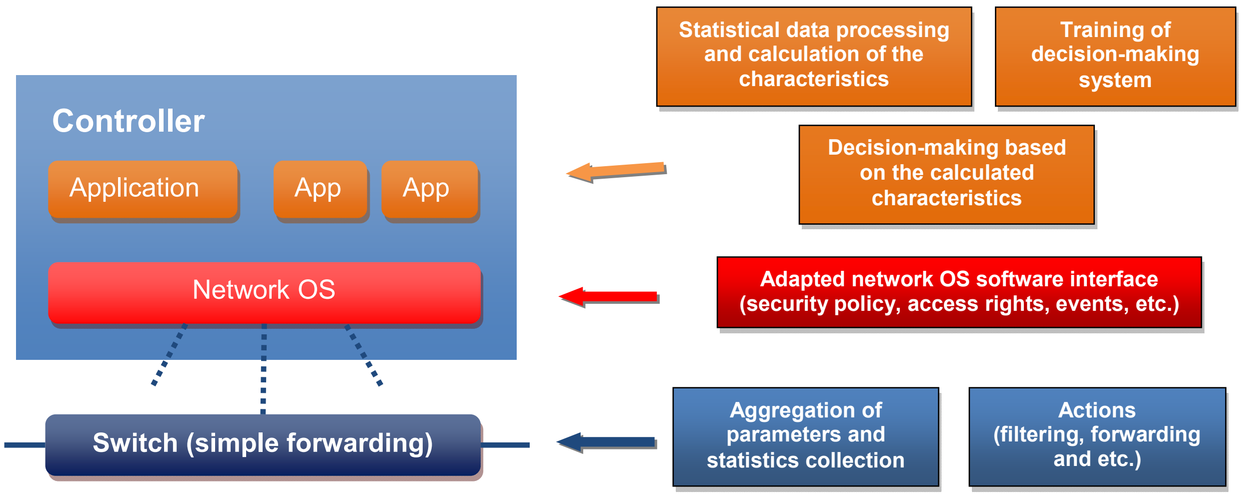
Controller
Application
App
App
Network OS
Switch (simple forwarding)
Statistical data processing and calculation of the characteristics
Training of decision-making system
Decision-making based on the calculated characteristics
Adapted network OS software interface (security policy, access rights, events, etc.)
Aggregation of parameters and statistics collection
Actions (filtering, forwarding and etc.)

Figure 2. Architecture of a protected SDN

TRW-CB algorithm is based on the fact that the probability of a connection attempt being a success should be much higher for a benign host than a malicious one. “Rate Limiting” algorithm uses the observation that in the process of port scanning or virus spread, malicious host tries to connect to the various hosts in a short period of time. On the other hand, benign host performs the connection attempts relatively rare, and mostly to the same hosts. In this paper, to make a decision about the presence of malicious traffic, we use a combination of these algorithms with fuzzy logic inference.

TRW-CB algorithm was implemented as follows:

1. Assume that the host A sends a TCP SYN packet to the new host B. Since there are no flows in the switch matching this packet, the packet is forwarded to the controller.
2. The algorithm instance running on the controller simply forwards this packet, through the switch, to host B, without setting any new flows. At the same time, the algorithm adds host B to the list of hosts that host A tried to contact and decrements host’s balance.
3. There are two possible answers from host B:
   a. If TCP SYNACK packet from B to A is received (switch again forwards this packet to the controller, because still no flows matching this packet) then algorithm sets two flows (from A to B and backwards), and deletes the request from A host queue, as well as increments balance.
   b. If TCP SYNACK packet from B is not received, the algorithm does standard counters processing for the case of connection failure without interacting with the switch and without setting any flows.

Credit-based connection rate limiting summarized in Table 1.

“Rate Limiting” algorithm was implemented as follows:

1. Whenever a new connection request arrives to a host which has recently been successful connected (working set), we set two flows in either direction between hosts.
2. If new request to connect arrives to a host, which is not in the working set, we add it into the delay queue.
3. Every d seconds, the new connection requests are moved from the delay queue to the working set and we forward these requests through the switch without installing any flows.
4. When receiving a positive reply (TCP SYNACK packet), we install a pair of flows in both directions.

TABLE 1. CHANGES TO A HOST’S BALANCE

Event	Change to Ci
Starting balance	$Ci \leftarrow 10$
Connection has issued by host i	$Ci \leftarrow Ci - 1$
Connection succeeds	$Ci \leftarrow Ci + 2$
Every second	$Ci \leftarrow \max(10; \frac{2}{3}Ci)$ if $Ci > 10$
Allowance	$Ci \leftarrow 1$ if $Ci = 0$ for 4 seconds

Figure 3 shows membership functions used in fuzzy decision-making system for inputs and output variables.

Additionally input variables for the system may contain statistical data from switches, for example data speed of selected flows and ports, minimum and maximum number of packets per one IP and etc.

IV. Experimental Simulations

For software implementation of algorithms for decision-making has been used jFuzzyLogic [7] library in Java. Fuzzy inference was carried out using Mamdani algorithm, which currently has the most practical application in problems of fuzzy modeling, since this algorithm is based on fuzzy logic and avoids the excessive amount of computation.

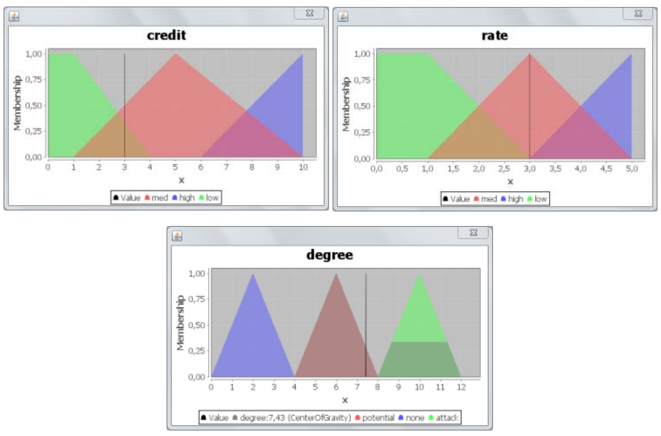

Figure 3. Membership functions for prototype implementation

We used standard Fuzzy Control Language (FCL). In this approach, variables presented above, has following code representation:

```
FUNCTION_BLOCK FuzzySec
// Define input variables
VAR_INPUT
        credit : REAL;
        rate : REAL;
END_VAR
// Define output variable
VAR_OUTPUT
        degree : REAL;
END_VAR
FUZZIFY credit
        TERM low := (0, 1) (1, 1) (4, 0) ;
        TERM med := (1, 0) (5,1) (10,0);
        TERM high := (6, 0) (10, 1);
END_FUZZIFY
FUZZIFY rate
        TERM low := (0, 1) (1, 1) (3, 0) ;
        TERM med := (1, 0) (3,1) (5,0);
        TERM high := (3, 0) (5, 1);
END_FUZZIFY
DEFUZZIFY degree
        TERM none := (0,0) (2,1) (4,0);
        TERM potential := (4,0) (6,1) (8,0);
        TERM attack := (8,0) (10,1) (12,0);
        METHOD : COG;      // Use 'Center Of
Gravity' defuzzification method
        DEFAULT := 0;
END_DEFUZZIFY
// Inference rules
RULEBLOCK No1
        AND : MIN;    // Use 'min' for 'and'
        ACT : MIN;    // Use 'min' activation
method
        ACCU : MAX;   // Use 'max' accumulation
method

        RULE 1 : IF credit IS high OR rate IS
low THEN degree IS none;
        RULE 2 : IF credit IS med OR rate IS
med THEN degree IS potential;
        RULE 3 : IF credit IS low OR rate IS
high THEN degree IS attack;
END_RULEBLOCK
END_FUNCTION_BLOCK
```

To evaluate the work of the modules we use mininet OpenFlow emulator which provides a rapid prototyping workflow to create software-defined network. Using mininet, we emulate one OpenFlow network switch and the four hosts connected to the switch, as well as a host which runs Beacon controller, as illustrated in Figure 4. For flow generation we choose two hosts, which initiate a TCP or UDP connections with various ports of other hosts. Our testbed was setup in a virtual environment (VirtualBox) on Intel i5 PC with 8 GB of RAM.

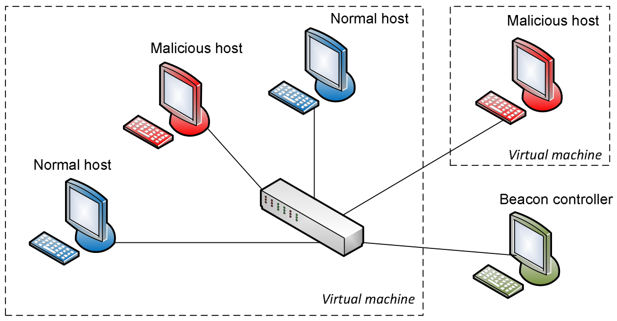

Figure 3. Testbed environment

During the experiment, each host initiated attacks at different speeds, three low (0.1/1/10 pps) and two high (100/1000 pps). Each attack lasted for two minutes. Attack traffic was mixed with normal user traffic. Thus, about 1700 connections were made and about 520,000 packets transferred. As a result, the proposed system was able to identify 95% of the attacks, at 1.2% false positives.

V. CONCLUSIONS

The proposed combined algorithm with the decision-making based on fuzzy rules has shown more accurate results than the algorithms used alone [4]. Percentage of detecting attacks approximately corresponds to the results obtained in [1]. An important feature is that the number of lines of code decreased by 20-30%, as well as the opportunity to easily integrate various external libraries and modules, as well as the opportunity to easily integrate the various external modules and libraries, thus greatly simplifies the implementation of the algorithms and decision-making system.

Software-defined networks provide a unique opportunity for effective detection and containment of network security problems, allowing the integration of complex network security applications in large networks. OpenFlow protocol may eventually become one of the most effective technologies for the development of various innovations in the field of network security.

Future work

In the future we plan to implement a prototype of a secure network operating system based on fuzzy security model, where an access control decision is based on the perception of the level of potential risk associated with the requested access.

This model making access control much more dynamic and flexible by applying different risk mitigation actions to different information flows, depending on the level of perceived risk.

REFERENCES

[1] Dotsenko S.M., Vladyko A.G., Letenko I.D. “Intrusion detection systems based on embedded microprocessor systems” Telekommunikatsii (Telecommunications) № S7 (2013): 15-18.

[2] Vladyko A.G., Letenko I.D., Dotsenko S.M. “Network protection system” RU Patent RU 133954 U1 G06F21/00 (2013.01)

[3] Shin, Seugwon, et al. "FRESCO: Modular composable security services for software-defined networks." Proceedings of Network and Distributed Security Symposium. 2013.

[4] Mehdi, Syed Akbar, Junaid Khalid, and Syed Ali Khayam. "Revisiting traffic anomaly detection using software defined networking." Recent Advances in Intrusion Detection. Springer Berlin Heidelberg, 2011.

[5] Schechter, Stuart E., Jaeyeon Jung, and Arthur W. Berger. "Fast detection of scanning worm infections." Recent Advances in Intrusion Detection. Springer Berlin Heidelberg, 2004..

[6] Williamson, Matthew M. "Throttling viruses: Restricting propagation to defeat malicious mobile code." Computer Security Applications Conference, 2002. Proceedings. 18th Annual. IEEE, 2002..

[7] Cingolani, Pablo, and Jesus Alcala-Fdez. "jFuzzyLogic: a robust and flexible Fuzzy-Logic inference system language implementation." Fuzzy Systems (FUZZ-IEEE), 2012 IEEE International Conference on. IEEE, 2012.