Zero-Day Attack Detection and Prevention in Software-Defined Networks

Huthifh Al-Rushdan[1], Mohammad Shurman[2], Sharhabeel H. Alnabelsi[3], Qutaibah Althebyan[4]

[1]Computer Eng. Dept., Jordan University of Science and Technology, Irbid, Jordan
[2]Network Eng. and Security Dept., Jordan University of Science and Technology, Irbid, Jordan
[3]Computer Eng. Dept., Faculty of Eng. Technology, Al-Balqa Applied University, Amman, Jordan
[3]Computer Eng. Dept., Al Ain University, Al Ain, United Arab Emirates
[4]College of Eng., Al Ain University, Al Ain, United Arab Emirates

[1]aahuthifh14@cit.just.edu.jo, [2]alshurman@just.edu.jo, [3]alnabsh1@bau.edu.jo,
[3]sharhabeel.alnabelsi@aau.ac.ae, [4]qutaibah.althebyan@aau.ac.ae

***Abstract*— The zero-day attack in networks exploits an undiscovered vulnerability, in order to affect/damage networks or programs. The term "zero-day" refers to the number of days available to the software or the hardware vendor to issue a patch for this new vulnerability. Currently, the best-known defense mechanism against the zero-day attacks focuses on detection and response, as a prevention effort, which typically fails against unknown or new vulnerabilities. To the best of our knowledge, this attack has not been widely investigated for Software-Defined Networks (SDNs). Therefore, in this work we are motivated to develop anew zero-day attack detection and prevention mechanism, which is designed and implemented for SDN using a modified sandbox tool, named Cuckoo. Our experiments results, under UNIX system, show that our proposed design successfully stops zero-day malwares by isolating the infected client, and thus, prevents these malwares from infesting other clients.**

Keywords*— *controller, switch, Mininet, Intrusion Detection System (IDS), Sandbox, SDN, Zero-day attack.

I. Introduction

Software-DefinedNetwork (SDN) is a new approach in computer networks domain that allows network engineers and administrators to install, control, manage and modify networks. SDN provides a fast response to network requirements and business needs such as installation, editing, and management using a centralized controller. Furthermore, it empowers administrators to reconfigure the network without any physical access to networks devices, e.g.; switches or routers. Therefore, it is extensively employed in data centers by using virtualization techniques.

SDN architecture has three layers, first layer is the forwarding layer that consists of routers and switches. Second layer is the controller layer that consists of controllers. Third layer is the application layer that consists of application and services used to utilize SDN and generate traffic. SDN has two planes: a data plane and a control plane. Data plane operates under open flow protocolplane that is responsible for forwarding packets, while the control plane decides the routing path for packets [1], [2].

When a packet arrives to the switch for the first time, a rule is inserted in the switch forwarding-table by a controller (a strategic control point in the SDN network that manages and controls the flows between network elements) [3], in order to deal with this packet, i.e., either forward it to a specific port or drop it. In fact, the switch sends all packets addressed to the same destination over the same route using the inserted rule. It is important to mention that the switch provides the controller with traffic information [4], where the communication between the controller and the switches is conducted using the OpenFlow protocol [5]. Thus, network administrator can centrally make necessary changes for the forwarding rules of switches (i.e., changing priorities or traffic blocking rules). This allows network administrator to have more control on the network using low cost switches.As a result, the popularity of SDN technology made it a target of security attacks.

Although researchers have paid effort in securing computer networks and sensitive data, there is still a risk of an unknown attacks, called the zero-day attacks [6], [7]. The term "zero-day" refers to the time available for software developer to fix the problem that has been exposed to the public [8]. During this time, vendors try to release an update or a patch file to fix the new vulnerability. When vendor fails to release a patch ontime, the hacker can exploit the exposed vulnerability, and hence, the zero-day attack reallyoccurs. The attacker executes a piece of code on the vulnerable system, in order to gain an illegal access. The term "vulnerability is exploited" occurs when an exploited code has successfully attacked the newly discovered vulnerability [9].

Generally, to defend the networks against unknown attacks, e.g.; zero-day attack, an isolated testing environment,named a sandbox, is used to execute untested programs or files that may contain viruses or malicious codes, such that the real environments is not infected [10], due to the fact that the sandbox is isolated from the real environment. The sandbox has a set of resources such as processors, memory, networks and applications that provided by a virtualization technique. In other words, the sandbox contains a number of Virtual Machines (VMs) with different Operating Systems (OSs). Each VM contains different programs, e.g., flash player or java. When a user downloads a file or visits a URL, the security system initially extracts the file or the URL, and then forwards it to the sandbox for execution on all VMs. Moreover, the sandbox sends some control instructions to the VMs, such as mouse movements, clicks, or system time changes, since some malwares require special actions like click on some button or a specific time in the future to run [10].

We are motivated to modify theexisting sandbox, under UNIX OS, to integrate it with SDNs, in order to protect clients PCs and network controller.This work is organized as follows: in section II, the security of SDN is discussed, section III presents the zero-day attack, section IV presents Cuckoo SandBox Analysis. In section V, the proposed solution against zero-day attack is illustrated. Section VI shows experiments results and discussion. Finally,section VII presents the conclusions and future work.

II. Security of Software-Defined Network

SDN security protocols are different from the standard networks, since its nature and characteristics are different. Therefore, SDN introduces new attacks to the controller platform and the connections between different planes. SDN treats control plane as a single entity, which indicates a single security implementation between the control plane and application plane, and between control plane and data plane.

Moreover, the implementation of distributed controls is not visible to SDN architecture, since this may increase network exposure to attacks. On the other hand, the actual controller implementation is more complex and distributed, forcing stronger security requirements. These security requirements can be achieved by providing SDN controllers with a secure environment.

A. *Preliminary*

In network management, a real-time monitoring can be very useful, because it allows to analyze and monitor log entries for forensic analysis and intruders/attacks detection. It is possible to build SDN security techniques that combine stations and network devices security procedures, in order to detect and prevent attacks. One method of security procedures is isolating traffic between SDN users, and between users and control plan. This separation could be more effective and more dynamic than traditional networks, due to the processing and functional capability of data plane component.

The main security issues in the SDN domain are the insiders and operator's errors that may compromise the overall system integrity. To address these issues, SDN architecture must contain strong identity to secure all entities and their associated states [11], in addition to monitoring running processes to make sure they operate properly.

B. *Protection methodology*

In traditional networks, data plane protection and restorations are defined and implemented, SDN does not change these standards as well as the associated protection and protocol state continue to exist and operate within network elements.

Furthermore, SDN controller will be responsible for pre-computing resource recovery, provisioning recovery, and subscribing notification. Moreover, this controller may restore traffic as well by re-establishing path or change routes to optimize resources. SDN resources may be shared between more clients to satisfy their demand,therefore, resources must be fulfilled by combinations of the following procedures:

- Define a resource pool based on availability and recovery time, thenserve clients accordingly.
- Protect the resources based on the most restricting requirement.
- Offer a default level of shared resources protection and provide clients with more restricting need.

III. Zero-Day Attack

The Zero-day attack is a computer attack that exploits a vulnerability. The vulnerability is a weakness in the software or in a security policy that allows the attacker to gain access to the system that has not been known to the public yet. Its aim is gaining illegal access or threat a running system [12], [13]. It is very difficult to defend against the zero-day attack, since it is always detected after the system has been compromised, while the vulnerability has no known signature and no mechanism to stop or detect the zero-day attack [12]. Once the vulnerability has been announced to the public, system administrator can patch the system, and the antivirus companies can insert it in to the signature update [13].

Although, system patching, upgrading, antiviruses, and IDS can tackle many kinds of attack, the zero-day attack cannot be tackled due to the lack of information about the attack's nature [14]. Discovering the zero-day vulnerability and figuring out how to stop it is a very difficult task. The zero-day vulnerability is considered as the most harmful threat for computer organizations, because their system and services are exposed to the public network and to the attacker before the patch becomes available. Generally, there are four kinds of traditional defense technology against attacks: statistical-based, signature-based, behavior-based, and hybrid-based [15].

IV. Cuckoo SandBox Analysis

Cuckoo sandbox has three VMs for testing whichcontains three versions of windows: Windows XP, Windows 7 and Windows 10. Once Cuckoo sandbox receives the files or the URLs for analysis,the analysis process starts by restoring the current VM snapshot that contains a clean windows environment with several installed applications. Next, cuckoo will execute the file or open the requested URL in the browser, where the agent collects all changes in the VMs by profiling memory dump and registry information. After that, the agenttransfers the collected changes to cuckoo sandbox for analysis by examining the memory dump, files created by the malware, and the registry information [10].

Once the analysis is completed, a report is generated for analysis result. The generated report has a score out of 10 representing the severity of attack for the file or the URL as follows:

- If (score< 2), this indicates the file or the URL is harmless.
- If ($2 \leq$ score ≤ 5), this indicates the file or the URL has a high probability of being harmful.

- If (score> 5), then this indicates file or the URL is definitely harmful.

The score value will be sent to the controller to make an appropriate decision including isolating the client or blocking allits incoming traffic.

V. Proposed Solution against zero-day attack

Security in SDN network is different from the traditional network security, since in SDN network the gateway is directly connected to the internal network, where all security devices are either installed in the application layer (controlled by the controller for traffic forwarding to the appropriate security device), or installed in the gateway layer (connected to the internal network), noting that in this case the controller has no control over the gateway devices[16]. Another main difference from traditional network is the controller controls every node in the network and can block the nodes' traffic or forward it to a specific path.

In this work, we implement a new system that protects two components: (1) the controller, and (2) client PCs from the zero-day attack in SDN. The system eliminates malware effects and protects the whole network from infection.We will use the mininet simulation tool [17], in order to implement SDN, forwarding switches and the client PCs. Forwarding switches are connected to the controller using OpenFlow protocol [18], where the controller manages traffic flow by forcing rules.

A. *Client PCs Protection*

In order to ensure client PCs protection, all traffic passes the OpenFlow switches goes through two stages: First, it is forwarded to the controller, on a specific network interface, where a customized python program extracts transferred files to the client or the requested URLs by the client. Once the files or the URLs have been extracted, the extraction program will submit them to the cuckoo sandbox for analysis in order to detect malwares, if exist.

B. *SDN Controller Protection*

The protection of SDN controller from the zero-day attack is different from the client protection. Cuckoo sandbox only tests the malware under windows environment, however, the controller is usually based on UNIX, which is not supported by Cucko. Therefore, we are strongly motivated to build our UNIX-based sandbox.

Our developed sandboxcontroller consists of an agent installed on a VM,as an SDN controller, and an application runs on a machine which is hosting the controller, where the communication between the sandbox and the controller will be carried out through a dedicated Ethernet channel.

In our scenario, the agent will monitor three main parts that affect controller's status:

- Added or removed features to/from the controller.
- The status of the service port in the controller.
- The status of specific service in the controller's OS.

If any feature is changed, then a new attack has occurred. The flowchart that demonstrates the steps of our developed controller sandbox is illustrated in Figure 1.

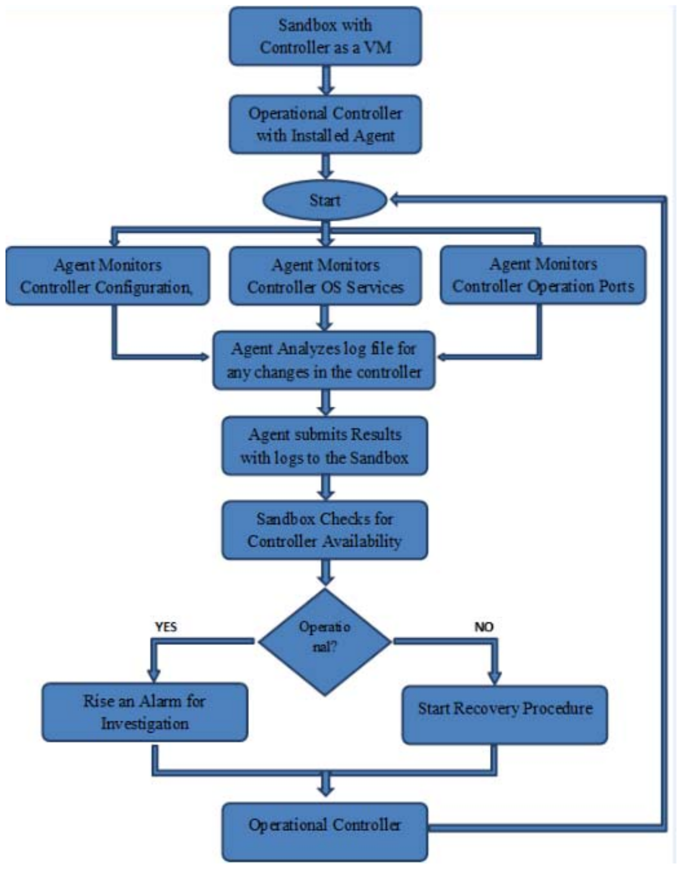
Sandbox with Controller as a VM
Operational Controller with Installed Agent
Start
Agent Monitors Controller Configuration,
Agent Monitors Controller OS Services
Agent Monitors Controller Operation Ports
Agent Analyzes log file for any changes in the controller
Agent submits Results with logs to the Sandbox
Sandbox Checks for Controller Availability
YES
Operational?
NO
Rise an Alarm for Investigation
Start Recovery Procedure
Operational Controller

Fig. 1. The steps for our developed controller sandbox.

VI. EXPERIMENTAL RESULTS AND DISCUSSION

A. *Platforms*

Simulation experiments areconducted using these tools:

- Client Machines: Intel® Core™2 i5-3230M CPU, 8 GB DDR3 RAM, which is used to download the malware and act as infected client.
- Controller Sandbox: Intel® Core™2 i7-4770M CPU, 24 GB DDR4 RAM, which has the application that monitors the SDN controller and the virtualization software that hosts the VM controller.
- VM Controller: Intel® Core™2 i7-4770M CPU, 4 GB DDR4 RAM, a virtual machine that installed with the controller software and with the agent that monitors the SDN controller.
- Cuckoo Host: Intel® Core™2 i7-4770M CPU, 12 GB DDR4 RAM, equipped with the cuckoo software.
- Cuckoo VM: Intel® Core™2 i7-4770M CPU, 2 GB DDR4 RAM, a virtual machine that used by cuckoo sandbox to test the malwares.
- Switch: OpenFlow switch v1.0.

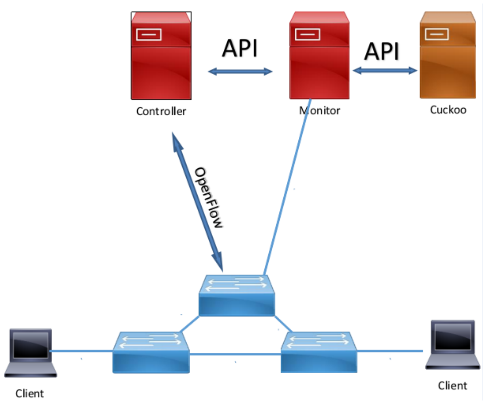

Fig. 2. Testing environment for SDN client.

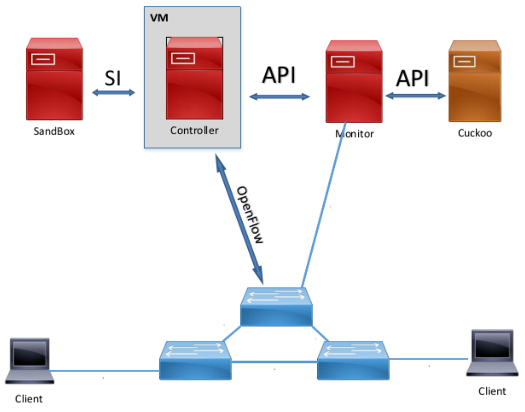

Fig. 3. Testing environment for SDN controller.

Figure 2 shows the testing environment for our proposed solution, in which SDN client is protected from the zero-day attack. Figure 3 shows the environment used to test the proposed solution that protects the SDN's controller from the zero-day attack.

B. Results

Table I shows different malwares with different sizes (KB) that tested by cuckoo sandbox, where the size affects the execution time for malwares detection. Results show that cuckoo has successfully identified 353 malwares of 361 with an overall percentage of 97.78% (which is a very good percentage).Note that this percentage can be further improved by more customizing cuckoo sandbox configuration.

VII. Conclusions and Future Work

The zero-day attack has become a very serious attackin recent years, since it is a random attack which cannot be predicted. The zero-day attack utilizes a weakness in a software to gain access to the system or make a huge damage, wheresystem developershave zero time to solve thisvulnerability in order to limit the threat. The proposed solution identifies and blocks malwares in the zero-day attack under Software-Defined Networks(SDNs), in order to protect two components: First, the clients PCs that are protected by our customized python code which resides in the controller. Second, attacks on SDNcontroller which are prevented using our developed UNIX-based sandbox where traffic is monitored using the added detection rules. As a future work, we plan to study the effect of malwares size, RAM and processor speed on analysis time.

TABLE I. Number of Tested and Identified Malwares.

Experiment number	**Malware size (KB)**	**Number of tested malwares**	**# of successfully identified malwares**
1	2	20	19
2	30	18	18
3	50	22	21
4	70	22	20
5	80	19	19
6	90	20	20
7	100	17	16
8	150	16	16
9	200	20	20
10	300	19	19
11	400	18	17
12	500	17	16
13	600	19	19
14	700	20	20
15	800	20	20
16	900	21	21
17	1000	13	13
18	1200	21	21
19	1400	19	18
Total		**361**	**353**

References

[1] Myung-Ki Shin, Ki-Hyuk Nam, Hyoung-Jun Kim, "Software-defined networking (SDN): A reference architecture and open APIs," *International Conference on ICT Convergence (ICTC)*, 2012.

[2] E. Haleplidis, S. Denazis, O. Koufopavlou, J. Halpern, J. Salim, "Software-Defined Networking: Experimenting with the Control to Forwarding Plane Interface," *In Proceedings of the European Workshop on Software Defined Networks (EWSDN)*, Germany, 2012.

[3] Doria, J. Salim, R. Haas, H. Khosravi, W. Wang, L. Dong, R. Gopal, J. Halpern, "Forwarding and Control Element Separation (ForCES) Protocol Specification," RFC 5810 (Proposed Standard), 2010.

[4] W. Braun, M. Menth, "Software-Defined Networking Using OpenFlow: Protocols, Applications and Architectural Design Choices*," Journal of Future Internet*, vol. 6, no. 2, 2014.

[5] Nishtha , M Sood, "Software defined network-Architectures," *International Conference on Parallel Distributed and Grid Computing*, India, 2014.

[6] M. Keramati. "An attack graph based procedure for risk estimation of zero-day attacks,"*the8th International Symposium on Telecommunications (IST)*, Iran, 2016.

[7] Meneely, S. Lucidi, "Vulnerability of the Day: Concrete Demonstrations for Software Engineering Undergraduates," *the 35th International Conference on Software Engineering (ICSE)*, USA, 2013.

[8] R. Kaur, M. Singh, "A Survey on Zero-Day Polymorphic Worm Detection Techniques," *IEEE Communications Surveys & Tutorials*, vol. 16, issue 3, 2014.

[9] M. Almukaynizi, E. Nunes, K. Dharaiya, M. Senguttuvan, J. Shakarian, P. Shakarian, “Proactive Identification of Exploits in the Wild Through Vulnerability Mentions Online,” *International Conference on Cyber Conflict (CyCon U.S.)*, USA, 2017.

[10] M, Vasilescu, L, Gheorghe, N, Tapus, “Practical malware analysis based on sandboxing,” *In 2014 RoEduNet Conference 13th Edition: Networking in Education and Research Joint Event RENAM 8th Conference, Moldova*, 2014.

[11] Understanding the SDN Architecture – SDN Control Plane & SDN Data Plane, Web Reference: https://www.sdxcentral.com/sdn/definitions/inside-sdn-architecture.

[12] Open network foundation “SDN architecture,” Technical report, issue 1, 2014.

[13] U. Singh, C. Joshi, S. Singh, “Zero-day Attacks Defense Technique for Protecting System against Unknown Vulnerabilities,” *International Journal of Scientific Research, Computer Science and Engineering*, vol. 5, issue 1, 2017.

[14] L. Bilge, T. Dumitras, “Before We Knew It an Empirical Study of Zero-Day Attacks in The Real World,” *ACM Conference on Computer and Communications Security*, USA, 2012.

[15] L. Wang, M. Zhang, S. Jajodia, A. Singhal, and M. Albanese, “Modeling network diversity for evaluating the robustness of networks against zero-day attacks,” *19th European Symposium on Research in Computer Security*, Poland, 2014.

[16] Kreutz, Diego, Fernando Ramos, and Paulo Verissimo. "Towards secure and dependable software-defined networks," *In Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, pp. 55-60, 2013.

[17] Rashma, B. M., and G. Poornima. "Performance Evaluation of Multi Controller Software Defined Network Architecture on Mininet," *In International Conference on Remote Engineering and Virtual Instrumentation*, pp. 442-455. Springer, Cham, 2019.

[18] Goto, Yuki, Bryan Ng, Winston KG Seah, and Yutaka Takahashi. "Queueing analysis of software defined network with realistic openflow–based switch model." *Computer Networks 164 (2019)*: 106892.