Vol.2016-CSEC-73 No.4
Vol.2016-IOT-33 No.4
2016/5/26

On a SNMP DoS attack against Vulnerable Architecture of Network Equipment

Motoyuki OHMORI[†1]

Abstract: We had experienced that our core network switch sometimes stalled. If the worst came to the worst, the switch got unable to forward any IP packets. The cause eventually appeared to be architectural vulnerability of the switch that could cause high CPU load when the switch received SNMP packets. The switch was, however, configured not to respond to SNMP packets that malicious attackers sent, i.e., this vulnerability was not related to so-called SNMP reflection or amplification attack. We spent approximately three months before finally solving this issue since the problematic phenomena spontaneously finished in a short period of time or the switch continued to almost hang up under attack. We then present how to find out and solve the issue, and what a network switch architecture should be like in this paper.

Keywords: SNMP, DoS, architecture design of network equipment

1. Introduction

A computer network in an organization is required to be more invulnerable against attacks in order to achieve high availability and a business continuity.

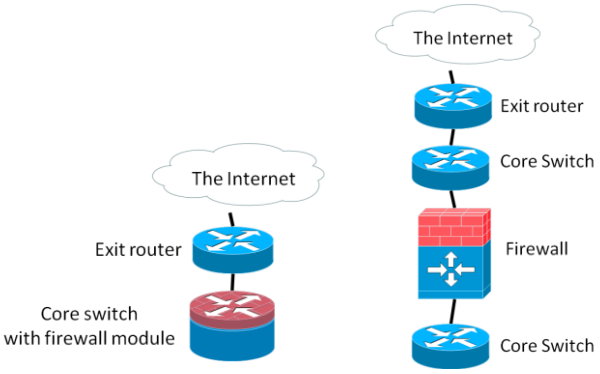
We, Tottori University, had experienced that our campus network totally stalled and no communications were available. The cause eventually appeared to be a high CPU load average on receptions of malicious Simple Network Management Protocol (SNMP)[1] packets. From early in the morning, 8:02 a.m., of 7th January 2015 to 12th February 2015, we suffered from these malicious SNMP packets. We then present how to find out and solve the issue, and what a network switch architecture should be like in this paper.

1. Even popular enterprise core switch is vulnerable to a simple DoS attack and stop forwarding any IP packets, and this results from its architecture design. Such core switch may be much vulnerable especially internal to unintentional attackers whose terminals are infected by malicious software or viruses.
2. Redundancy mechanisms of network equipment can be useless or even be obstacles to shoot troubles caused by DoS attacks.
3. CPU usage monitoring with SNMP is ineffective against DoS attacks if polling interval is long such as 5 minutes. Only second-order polling interval can detect the attacks.
4. It is important to monitor normal traffic and daily analyze them.
5. An Intrusion Prevention System (IPS) and its monitoring service may be unable to detect and to avoid DoS at all.
6. An architecture of network equipment should carefully consider which packets are *punt* to CPUs.

The rest of this paper is organized as follows. Section 2 presents a brief network configuration in our university where we experienced the attacks. Section 3 presents symptoms that we observed and how we struggled to find out and to fix the causes. Section 4 discusses our operations against the attacks, symptoms and important points when ones design network equipment. Section 5 then refers to related works, and Section 6 concludes this paper.

2. Network Configuration

This section briefly present our network configuration and a background of its structure. Figure 1 shows a brief network configuration in Koyama campus of our university. As shown in Figure 1 (a) and (b), our campus network connects to the Internet via the exit router that has a BGP peering with a router of the Science Information NETwork (SINET). The exit router connects to the core switch, Cisco Catalyst 6509, which is a popular L3 switch in an enterprise network environment and was attacked and stalled in our campus network. The core switch embeds firewall module called FWSM.



(a) Physical topology (b) Logical topology

Figure 1 Network configuration.

As shown in Figure 1 (b), the core switch logically connect the firewall and the exit routers, and the core switch itself. Ones may think that this network configuration is dangerous since the core switch has an network segment outside of the firewall. We, however, cannot help but have this network configuration due to a limitation of the firewall, FWSM. We have three virtual firewall contexts in the firewall in order to filter three networks dedicated for education, research and administrative

†1 Tottori University

office. The limitation of the firewall is that the firewall cannot have multiple L3 interface on the same segment between different virtual firewall context. This is the reason why we have an network segment of the core switch outside of the firewall. As described later, this network configuration was one of causes of a network failure that we present in this paper.

3. Observed Symptom and Tried Fixes

3.1 The beginning of a nightmare

Our nightmare had begun with a small indication that modules in our core switch did not respond to keep alive messages from the core switch CPU for short period early in the morning, 8:02 a.m., of 7th January 2015 as ones can see the first spike in Figure 2. As Figure 2 shows, not only the modules but also the core switch itself did not respond to SNMP requests. As ones may know, it is common that network equipment does not respond to a keep alive message or SNMP request when the equipment is busy to process other tasks. We then considered this spikes in the graph as a rare case and it might not occur again so often since users reported no major failures of networks availability and the symptom finished in a short period without any operation.

We, however, consulted with our network vendor in order to locate the cause of no response for keep alive message and SNMP requests for safety since this was the first time that we experienced this phenomenon. Unfortunately, we could not get no reasonable answers from the vendors, and were advised to locate a process that caused the high CPU usage.

We then experienced the more phenomena that the core switch did not respond for a SNMP request. It then seemed that the symptom was going to get worse and worse since the core switch could not sometimes be logged in even via command line interface (CLI). We gradually considered that something wrong happened in our core switch. Under this circumstances, it was so difficult for us to locate the cause of the high CPU usage since the core switch did not respond when we would like to locate the causing processes in the core switch. In addition, we could not see any foresight of the high CPU usage of the core switch by the SNMP retrieving CPU usage of the core switch as shown in Figure 2. This was because the polling interval of the CPU usage of the core switch was 5 minutes and the symptom usually finished in a short period of time.

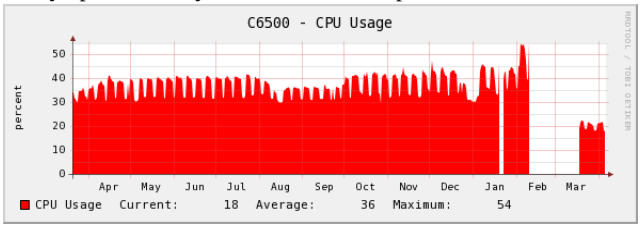


Figure 2 the CPU usage graph of the core switch.

3.2 Stall of IP packet forwarding

While we could not locate the cause of the high CPU usage, the core switch became to sometimes stall and not to forward IP packets for a short period of time. Even though the period was short, our users began to notice that they could not sometimes communicate with correspondent hosts in our campus network. The core switch then gradually stalled for longer and longer time while time goes by.

3.3 Attempt to recover stalls of IP packet forwarding

When the time that the core switch stopped to forward any IP packets in our campus network, we attempted to locate the cause of high CPU usage. One attempt was that isolated the core switch from our campus network because we suspected any loops or broadcast storms in our campus networks and we saw the flapping of a spanning tree of Spanning Tree Protocol (STP)[2]. The other attempt was switching over a running routing engines to a other backup secondary routing engine because we also suspected a routing engine failure. We suspected the routing engine failure because some line cards or modules had appeared to have memory failure. The other attempt was to check to see if switching buses in the core switch saturated or not since we utilized the core switch with oversubscription modes. The buses, however, appeared not to saturate. We then tried to reboot the core switch, and the core switch sometimes recovered.

Even though we attempted to recover failures as described above, we could not recover. At this moment, we still did not locate the cause of the high CPU usage because CLI of the core switch did not respond on the high CPU usage.

3.4 Attempt to locate the cause of high CPU usage

In order to locate the cause of high CPU usage, we programmed a simple script to check to see if CPU usage was about to get high. The script is written in Ruby[3], and always to connect to the CLI of the core switch. The script then polling recent CPU usages of 1 second average at intervals of 5 seconds, not 5 minute of general monitoring of CPU usage with SNMP. When the script detects a recent CPU usage exceeds the threshold that is configured in advance in the script, the script tries to retrieve information of the core switch by executing CLI commands. The script then mails the result to an administrator. As described above, we eventually could obtained necessary information to locate the cause of the high CPU usage with the script.

3.5 Suspicious processes causing of high CPU usage

The script described in section 3.4 could present suspicious processes that caused high CPU usage as follows:

1. iprouting: RIP send
   "RIP send" is a process that announce the routing information using Routing Information Protocol (RIP) [4].
2. ARP
   This is a process that resolves an Ethernet address associated with an IP address of a correspondent host using Address Resolution Protocol (ARP) [5]
3. tcp.proc
   This is a process to process outgoing TCP connection from the core switch.
4. IP SNMP

```
08:50:02.617911 IP (tos 0x0, ttl 238, id 54321, offset 0, flags [none], proto UDP (17), length 61)
   x.x.x.x.40866 > x.x.x.x.161: [udp sum ok]  { SNMPv2c { GetBulk(18) R=47  N=0 M=127 .1.3 } }
      0x0000:  4500 003d d431 0000 ee11 355a xxxx xxxx  E..=.1....5Z]...
      0x0010:  xxxx xxxx 9fa2 00a1 0029 7569 301f 0201  ...F.....)ui0...
      0x0020:  0104 0670 7562 6c69 63a5 1202 012f 0201  ...public..../..
      0x0030:  0002 017f 3007 3005 0601 2b05 00        ....0.0...+..
```

Figure 3 Received Malicious SNMP packets.

This is a top-half process to handle SNMP packets. Note that this process is a different from a process called, "SNMP ENGINE," which is a bottom-half process to handle the SNMP packets and which is a famous of causing high CPU utilization.

Even though the script located suspicious processes, all of them seemed not to be suspicious; RIP is a simple and very old protocol, thus it was considered to be stable enough, and it should not be the cause. ARP is very important protocol and the core switch handles almost all APR packets in our campus network, and its CPU usage was considered to be reasonable even though the usage was a little bit higher in comparison with other processes. "tcp.proc" then consumes if and only if the core switch open outgoing TCP connection. "tcp.proc" was then considered not to be the cause since there were few cases that the core switch initiated TCP connection. We then suspected the left process, "IP SNMP,". We, however, thought that "IP SNMP" was not the case since there was no information about "IP SNMP" that caused the high CPU utilization. Even the vendor says that there was no suspicious process among above processes, and they advised us to locate other processes.

3.6 Locating the CPU packet capture

We then began to capture packets that are "*punt*" to CPU of the core switch even though we were not so confident if the cause resulted from packet receptions or not. Since the failure might finish in a short period of time and might not continue sometimes, we kept capturing packets even though the core switch properly worked. After we kept capturing packets, the symptom eventually occurred. After we checked captured packets, we could, however, see many suspicious packets, and we could not then locate the causes.

We finally found that there were many SNMP packets during the failure occurs. We then suspected the process, "IP SNMP," again. Figure 3 shows received suspicious SNMP packets. As shown in Figure 3, packets included "GetBulk" request that were common to be utilized for Distributed Reflection Denial of Service (DrDos). Those requests also includes requests to try to obtain an object ID, ..1.3.6.1.2.1.4.21.1.1 (ipRouteDest) or .1.3, and the community was " public" that were usually used for attacks. Interestingly, some of these packets did not have UDP check summary (i.e., check summary field were zero). It was common to receive those packets. After the failure occurred several times after starting packet captures, we, however, noticed that we always saw that those many SNMP packets received, and the process, "IP SNMP," seemed to consume CPU usage.

4. Discussions

4.1 Temporal recovery by reboot

As described in section 3.3, the core switch temporally recovered when we reboot the core switch. These phenomena confused us to locate the actual causes. These phenomena, however, are very interesting because this means that some attackers may stop after attacks fail due to routing errors or something.

4.2 Packets punt to a CPU

As described in section 3.6, many packets are *punt* to a CPU of the core switch. This is necessary for L3 switch or router to generate ICMP message. These packets should be rate-limited to be punt to CPU in order to preserve a CPU of a network equipment.

4.3 Redundancy of network equipment

As described before, we switched over a routing engine of the core switch. We, however, were confused by this switching over because this feature had a bug to synchronize a configuration.

4.4 Unbelievable architecture of firewall

What we were surprised is that our firewall cannot share the same network segment between different virtual firewall context. Because of this feature, our core network received malicious SNMP packets.

5. Related Works

V. Paxson[6] has analyzed Distributed Denial-of-Service (DDoS). He, however, has not found that there is the case where the core switch is vulnerable against even normal SNMP.

6. Concluding Remarks

We have presented that even an popular network equipment in a enterprise network may be vulnerable against normal SNMP packets. We have then reported our case and how to find and how to recover from failures.

Reference

[1] D. Harrington, R. Presuhn and B. Wijnen, " An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks," RFC3411, December, 2002.
[2] IEEE SA, "802.1D-2004 - IEEE Standard for Local and metropolitan area networks: Media Access Control (MAC) Bridges," 2004.
[3] Ruby association, "A Programmer's best friend Ruby," Available

on line: https://www.ruby-lang.org/, accessed on 19th April 2016.
[4] C. Hedrick, "Routing Information Protocol," RFC1058, June 1988.
[5] David C. Plummer, "An Ethernet Address Resolution Protocol," RFC826, November 1982.
[6] V. Paxson, " An analysis of using reflectors for distributed denial-of-service attacks," ACM SIGCOMM Computer Communication Review, Vol. 31, Issue 3, pp.38-47, July 2001.