A Case Study of Botnet Attacks against Linux Systems

James Yu and Imad Al-Ajarmeh
College of Computing and Digital Media
DePaul University
Chicago, IL 60604
jyu@cdm.depaul.edu and iajarmeh@cdm.depaul.edu

Abstract

This paper presents a case study of Botnet attack against a Linux server at the CDM network lab. After studying the characteristic of the attack, we developed a comprehensive security measure to prevent similar attacks in the future. There are many reports about Botnet attacks which are considered the most serious and prevalent Distributed Denial of Service (DDoS) attack. Academic environments are particularly vulnerable to these attacks due to the requirements to support student learning and practicum. In this case study, we quarantined the compromised machine and traced the software code and data traffic to study the attacking mechanism and its impact to the whole network. The problem cause is identified as vulnerability in a web tool for system administration.

From this Botnet attack experience, we implemented a comprehensive security measure to protect potential future attacks. The key measure is to create a Virtual Private Network (VPN) environment which restricts lab machines with public Interface access. In addition to VPN, we conduct regular audits on the lab environment. The audit includes (a) monitoring system log files to identify abnormal behavior, (b) monitoring system processes to identify improper usage, (c) generating 15-min traffic reports to identify *spikes*, and (d) keeping the system and web services up-to-date with new patches. The case study presented in this paper is to raise the awareness of Botnets, and the implemented security measures could be helpful for administrators to prevent similar attacks.

1. Introduction

The growing popularity of the Internet also generates new waves of security attacks which are more advanced than their predecessors. Of these new attacks, Botnet receives a lot of publicity due to its magnitude and capability to cause severe financial loss [1] through Distributed Denial of Service (DDoS), phishing, spam or identity theft. Bot (short for robot) is an autonomous program running on a compromised computer, and these Bots could be controlled remotely by a hacker to launch attacks at a specified time against a specified target(s). According to [2], Botnets are collections of computers infected with malicious code that can be controlled remotely through a command and control infrastructure. Usually, the command and control is achieved via Internet Relay Chat (IRC) channel, peer-to-peer (P2P) connections or web traffic. The size and danger of Botnet threat is now widely recognized [3], but the actual number of machines involved in Botnet activities is difficult to estimate. Many estimate the bot-infected computers are in the order of millions and it grows rapidly. A security report shows that there were "*52,771 active bot-infected computers per day during the 1st half of 2007.*"[1] As more and more machines have broadband Internet connections, there are also more machines become bot targets.

In the network lab at DePaul University, we recently experienced a Botnet attack where a Linux server was compromised by hackers and became a Command and Control (C&C) server. After we identified the abnormal traffic from the server, we immediately quarantined the compromised machine from the network and then performed detailed analysis to identify the problem cause and its effect. This paper presents the analysis of this Botnet attack, and the protection measures to prevent similar attacks in the future.

2. Botnet Attacking Scenario

The Botnet attack starts with a hacker (known as the BotMaster) who plants malware on victim's computers as illustrated in Figure 1.

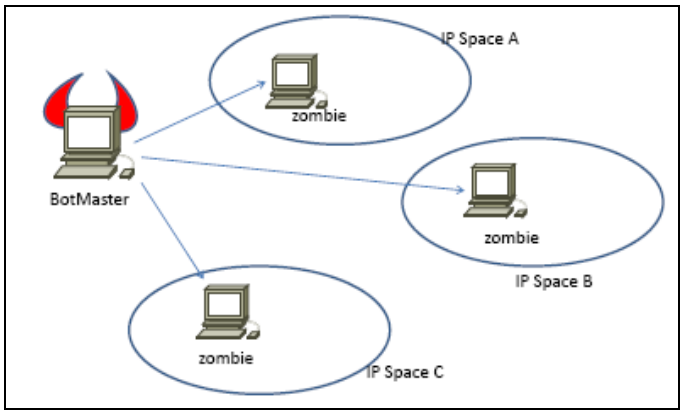

Figure 1: BotMaster Implants Malware on Zombie Machines

[1] A security presentation at the Tech-Security conference in Chicago on January 29, 2009.

The BotMaster may exploit vulnerability on victims' computers and plant the malware, or a victim may inadvertently download the malware to his/her computer. These machines are also known as *zombie* machines. The malware is sometimes referred to as the *bot* (short for robot). A zombie machine could be on a private network which is behind a Network Address Translation (NAT) router. A zombie machine may infect other machines on the same IP space, which is within an Internet Service Provider (ISP) network or within an organization IP network. Infected machines become new zombies which could infect more machines [4]. Each IP space now has an army of zombies as illustrated in Figure 2. Note that new zombies may continue recruiting more zombies.

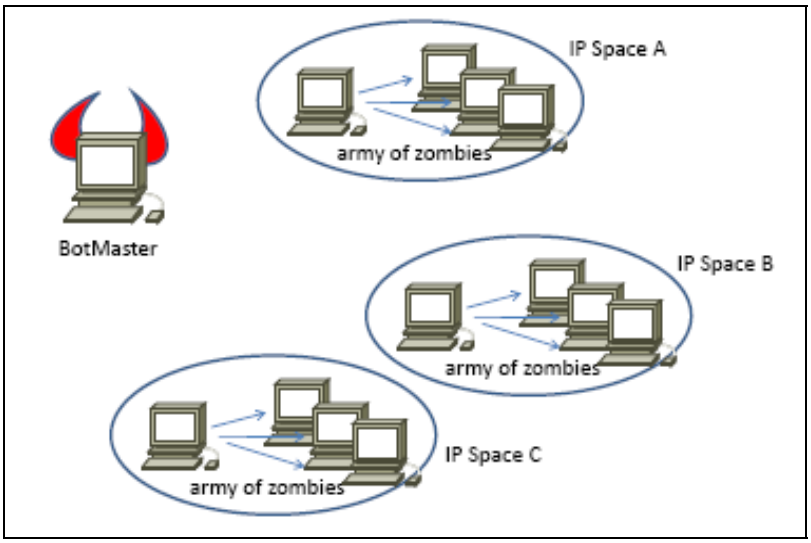

Figure 2: More Machines are Infected and Become Zombies

The BotMaster also needs to establish a Command and Control (C&C) center as illustrated in Figure 3. The purpose of the C&C center is to control the zombie machines and instruct them to download software (or malware) that contains commands and instructions for attacks [5].

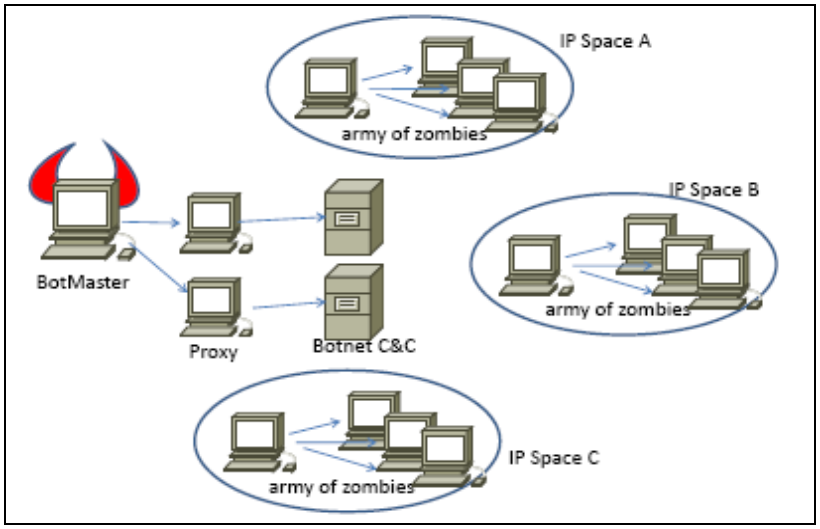

Figure 3: BotMaster Creates Distributed C&C Servers via Proxies

The C&C server is always on the public domain so that the zombie machines can communicate with C&C directly. The new generation of Botnets has the following characteristics:

1. The 1st generation Botnet always uses the Internet Relay Chat (IRC) channel for communication [6]. Because IRC is known for its problem, many firewalls block IRC traffic (usually on TCP port 6677). New Botnets may use a web proxy which is difficult for the firewall to block [2]. It may also use Point-to-Point (P2P) connections or any communication mechanism that could fool the firewall.
2. The BotMaster does not access the C&C server directly; instead, BotMaster usually connects to the C&C server via a proxy or a series of proxies. Since the access is via a proxy, it is difficult to trace the traffic back to the BotMaster.
3. Instead of a single C&C server, BotMaster may establish multiple C&C servers via multiple proxies. Each C&C server serve a distinct function on the Botnet. For example, one C&C server is to receive commands from the BotMaster; one C&C server is to distribute commands to the zombies; and one C&C server could be the download center for the zombie machines to retrieve or update its software (malware).
4. A BotMaster may sell his/her C&C to another BotMaster or to exchange C&C with one another. Because of the potential financial gain, the problem of Botnet is widespread on the Internet.

After the Botnet setup is complete, the BotMaster may set a time to launch an attack on a target machine (the victim) as illustrated in Figure 4. The zombie machines may start updating their malware from a C&C server, and then run the malware against the target machine.

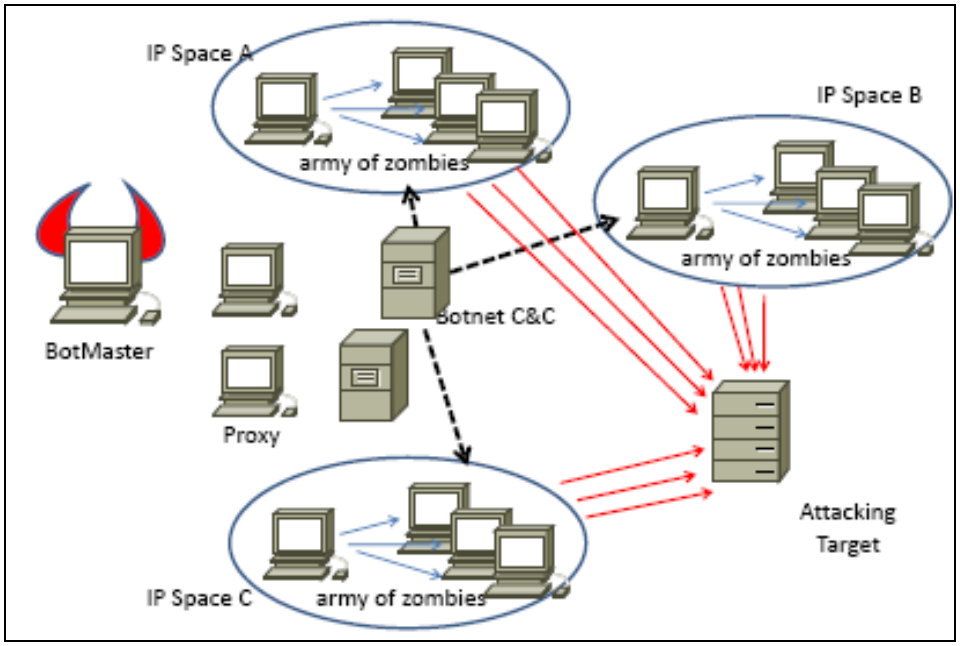

Figure 4: BotMaster Launches Botnet attacks

As we can see from Figure 4, there is no communication with the BotMaster or C&C servers during an attack, so it is extremely difficult to trace back to the BotMaster. It should also be noted that the target machine may not have any malware on it, so anti-virus program would not be able to prevent a victim from being a target of the Botnet attack. The effect of such an attack is known as the Distributed Denial of Service (DDoS) attack as the victim could be out of service during the attack. BotMaster could easily use different malware (downloadable from C&C) to launch different attacks on the same target or on different targets.

3. Case Study of the Botnet Attack

The incident of the Botnet attack at our network lab occurred in the winter quarter of 2008. This section provides a detailed anatomy for the attack, and presents the symptoms, mechanisms, and results of the attack.

3.1 Event Description

The incident was discovered when the University IT team informed the lab administrator that a lab web server was being used actively to spread malware throughout the campus network. The compromised machine was immediately removed from the network. However, it was difficult to determine the problem cause, so we put the machine back on the network and enabled the monitoring tool (*tcpdump*) on this machine. For 40 hours, we did not observe any abnormal behavior. At the 40^{th} hour after the machine was back on the network, we observed a large amount of *http* traffic coming out of the machine. We enabled and disabled *httpd* a few times and were able to easily reproduce the attacking scenario. After collecting sufficient data, the machine was quarantined to a private environment for further analysis.

As a standard practice at the lab, we configured a monitor tool running the Linux *top* command every 15 minutes. The log of the *top* commands does not show any abnormal process, but we observed several spikes of the *httpd* process as shown in Figure 5. Normally, we observed 7-10 *httpd* processes (in one *top* output). When the machine was under attack, we observed 150+ *httpd* processes in one *top* output. It is clear that the attack is related to the web service, and we suspect it may be related to a weakness of the Apache web server.

From the data packets collected using *tcpdump*, we observed more than 60,000 IP packets (60MB of data) in a 3-minute interval during the attack. The packet analysis shows that the 3-way TCP hand shaking is complete, so this is not a typical TCP-SYN attack. The suspected packets are all HTTP packets. The distributed nature and size of the traffic has the effect of a typical Distributed Denial of Service (DDoS) attack given that our web server was not designed to handle this amount of traffic. However, at this point we are not sure whether the Linux server is the source or the victim of the attack (or both). Although we could remove the symptom of the problem, we are still not sure of the problem cause nor are we sure of the intention of the attack.

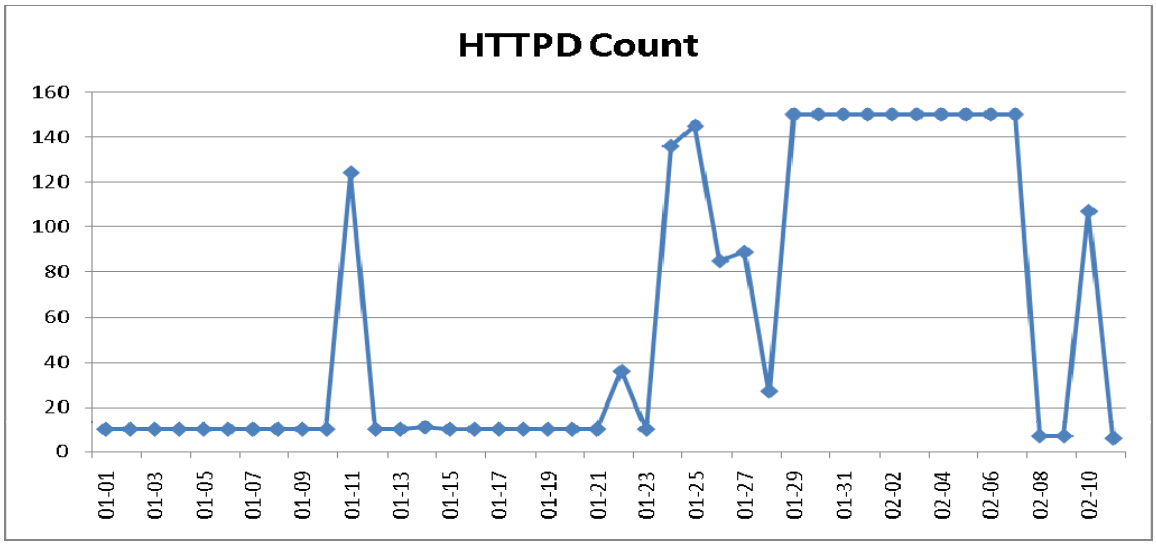

Figure 5: Max HTTPD Count of Individual *top* commands (per day)

.

After the Linux server is moved to a private network, we conducted further analysis of the problem and examined individual HTTP packets. From the HTTP GET packets, we could see its content and URL of the requested files (see Figure 6).

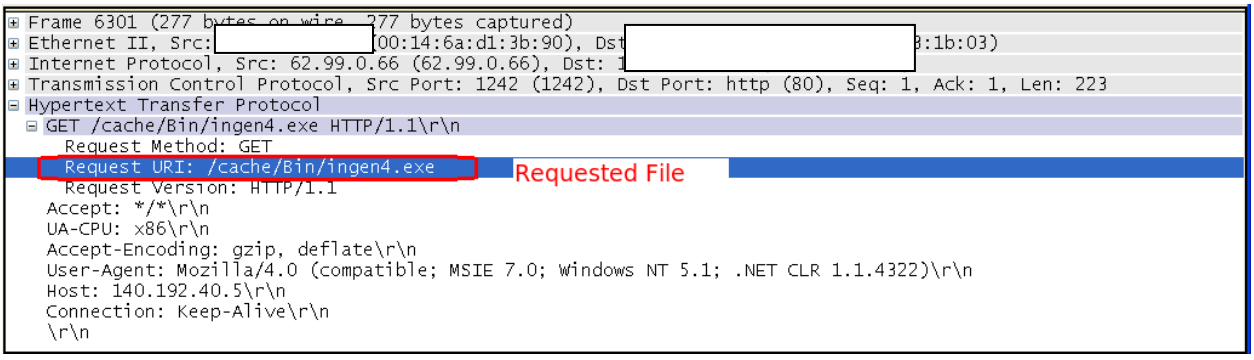

Figure 6: URL and File of HTTP GET

The HTTP GET requests come from a wide range of IP addresses, and it is an indication of an army of zombies involved in this attack. The URL in the HTTP GET shows that it is trying to retrieve files in the root directory of the web document (not the system root directory). In this directory, it has the fingerprint of the malware implanted by the hacker. There are multiple files of malware, and their filenames have no significance as we cannot find any information about them. The web log shows failed execution of the malware on the Linux server since these files (malware) are valid win32 files and hence they are executable on MS Windows but not on Linux environment. This is the reason that the University IT team complained that the compromised machine was sending malware all over the net. We exported one of the malware files to the MS Windows environment and confirmed it is malware as identified by our anti-virus program. Lastly, we also traced a hidden *cron* job running at */etc/.etc* which is likely created by the hacker. Again, this fingerprint matches the description of Botnet, where the sources are probably *zombie*

machines trying to download their command and control instructions, update their malware, or acquire new malware. They could also be potential zombie machines trying to download the Bot control engine placed by the hacker on our server. Although the symptom of the Botnet could be easily fixed, it is not clear how those malware files get onto the server, and who created the mysterious */etc/.etc* directory.

3.2 Problem Cause

After we identified the existence of this hacker-created directory, we did a thorough check of this directory (/etc/.etc) and the finding is alarming. This directory has two subdirectories mirroring the system */etc* directory structure, except that the information is for the hacker to access the Linux machine. We traced the hacker script to the *webmin* application[2] which is a web-based interface for system and network administration on Unix/Linux. The *webmin* application was installed by a student lab assistant to provide an easy-to-use service for the faculty and other administrators. However, no one was interested in the web interface for system administration, but the tool was never removed from the machine. Our analysis shows that the hacker script is a valid script used by *webmin* and its problem is well documented[3]. The report shows that a hacker can exploit the vulnerability of *webmin* and gain access to the Linux server:

> *"Anyone can grab any file from your server and in particular the passwords file /etc/shadow. Once a cracker has this file, he can run a brute force or rainbow attack to get the original Linux user passwords, so he could have login privileges including root and therefore fully compromising the server."*

With this finding, we conclude that the problem symptom (which is Bonet attack) and the problem cause (vulnerability in *webmin*). Because the root could potentially be compromised on this machine, we decided that the only effective measure to clean this machine is to reformat its disk. Our final decision is to permanently remove this Linux server from the public access and place it at a private environment for further investigation (and education) of intrusion data. All services on this machine are disabled, except for the login process. If this machine is to be put back for practical use, we will reformat its disk first.

3.3 Summary of the Botnet Attack

From the analysis of the incident, we construct the event sequence as follows:

1. The hacker exploited the vulnerability of *webmin* and gained access to the Linux server with a new user ID (id=0) which has the root privilege. However, the access was via web, not via *ssh*. Although we found many attempts to break into the system using *ssh* with this ID, however, there is no evidence that any of those login attempts succeeded.
2. The hacker planted the malware (multiple versions) on the root directory of web documents. The directory structure (similar to cgi-bin) could allow a web user to execute

[2] http://www.webmin.com/
[3] http://www.fduran.com/blog/webmin-vulnerability-exposing-your-password-or-any-file/

the malware on the Linux machine. However, the malware is win32 and not executable on Linux.

3. The hacker (BotMaster) sets up hundreds of zombie machines which make frequent HTTP GET requests to the malware through the web server.
4. One attacking scenario is that the hacker used the compromised Linux server as a distribution/download center for his/her malware. It is clear that the hacker has programmed the zombies to download the malware from our server instead of his/her own server so that he/she cannot be traced. The malware may also contain instructions and commands to update/upgrade the zombies.
5. Another attacking scenario is that the malware is the Bot command and control engine to recruit more Botnet members. It should be noted that using HTTP traffic to transfer malware is very effective for the hacker since the traffic could easily pass through most firewalls. This attacking scenario is a typical web-based Botnet in which zombies obtain command and control through a web server instead of IRC or peer-to-peer connections [7]. This type of Botnet is the most recent and most dangerous for the reasons mentioned above. Web-based Botnets also take advantage of the wide spread of web servers and vulnerabilities associated with them.
6. Because of the large number of zombies involved in the Botnet, the compromised Linux server sent a lot of HTTP packets and flooded the campus network. This unusual traffic spike triggered the alerts to the University IT team, where their network analyzer recognized malware in the HTTP packets. At this time, we are still not sure about the exact nature (behavior) of the malware, but it is irrelevant to this attack as the malware is not executable on the Linux machine.
7. It is a little embarrassing to report that the Botnet attack actually happened multiple times from the compromised Linux server. In the earlier cases, the scope of the attack was not as significant as the latest one, and we did not detect it. However, the log of the *top* command clearly showed the spike of the *httpd* process (see Figure 5) and a security audit would have detected this abnormal traffic pattern.

4. Corrective Actions

The purpose of the networking lab is to provide students with the opportunity to strengthen their understanding of technologies through hands-on exercises. The design of the networking lab is to support both local students and remote students who could perform hands-on exercises from any place at any time [8]. From this Botnet attack, we see the need to improve the security of the lab but we must not compromise any user requirements for functionality and ease-of-access to the lab. The functionality of the lab environment must be maintained; otherwise, it would defeat the pedagogical purpose of the lab. The capability to support remote access is also a *must* requirement as we have a growing population of distance learning students.

We are aware of many commercial tools that have features to detect and provide early warning of potential Botnet attacks. The majority of those tools is designed to detect specific command and control (C&C) mechanisms and could become ineffective when the C&C mechanism is changed [9]. We do not see the need of such commercial tools as the primary use of the lab is for

students to configure and manage various network devices. There are few applications running at the lab environment. The strengths and capabilities of the commercial tools (for Botnet prevention) are more appropriate for the enterprise environment, and their high cost (both procurement and management) is not justified to our lab environment. Our major concern is with the public access and management of student accounts, and not about user applications (which are very few). From this Botnet experience, we conducted a security audit of the lab environment and implemented a comprehensive security measure to protect against any potential future attack. Our approach to addressing the security issues and maintaining the full lab capabilities are given in the following sections.

4.1 Virtual Private Network (VPN)

The most important protective measure is to create a Virtual Private Network (VPN) environment as illustrated in Figure 7. The VPN gateway is a server-grade Linux machine with an open source VPN solution (OpenVPN) [10]. The authentication of student accounts is controlled by the security server which is also an open source solution (OpenLDAP) on a server-grade Linux machine [11].

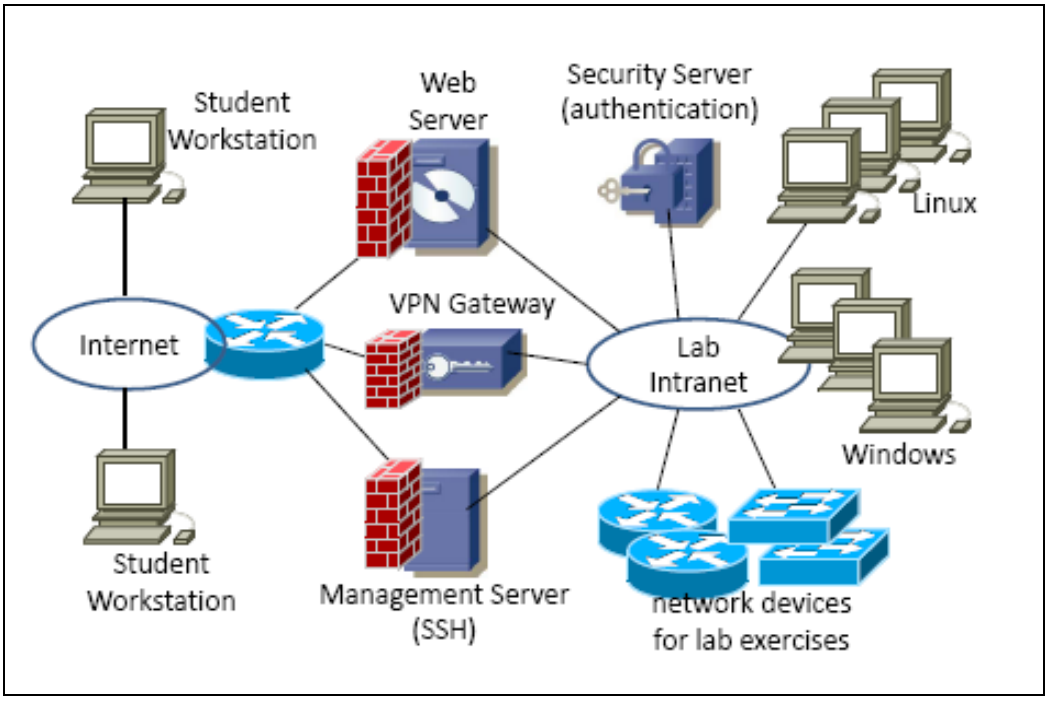

Figure 7. VPN Lab Environment (All servers are Linux Machines)

The authentication process between the VPN gateway and the authentication server is over the lab private network. The security server also supports authentication to other lab Linux machines. Students are required to install the OpenVPN client and download a security profile to establish a VPN connection to the private lab environment. The security profile expires at the end of each quarter to prevent potential abuse by hackers. After the VPN configuration, students

have the administrative (root) privilege to configure the network devices and Linux machines on the private lab environment. Machines used for lab exercises do not have public Internet access.

Before this Botnet incident, we used a single Linux machine to support various lab work: lab gateway, web server, research (running network simulation), network management, and application development. As a result, it is difficult to set an effective firewall policy or to control the processes running on the machine. From the Botnet experience, we decided to dedicate a single function to each lab machine and to minimize the interface to the public Internet. For example, all research work is moved to the private lab intranet environment. We still maintain a web server to provide course information; however, students do not have accounts on the web server. We also created a dedicated VLAN for network management and provisioned a management station connected to this management VLAN. Faculty and lab administrators can use SSH to access the management station and then perform various administration functions via this management VLAN, including the web development. As illustrated in Figure *7*, these three servers (VPN, Web, and SSH/management) have an interface to the Public Internet, and we implement a strict firewall policy on these servers. The VPN gateway allows only the VPN traffic. The web server opens only one TCP port for the web traffic, and the management server allows SSH only. The development of the web services and upload of the web files must be performed from the private side of the web server. This environment would not allow any IRC traffic and would not accept file upload from the public Internet. Therefore, it significantly reduces the risk of being exploited by a hacker to become a C&C server or to become a zombie machine.

4.2 Security Audits

In addition to the new network design for security protection, we conduct regular audits on the servers connected to the public Internet.

1. We develop an alert system to monitor system log files to identify abnormal behavior,
2. We enable Simple Network Management Protocol (SNMP) on the lab Ethernet switch (which is on a private network) and configured MRTG [12] which shows the traffic report at the 15 min interval (see Figure 8). The purpose of traffic report is to identify abnormal (spikes) traffic patterns.
3. It is important to periodically review and analyze the *top* log and to decide if there is any abnormal behavior. We have created a few signatures (patterns) to indicate the potential security attacks. It should be noted that the incident could have been detected earlier if we had a regular audit of the *top* and server logs.
4. The web server is probably the weakest link in the network configuration, and there are constant exploitations of the weaknesses of the Apache web server. Therefore, it is important for the lab administrator to keep the server up-to-date with new patches and be aware of any security alerts.

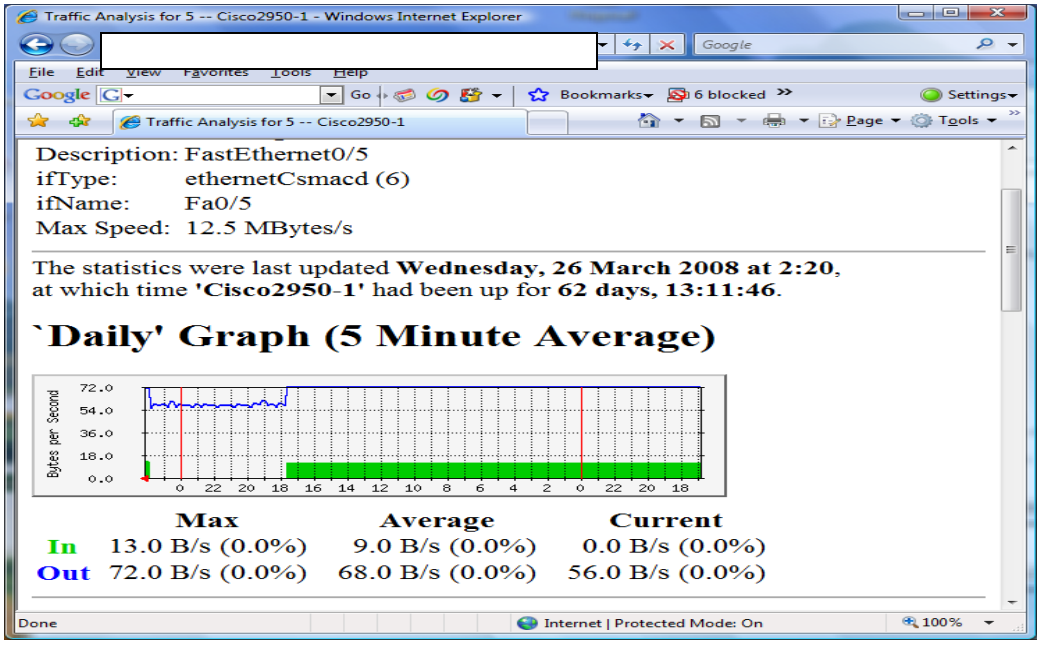

Figure 8. MRTG Traffic Report

5. Conclusion

It is well known that academic environment is a popular target for security attacks, especially for this new wave of Botnets. The case study of Botnet reported in this paper shows the vulnerability of the lab environment, and the potential risk of being exploited by a hacker. In this incident, a Linux server became the download/distribution center to spread malware on the Internet, and it also has a high risk to become a zombie. Although we do not know if the hacker has used our environment to launch a real attack against another target, the potential damage to our network environment is very alarming. The analysis presented in this paper is expected to help other educational institutes be aware of the risk and implement protective measures against the Botnets. In addition, we also conclude that *reactive* methods for network security are not sufficient and *proactive* methods are required. Disabling unused services, monitoring active services, keeping system and services up-to-date with patches, and conducting regular audits are important proactive measures for security protection.

References

[1] Michael Riezenman, "Botnets Pose Growing Silent Threat," IEEE News, February 2009.

[2] Nicholas Ianelli, and Aaron Hackworth, "Botnets as a Vehicle for Online Crime," CERT Coordination Center, December 1, 2005

[3] Bradford Paul, and Blodgett Mike, "Towards Botnet Mesocosms," Proceedings of the USENIX First Workshop on Hot Topics in Understanding Botnets (HotBots), Cambridge, MA, April, 2007.

[4] Sandvine whitepaper, **"**Dynamic Botnet Detection," June 2006.

[5] Moheeb Abu Rajab, Jay Zarfoss, Fabian Monrose, and Andreas Terzis **"**A multifaceted approach to understanding the botnet phenomenon**",** Proceedings of the 6th ACM SIGCOMM conference on Internet measurement, Rio de Janeriro, Brazil, 2006

[6] Evan Cooke, Farnam Jahanian, Danny McPherson, "The Zombie roundup: understanding, detecting, and disrupting Botnets," Proceedings of the Steps to Reducing Unwanted Traffic on the Internet Workshop, p.6-6, Cambridge, MA, July, 2005

[7] Craig A. Schiller et al, "Botnets: the Killer Web App.", 1st ed., Syngress Publishing, 2007, ch. 3, pp. 77-95

[8] James Yu, "An Innovative Lab Environment for Supporting Hands-on Networking Exercises to Distance Learning Students," The IASED International Conferences on Internet Multimedia Services and Applications (IMSA), Honolulu, August 2007.

[9] Guofei Gu, Roberto Perdisci, Junjie Zhang, and Wenke Lee. "BotMiner: Clustering Analysis of Network Traffic for Protocol- and Structure-Independent Botnet Detection." In *Proceedings of the 17th USENIX Security Symposium (Security'08)*, San Jose, CA, 2008

[10] OpenVPN, http://www.openvon.net.

[11] OpenLDAP, http://www.openldap.org

[12] Multi-Router Traffic Grapher (MRTG), http://www.mrtg.com/