A Comprehensive Security Analysis Checksheet for OpenFlow Networks

Yoshiaki Hori[1,2], Seiichiro Mizoguchi[3], Ryosuke Miyazaki[2,4], Akira Yamada[3], Yaokai Feng[2,4], Ayumu Kubota[3], and Kouichi Sakurai[2,4]

[1] Organization for General Education, Saga University,
1 Honjo, Saga 840-8502, Japan
[2] Institute of Systems, Information Technologies and Nanotechnologies,
2-1-22 Momochihama, Sawara-ku, Fukuoka 814-0001, Japan
[3] KDDI R&D Laboratories, Inc.,
2-1-15 Ohara, Fujimino, Saitama 356-8502, Japan
[4] Faculty of Information Science and Electrical Engineering, Kyushu University,
744 Motooka, Nishi-ku, Fukuoka 819-0395, Japan

Abstract. Software-defined networking (SDN) enables the flexible and dynamic configuration of a network, and OpenFlow is one practical SDN implementation. Although it has been widely deployed in actual environments, it can cause fatal flows. In this paper, we consolidate the security threats to OpenFlow mentioned in previous work and introduce a new security checksheet that includes risk assessment methods. We compare the Kreutz et al. threat vectors with the SDNSecurity.org attack list to discover new threats. Our checksheet enables the security of a given OpenFlow network design to be comprehensively assessed. Furthermore, we evaluate the performance of an OpenFlow network with two attack scenarios using the checksheet and identify critical performance degradations.

Keywords: SDN, OpenFlow, System Security, Risk Assessment

1 Introduction

Software-defined networking (SDN) is an emerging networking paradigm that is a good candidate for relieving the limitations of the current network infrastructures [1][2]. By separating the control logic (the *Control-Plane*, referred to as the *C-Plane* hereafter) of the network from data packet forwarding mechanisms (the *Data-Plane*, referred to as the *D-Plane* hereafter) such as traditional routers and switches, it enables dynamic and flexible configurations of the network in order to properly allocate network resources.

When building a network infrastructure, considering the security of a network for social infrastructure to reduce its security risk is mandatory. When building a network using a SDN, the security of the SDN is one of the requirements of its system design. An SDN tends to be more complicated than traditional non-SDN networks because it consists of many components and their interfaces.

Therefore, building a secure SDN is a mandatory challenge for future various network infrastructures, from a campus network to a carrier's backbone network. We focus on the OpenFlow [3] network, which is one implementation of an SDN. OpenFlow has interface protocols between the C-Plane and D-Plane that are widely used in actual network environments and will be deployed in the future.

In this paper, we deal with the threats to the OpenFlow network and their countermeasures. We classify the security threats of the OpenFlow network and make clear its security risks. Furthermore, we discuss a method for risk assessment and countermeasures for every security risk. We devise a security checksheet for the security of the SDN system. We believe our SDN security checksheet is useful for designing a secure SDN network. The contributions of this paper are as follows:

– We classify the security threats of the OpenFlow network system by consolidating the Kreutz et al. threat vectors and the SDNSecurity.org attack list, and we introduce some new significant risk items to complete our security threat list.
– We create a security checksheet that includes practical assessment methods for risks and their countermeasures. This security checksheet is useful for the risk assessment of an OpenFlow network system design and its operation.
– We evaluate two DoS (Denial of Service) risk scenarios that are included our proposed security checksheet with a given actual OpenFlow network testbed consisting of commercial OpenFlow switches and an open source OpenFlow controller implementation. As a result, we obtain quantitative conditions for the risk.

2 Organizing SDN Security Threats

In 2003, the National Institute of Standards and Technology (NIST) originally published the "Guideline on Network Security Testing" (NIST SP800-42) [4] as a guideline for security when constructing a network. In 2008, NIST also published the "Technical Guide to Information Security Testing and Assessment" (NIST SP800-115) [5], updating NIST SP800-42. Although these documents mention network security, they do not consider an OpenFlow network system. There are some existing studies that analyze the security of SDN. For example, Shin et al. presented an early discussion about attacks on SDN [9]. They briefly mention the C-Plane's resource consumption or DoS attacks, and D-Plane's resource consumption or DoS attacks. Kilöti et al. performed a security analysis of OpenFlow using STRIDE and an attack tree approach [10]. They focused on a Data Flow Diagram of the OpenFlow protocol, which does not include OpenFlow applications or the system environment. Hayward et al. recently presented a survey on security in SDN [11]. They summarized several security analysis studies. However, their work focused on specific layers and interfaces and did not provide a comprehensive security analysis. Kreutz et al. [6] and SDNSecurity.org [7] separately summarized OpenFlow's security threats in 2014 and 2015, respectively, but they do not provide assessment methods and countermeasures for a given OpenFlow network. We consolidate the security threats of an OpenFlow network system by comparing the Kreutz et al. threat vectors and the SDNSecurity.org attack list, and we introduce some new significant risk items to create our final security threat list.

2.1 Seven Threat Vectors of Kreutz et al.

Kreutz et al. pointed out the seven main potential threat vectors in SDN [6], which are as follows: **Threat vector 1:** forged or faked traffic **Threat vector 2:** attacks on vulnerabilities in switches **Threat vector 3:** attacks on C-Plane communications **Threat vector 4:** attacks on and vulnerabilities in controllers **Threat vector 5:** lack of mechanisms to ensure trust between the controller and management applications **Threat vector 6:** attacks on and vulnerabilities in administrative stations **Threat vector 7:** lack of trusted resources for forensics and remediation

They state that threat vectors 3, 4, and 5 are specific to SDN, as they stem from the separation of the C-Plane and D-Plane, and the others are not specific. In addition, they proposed nine solutions for making control platforms dependable and secure against their threat vectors [6]: replication, diversity, self-healing mechanisms, dynamic device association, trust between devices and controllers, trust between application and controller software, security domains, secure components, and fast and reliable software update and patching. They proposed a general design for a secure and dependable control platform. However, a detailed assessment is required for the actual security design of a given SDN network.

2.2 SDNSecurity.org SDN Threat Analysis

The Network and System Security Laboratory of KAIST analyzed the threats to SDN architecture and created an “attack list” for SDN [7]. They categorized the components of an SDN by whether they reside in the Application Layer, Control Layer, Infrastructure Layer, or the Control Channel between the Control Layer and Infrastructure Layer. They then pointed out security threats for every SDN component. Figure 1 shows their list of security threats. For instance, one item on the attack list, **[A-1] packet-in flooding**, is a threat to network operating systems. These details were posted on the SDNSecurity.org site in the summer of 2015. However, this site was only partially online as of May 2016, and the attack list is no longer available.

2.3 Reported Vulnerabilities of OpenFlow

Benton et al. provided a brief overview of the vulnerabilities present in the OpenFlow protocol [8]. They highlighted the classes of vulnerabilities that emerge from the separation and centralization of the protocol plane in OpenFlow network designs. They discuss Man-in-the-middle Attacks, Listener Mode, Switch

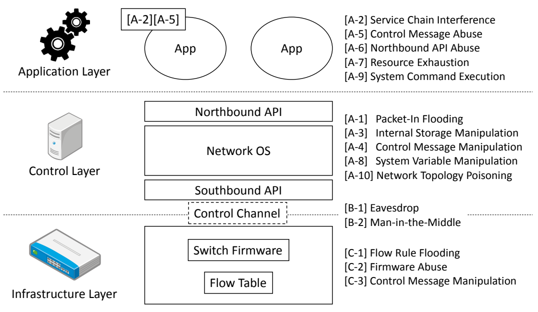
[A-2][A-5]
App
App
Application Layer
[A-2] Service Chain Interference
[A-5] Control Message Abuse
[A-6] Northbound API Abuse
[A-7] Resource Exhaustion
[A-9] System Command Execution
Northbound API
Network OS
Southbound API
Control Layer
[A-1] Packet-In Flooding
[A-3] Internal Storage Manipulation
[A-4] Control Message Manipulation
[A-8] System Variable Manipulation
[A-10] Network Topology Poisoning
Control Channel
[B-1] Eavesdrop
[B-2] Man-in-the-Middle
Switch Firmware
Flow Table
Infrastructure Layer
[C-1] Flow Rule Flooding
[C-2] Firmware Abuse
[C-3] Control Message Manipulation

Fig. 1. Threat analysis of SDNSecurity.org. The authors drew this figure based on [7].

Authentication, Flow Table Verification, DoS Risks, and Controller Vulnerabilities. However, they discuss them only briefly. For OpenFlow network design and operation, it is important to organize the details of the OpenFlow network system vulnerabilities and discuss them.

3 Our Proposal

In order to improve the security of a given SDN system, it is important to prepare a checksheet for risk assessment. Before we provide the checksheet, we list the security threats against SDN systems.

3.1 OpenFlow Network System Security Threat List

To list the threats against SDN, we refer to the comprehensive survey by Kreutz et al. [6] and the vulnerability list by SDNSecurity.org [7]. Table 1 lists these threats.

The "Category" column represents the objects that would be damaged by the threats. There are three categories: D-Plane, C-Plane, and Others. The D-Plane includes the data path and switches, and the C-Plane includes the southbound API (Application Programming Interface), controller itself, northbound API, and applications. The Others category consists of the systems that operate administrative stations, forensics, or remediation.

Next, by referring to and supplementing the vulnerability list of SDNSecurity.org, we define additional SDN security threats as follows:

Switch Table Manipulation: If an SDN switch has a forwarding table, adversaries could try to manipulate this table to redirect traffic to invalid destinations. If a controller has such a table and synchronizes the switches under the controller, this controller can also be a target of switch table manipulation.

Firmware Manipulation: An SDN switch stores its firmware image in memory. Adversaries could try to manipulate this image in order to inject malware functions so as to start the malware at every boot instance. If this firmware is stored in other components, such as a controller or an administrative station, these components could also be targets.

Vulnerability Exploitation of the Switch Program: If the firmware of a switch has a vulnerability that is not yet publicly known, a zero-day attack against the switch is possible. Because we cannot prevent zero-day attacks in general, we have to construct a security incident response team (CSIRT) to monitor the vulnerability information and create an incident response manual.

Vulnerability Exploitation of the Controller Program: This is the same as the vulnerability of the switch program, and a CSIRT must also be organized for its response.

In addition, we assign an ID number to each threat. The “Basic Mechanism” column shows how these threats are launched by adversaries. This information is used for the next risk assessment step.

The first column of Table 1 shows the threat category. The second and third columns show the threat vectors from Kreutz et al. [6] and SDNSecurity.org [7], respectively. For the actual network design, we analyze the Kreutz et al. threat vectors to determine finer threats and carry out a risk analysis for each finer threat and determine its countermeasure. The SDNSecurity.org work can also be classified into finer threats. However, we should add some switch and controller related threats: Switch Table Manipulation, (Switch) Firmware Manipulation, Vulnerability of the Switch Firmware in the D-plane, and Controller Vulnerability Exploitation in the C-Plane. We add these security threats in the fourth column.

3.2 OpenFlow Network System Security Assessment Checksheet

Using our proposed table (Table 1), we created a security checksheet for the OpenFlow network system that consists of security risk assessment items and candidate countermeasures. This security checksheet is useful for risk analysis during OpenFlow network system design and operation. This checksheet makes it easy for a network designer or operator to determine risk items and countermeasures for reducing related risks. Table 2 shows the proposed OpenFlow Network System Security Checksheet.

We created the SDN security assessment checksheet (Table 2) based on the threat list shown in Table 1. The contents of each column are explained in detail in the following list.

ID: This column lists the sequence number.

Category: This column includes the D-Plane, C-Plane, or Others categories.

Condition: Using the basic mechanisms shown in Table 1, this column represents the condition under which the threats occur. For example, in order to login to a switch, adversaries must be able to access its management port. If the switch does not have such a management port or there is no path to the

Table 1. OpenFlow System Security Threats

ID	Vulnerability Check Items	Category	Basic Attack Mechanisms	Threat Vector by Kreutz[]	Vulnerability Genome Project by SDNSecurity.org[]	Our Original
1	Forged or Fake Traffic Flows in Data Plane	D-Plane	Adversaries send forged packets to data plane from the outside of the SDN or from local network.	Threat Vector 1: Forged or Fake Traffic Flows		
2	Firmware Abuse		Adversaries intrude control plane and login to switches.	Threat Vector 2: Vulnerabilities of Forwarding Devices	C-2: Firmware Abuse	
3	Packet_IN Flooding (Switch)		Based on ID 1, adversaries intentionally raise Packet_IN events.		A-1:Packet_IN Flooding	
4	Flow Rule Flooding		Adversaries intrude control plane and issue flow rule configurations.		C-1:Flow Rule Flooding	
5	Control Message Manipulation		Adversaries intrude control plane and send fake control messages.		C-3:Control Message Manipulation	
6	Switch Table Manipulation		Adversaries login switches and manipulate its switch table, or if controllers have switch tables, adversaries login these controllers and manipulate switch table database.			✓
7	Firmware Manipulation		Adversaries login switches and manipulate its firmware images, or manipulate firmware on the management stations.			✓
8	Vulnerability Exploitation of Firmware (Switch)		Adversaries exploit unknown or known vulnerabilities in switch firmware.			✓
9	Packet_IN Flooding (Southbound)	C-Plane	Based on ID 1, adversaries try to waste bandwidth between switches and controller.	Threat Vector 3: Compromise Southbound API	A-1:Packet_IN Flooding	
10	Eavesdrop		Adversaries intrude control plane and eavesdrop messages.		B-1:Eavesdrop	
11	Man-In-The-Middle		Adversaries highjack southbound or northbound to eavesdrop or manipulate messages.		B-2:Man-In-The-Middle	
12	Control Message Manipulation		Adversaries intrude data plane and send forged control messages to controllers.	Threat Vector 4: Compromise Controllers	A-4:Control Message Manipulation	
13	Packet_IN Flooding (Controller)		Based on ID 1, adversaries try to waste computational resources on controllers.		A-1:Packet_IN Flooding	
14	Vulnerability Exploitation of Firmware (Controller)		Adversaries exploit unknown or known vulnerabilities in controller program.			✓
15	Internal Storage Manipulation		Adversaries login the controller and manipulate storage.		A-3:Internal Storage Manipulation	
16	System Variable Manipulation		Adversaries login the controller and change system variables.		A-8:System Variable Manipulation	
17	System Command Execution		Adversaries login the controller and issue system commands.		A-9:System Command Execution	
18	Network Topology Poisoning		Based on ID1 or just login the controller, adversaries manipulate topology database.		A-10:Network Topology Poisoning	
19	Service Chain Interference		Adversaries exploit service chain logic to interfere service chain.	Threat Vector 5: Compromise Northbound API and Applications	A-2:Service Chain Interference	
20	Control Message Abuse		Adversaries abuse northbound API or application to issue invalid control messages.		A-5:Control Message Abuse	
21	Northbound API Abuse		Adversaries at the application layer abuse northbound API to damage controllers and applications.		A-6:Northbound API Abuse	
22	Resource Exhaustion		Based on ID 20-21, adversaries try to waste resources for controllers and applications.		A-7:Resource Exhaustion	
23	Vulnerabilities in Administrative Station	Others	Adversaries exploit the vulnerabilities of administrative stations to launch another attacks.	Threat Vector 6: Vulnerabilities in administrative stations		
24	The Lack of Trusted Resources for Forensics and Remediation		Adversaries intrude control plane and damage to forensics system and data.	Threat Vector 7: The lack of trusted resources for forensics and remediation		
25	The Lack of Trusted Operations for Forensics and Remediation		Adversaries exploit remediation logic and damage to remediation process.			✓

port, this threat may not occur. When the network administrator conducts a security risk assessment, this information is useful for selecting the items for risk analysis and countermeasures.

Risk: This column represents the damage against the system when the threat occurs. In order to determine appropriate countermeasures, this information is useful.

Evaluation Points: If the system design meets the attack conditions and the risk is not ignorable, the network administrator conducts an additional evaluation using the points in this column. Based on the evaluation result, the administrator can choose adequate countermeasures.

Countermeasures: This column represents the list of countermeasures, and the network administrator can select solutions from these items. This list should be updated periodically.

Table 2. OpenFlow Network System Security Checksheet

ID	Category	Condition	Risk	Evaluation Points	Countermeasures
1	D-Plane	Adversaries can send packets to data plane of switch.	Waste data plane bandwidth, launch Packet_IN flooding, then service down.	Evaluation with packet generator.	Use of IDS/IPS
2		Adversaries can access management port of switches.	Lead to several risks.	- Check user manual of switches. - Check logging function. - Check intrusion detection function.	- Login Password Management - Logging - Use syslog-based IDS
3		Adversaries can send packets to data plane of switch.	Switch down	- Evaluation with packet generator. - Check monitoring function of abnormal Packet_IN behavior.	Anomaly detection against Packet_IN messages
4		Adversaries can access southbound or controller.	Switch down or flow table disruption.	- Evaluation with flow rule generator - Check monitoring function of abnormal flow rule insertion - Check authentication for flow rule insertion	Anomaly detection against flow rule insertion
5		Adversaries can access southbound or controller.	Switch anomaly	- Evaluation of arbitrary control message insertion - Check message authentication function	Message Authentication
6		Adversaries can login switches or controller.	Flow redirection	- Check switch table integrity check function - Check authentication of switch table manipulation	- Memory Protection - Software Attestation
7		Adversaries can login switches or firmware at the control plane.	Switch untrusted	Check firmware image integrity check function	Secure Boot
8		Adversaries can access from data plane or control plane.	Lead to several risks.	- Check firmware update function - Check ISAC	Firmware Update
9	C-Plane	Adversaries can send packets to data plane of switch.	Waste control plane bandwidth.	- Evaluation with packet generator - Check monitoring function of abnormal Packet_IN behavior	- Anomaly detection against Packet_IN messages at controller - Resource monitor
10		Adversaries can access control plane.	Disclosure of user data.	Check C-Plane confidentiality	C-Plane encryption
11		Adversaries can access control plane.	Disclosure of user data or highjack of controller.	Check authentication between switches and controllers.	Authentication
12		Adversaries can login switches or access southbound.	Highjack of controller.	Check message authentication between switches and controllers.	Message Authentication
13		Adversaries can send packets to data plane of switch.	Waste controller resources.	- Evaluation with packet generator - Check monitoring function of abnormal Packet_IN behavior	- Anomaly detection against Packet_IN messages at controller - Resource monitor
14		Adversaries can access controller.	Lead to several risks.	- Check firmware update function - Check ISAC	Firmware Update
15		Adversaries can login the controller.	Disclosure, manipulation, destruction of data.	Check confidentiality of data store in controllers	- Encryption - Access Control
16		Adversaries can login the controller.	System unstable.	Check integrity check function for system variables	- Memory Protection - Access Control - Logging - Secure Boot
17		Adversaries can login the controller.	Lead to several risks.	- Check system command log - Check anomaly detection function	- Access Control - Logging - Anomaly detection
18		Adversaries can send packets to data plane, or login switches, or login controller.	Hide network anomaly or flow redirection, denial of service.	Check network topology integrity check function	- Topology Database Monitoring - Access Control
19		Adversaries can access data plane, switch, southbound or controller.	Denial of network services.	- Check application behavior logging function - Check application anomaly detection function	- Anomaly Detection of application - Access Control
20		Adversaries can access controller, northbound, application.	Application anomaly or denial of network services.	Check the integrity check function of flow tables and policies.	- Anomaly Detection of application - Access Control
21		Adversaries can access northbound, application.	Block the other application's operation.	- Check Northbound API usage logging function - Check anomaly detection function for Northbound API	Logging and Monitoring Northbound API call
22		Adversaries can access controller, northbound, application.	Application resource exhaustion.	Check application resource monitoring function	Resource monitoring and anomaly detection
23	Others	Adversaries can access administrative station.	Lead to several risks.	Check the behavior logging and monitoring function of administrative stations	- Access Control Logging - Anomaly detection
24		Adversaries can access forensicsremediation system.	Erase attack logs.	Check the confidentiality and integrity of logs	- Encryption - Access Control
25		Adversaries can access forensicsremediation system, and controller.	Drop remediation or backuped firmware and configuration manipulation.	- Check the integrity of config and firmware image - Check the periodic backup functions	- Encryption - Access Control - Periodical Updates

4 Use of OpenFlow Network System Security Assessment

In this section, we evaluate two DoS risk scenarios in an actual OpenFlow network test-bed with typical commercial OpenFlow switches and an open source OpenFlow controller implementation.

4.1 Out SDN/OpenFlow Testbed and Security Assessment

We created an OpenFlow network evaluation environment using the Ryu3.24 OpenFlow controller software and Pica8 P-3297 OpenFlow switch. We evaluated our OpenFlow testbed using our proposed assessment checksheet, which gives qualitative security assessment results. However, quantitative results are desirable for actual network operation. Therefore, we evaluated our OpenFlow network testbed under two DoS scenarios to obtain quantitative results.

4.2 Quantitative Evaluation of DoS Scenario 1 (PACKET_IN Flooding)

In this scenario, we assume that adversaries intentionally send packets that raise vast numbers of PACKET_IN messages to the controller. This results in PACKET_IN messages flooding the controller. The evaluation of this attack can be replaced with an evaluation of the OpenFlow controller, which does not update the flow table.

Experiment Environment For the quantitative evaluation of PACKET_IN flooding, we used our testbed. Figure 2 shows the testbed environment. The OpenFlow switch is connected to hosts A, B, and C. The Ryu OpenFlow controller runs on a VM (virtual machine, consisting of four virtual core CPUs, 4 GB RAM, Ubuntu 14.04 LTS). The VM runs on the physical host machine (Intel Core i7 860 2.8 GHz, 16 GB RAM, Ubuntu14.04 LTS).

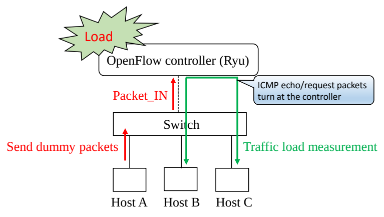

Fig. 2. Environment of PACKET_IN flooding experiment

Experiment We used host A for sending dummy packets, and hosts B and C for measuring the packet's arrival rate. We used packETH for dummy packet generation. Dummy packets were sent at predetermined intervals. When the OpenFlow controller receives a Packet_IN message, it raises a Packet_OUT message; however, it does not update the flow table of the OpenFlow switch. This means that every time the OpenFlow switch receives a packet, a Packet_IN arises from the OpenFlow switch to the OpenFlow controller. However, in order to prevent dummy packets between hosts B and C from affecting the evaluation, the controller discards dummy packets after their Packet_IN is received. Packets between hosts B and C go through the controller. By probing packets' behavior between hosts B and C, we can determine the controller's load during Packet_IN flooding.

We measured the round-trip time (RTT) between hosts B and C when host A sent dummy packets. At the same time, we calculated the packet arrival rate of the number of ICMP echo-reply packets that successfully arrived at host C and compared it to the number of ICMP echo-reply packets that were actually sent from host B.

Results Figure 3 shows the results of the evaluation experiment. The RTT between hosts B and C was measured by the `ping` command. Reachability represents the ICMP packet arrival rate. When the input of dummy packets is less than or equal to 1,000 pps (packets per second), RTT shows no increase and reachability stays at 100%. However, when the input of dummy packets is more than 2,000 pps, RTT starts increasing, and when the input of dummy packets is more than 6,000 pps, reachability falls below 50%. As the pps further increases, RTT increases rapidly and reachability decreases.

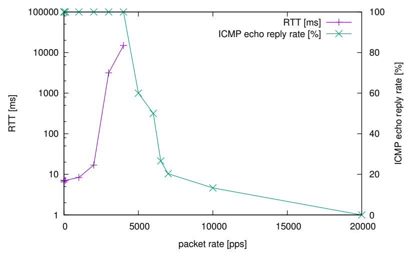

Fig. 3. PACKET_IN flooding experiment using Ryu

Discussion The result of this experiment appears to show that, at most, a rate of Packet_IN messages on the order of thousands of pps causes a serious performance decrease in the OpenFlow controller. This seems to be a result of the limit of the OpenFlow controller's processing ability when running on the host machine. The bandwidth between the switch and the controller may also cause the performance to decrease. If the processing ability of the OpenFlow controller is not sufficiently high, there may be a sudden decrease in performance during DoS attacks.

4.3 Quantitative Evaluation of DoS Scenario 2 (FlowRule Flooding

In this scenario, we assume adversaries intentionally send packets that send vast numbers of Flow_Mod messages to the controller. This results in FlowRule flooding on the controller. The evaluation of this attack can be replaced with an evaluation of an OpenFlow controller that raises a Flow_Mod for every new packet.

Experiment Environment For the quantitative evaluation of FlowRule flooding, we used the testbed. Figure 4 shows the testbed environment. Considering that the OpenFlow system has the ability to run on various kinds of machine, we used a different machine from the one used for the Packet_IN flooding experiment. This machine has a lower performance.

Here, the OpenFlow controller runs on a Raspberry Pi2 and controls an OpenFlow switch. The switch is connected to hosts A, B, and C. We used host A for sending dummy packets at a rate of around 4,000 pps, and hosts B and C for measuring the packet arrival rate. Each dummy packet has a different IP address so that the controller raises a Packet_IN and a flow rule is added to the flow table every time a dummy packet comes to the switch. This causes vast number of flow rules to be added to the flow table of the switch, which could result in flow table overflow. We also set some flow rules in advance on the switch in order to enable communication between hosts B and C. A packet from host B should to be sent toward host C, and a packet from host C should be sent toward host B. That is, all packets between hosts B and C are dealt with within the switch and should never raise a Packet_IN message.

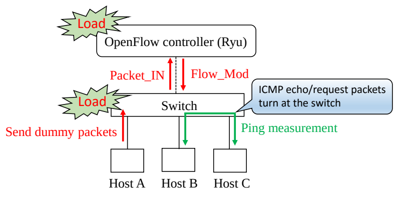

Fig. 4. PACKET_IN flooding evaluation experiment environment

Experiment In this environment, we ran the `ping` command to investigate the effect of flow rule flooding. When an ICMP echo request packet from host B arrives at the switch, the switch looks up its flow table. If a flow rule stating that a packet from host B is supposed to be sent toward host C is already on the flow table, then an ICMP echo request packet is sent toward host C without raising a Packet_IN message. An ICMP echo reply packet from host C is sent to host B in the same way. We sent 50 ICMP echo packets in total. Additionally, we measured the bandwidth, one-way delay time, and packet drop rate between hosts B and C using iperf. We used both TCP and UDP modes for measurement. Note that the maximum UDP bandwidth is limited to 100 Mbps because of the performance limitation of the network interface cards of hosts B and C.

Results Table 3 shows the `ping` evaluation result compared with the values when the controller works as a normal repeater hub. Overall, the RTT of flow rule flooding was higher than that of normal operation. Specifically, the mdev (standard deviation) was higher than normal, which meant there was great variability of the RTT when flow rules flooded the switch. Reachability was lower than the normal value, and 30% of the ICMP packets were dropped when flow rules flooded the switch.

Table 4 shows the iperf evaluation result compared with normal operation values. All values were worse than normal. Specifically, UDP packets dropped by 75%.

Table 3. RTT and reachability of flow rule flooding experiment

		Flow rule flooding	Normal
RTT[ms]	Min	2.882	0.477
	Max	92.476	0.603
	Average	22.409	0.547
	Mdev	21.215	0.04
Reachability[%]		70	100

Table 4. iperf result of the flow rule flooding experiment

		Flow rule flooding	Normal
TCP	Bandwidth [Mbps]	1.04	146
UDP	Bandwidth [Mbps]	24.6	101
	One-way delay [ms]	26.573	0.013
	Reachability [%]	25	99.9

Discussion The results of this experiment appears to show that, at most, a Flow_Mod rate on the order of thousands of pps causes a serious performance decrease in the OpenFlow switch. Vast numbers of Flow_Mod messages may consume the CPU resources of the OpenFlow switch, which may result in an increase of the packet drop rate. We should investigate the data transfer mechanism during Flow_Mod, and, for a secure OpenFlow system, we should design a controller that detects abnormal numbers of Flow_Mod messages.

5 Concluding Remarks

This paper addressed the security threats of OpenFlow network systems and their countermeasures. We classified security threats of the OpenFlow network and clarified its security risks. Furthermore, we discussed a method for the risk assessment and countermeasures for every security risk. We devised a security checksheet for SDN system security. We believe our SDN security checksheet is useful for designing a secure SDN network. In addition, we reported the results of two quantitative evaluation experiments using our OpenFlow testbed.

As future work, we will continuously revise the proposed checksheet to include new threats. In addition, we plan to create decision rules to adopt one or more proper countermeasures for each security threat of the OpenFlow network system.

Acknowledgement

This work was supported by JSPS KAKENHI Grant Number 16K00132.

References

1. Y. Jarraya et al.: A Survey and a Layered Taxonomy of Software-Defined Networking, In IEEE Comm. Surveys & Tutorials, Vol. 16, No. 4, pp. 1955–1980 (2014)
2. D.Kreutz et al.: Software-Defined Networking: A Comprehensive Survey. Proc. of the IEEE, Vol. 103, No. 1, pp. 14–76 (2015)
3. N.McKeown et al: OpenFlow: enabling innovation in campus networks, ACM SIGCOMM Computer Communication Review, Vol. 38, Issue 2 (2008)
4. J. Wack, M.Tracy, M. Souppaya: Guideline on Network Security Testing, NIST Special Publication 800-42 (2003)
5. K. Scarfone, M. Souppaya, A. Cody, A. Orebaugh: Technical Guide to Information Security Testing and Assessment, NIST Special Publication 800-115 (2008)
6. D.Kreutz et al: Towards Secure and Dependable Software-Defined Networks, In: ACM SIGCOMM workshop HotSDN'13, pp.55–60 (2013)
7. SDNSecurity.org: An Overview of Misuse / Attack Cases, `https://web.archive.org/web/20150423094535/http://sdnsecurity.org/project_SDN-Security-Vulnerbility-attack-list.html`(access 2015-12-14)
8. K. Benton, L. J. Camp, C. Small: OpenFlow vulnerability assessment, In: ACM SIGCOMM workshop HotSDN'13, pp.151–152 (2013)
9. S. Shin, G. Gu: Attacking Software-Defined Networks: A First Feasibility Study, In: ACM SIGCOMM workshop HotSDN'13, pp.165–166 (2013)
10. R. Klöti et al.: OpenFlow: A security analysis, In: 21st IEEE Int'l Conf. on Network Protocols (ICNP 2013), pp. 1-6 (2013)
11. S. Scott-Hayward et al.: A Survey of Security in Software Defined Networks, In: IEEE Comm. Surveys & Tutorials, Vol. 18, No. 1, pp. 623-654 (2016)
12. Pica8 switches, Pica8, Inc. `http://www.pica8.com/products/pre-loaded-switches`(access 2015-12-15)
13. PACKETH. `http://packeth.sourceforge.net/packeth/Home.html`(access 2015-12-15)
14. Ryu SDN Framework. `http://osrg.github.io/ryu/`(access 2015-12-15)