Mobile Netw Appl (2013) 18:122–128
DOI 10.1007/s11036-012-0375-1

A Novel Approach to Analyzing for Detecting Malicious Network Activity Using a Cloud Computing Testbed

Junwon Lee · Jaeik Cho · Jungtaek Seo · Taeshik Shon · Dongho Won

Published online: 9 May 2012

Abstract Recent developments have caused the expansion of various cloud computing environments and services. Cloud computing environments have led to research in the areas of data processing, virtual environments, and access control. Information security is the most important research area for these environments security. In this study, we analyzed typical example of network testbeds, which have been used for malicious activity data collection and its subsequent analysis. Further, we propose an effective malicious network application testbed, which is based on a cloud system. We also verified the performance of our new testbed by comparing real malicious activity with the cloud-based testbed results.

Keywords mobile cloud · cloud test bed · cloud network anomaly

J. Lee · D. Won
School of Information and Communication Engineering, Sungkyunkwan University,
Suwon, Republic of Korea

J. Lee
e-mail: junwo1lee@gmail.com

D. Won
e-mail: dhwon@security.re.kr

J. Cho
Graduate School of Information Security, Korea University,
Suwon, Republic of Korea
e-mail: chojaeik@korea.ac.kr

J. Seo
The Attached Institute of ETRI,
Suwon, Republic of Korea
e-mail: seojt@ensec.re.kr

T. Shon (✉)
Division of Information Computer Engineering, Ajou University,
Suwon, Republic of Korea
e-mail: tsshon@ajou.ac.kr

1 Introduction

The advent of cloud computing means that PC environments are changing to cloud computing environments, where a large set of computers(or virtual environment, cloud) can connect to applications, storage, OS, and the requisite IT resources over the Internet at any time they are needed. Thus, the IT services paradigm has been changed [1]. The advantage of cloud computing is that users have the ability to store information and manage it cost effectively at any place or time via a network that uses a free program, which requires no professional knowledge of specific software or hardware. There are several ways of analyzing and checking whether malicious code is present in an existing network. A diffused experiment can be conducted inside a network after it has been shut down. In such experiments, the test bed and physical network are formed and a malicious application spreads while network packet data are collected. However, the massive scale of this experimental method means that vast amounts of time and space are consumed [2]. In this study, an effective cloud network testbed was used in experiments to analyzing for detecting malicious code with lower time and space requirements.

This paper contains the following sections: section 2 provides an explanation of the cloud system structure; section 3 summarizes the existing physical testbed; section 4 describes the basic cloud system malicious code testbed; section 5 analyzes the test results; and section 6 states our conclusions.

2 The cloud computing service

Cloud computing provides virtualized IT resources via Internet technology where users can borrow IT resources (software, storage, servers, and networks) as required, which support real-time scalability depending on the services available, such as pay-as-you-go computing.

Compared with traditional computing environments, cloud computing is based on IDC connections between multiple users, which provides a greater amount of resources [3].

'Highly scalability 'and' abstract computer resources' are two main features of cloud computing IT resources. 'High scalability' refers to the processing capacity where computing resources can be increased or decreased to ensure that transactions are processed rapidly and flexibly. 'Abstract computer resources' means that users do not need to consider where the computer that processes the calculations is located [4].

2.1 Global cloud computing

Amazon EC2 (Elastic Compute Cloud) and S3 (Simple Storage Service) provided by IaaS (Infrastructure as a Service), and Google AppEngine and Apps provided by PaaS (Platform as a Service), are the major business models [5]. Google has extended into Android over recent years because it is aimed at mobile devices. Microsoft Azure platform, Live Services, and Win Mo are major mobile terminal solutions. Salesforce.com is focused on SaaS trading as Salesforce. Sun, IBM, and HP are hardware companies that have also produced cloud computing solutions.

Venture capital companies in Wuxi, China, have been developed to spur the innovation of technology to support the industrialization of cloud computing and to ensure that computing resources are available for development and testing. This is part of the Chinese Government's strategy of converting the Chinese economy into a service-led economy based on IDC with IBM solutions, rather than producing technology. To support the future growth of software companies in China, 11 regional development agencies have been formed to ensure software development across China. Wuxi provides virtualized infrastructure and development platforms, while it also produces service management technology to supports the developer.

The Japanese government also announced the 'Kasumigaseki cloud' project in 2015, in which government IT systems will develop a single cloud infrastructure. The Japanese Secretary launched a public-private joint initiative, 'Cloud Computing Research,' during June 2009 to enhance Japanese owned technology and regulations, which will provide support for companies [6]. Nippon Telegraph and Telephone Corporation (NTT) have also announced a cloud computing-related R & D investment plan, which will develop a next generation cloud platform. NTT plans to provide cloud-computing services nationwide, in conjunction with 18 IDC Centers.

NTT has entered the SaaS, PaaS, and IaaS markets as NTT Communications and NTT Data, where it aims to drive this venture by attracting application business to the cloud platform. In addition, NTT has developed a common capability platform for SaaS providers, where SaaS providers can leverage their capabilities via common authentication and fee collection systems without developing their own environments.

The NTT group aims to solve the problem of security by coupling networks (VPN) and cloud services, thereby strengthening the reliability of the platform [7].

3 Previous malicious network testbeds

The basic network testbed contains dozens to hundreds of physical networks and systems. There are many nodes and virtualized nodes; hence, the virtual environment can include additional dozens or hundreds of nodes. This section introduces the standard testbed used by DARPA for Intrusion Detection Evaluation research, which is comprised of a testbed and the physical network [8, 9]. We also describe the testbed used for studying the spread of general malicious worms at Korea University

3.1 DARPA intrusion detection evaluation

MIT Lincoln Laboratory (MIT/LL) developed a normal and attack dataset for the evaluation of intrusion detection algorithms. Air Force network data were collected and reconstructed to form the MIT/LL's network dataset [10]. MIT/LL's data collection and reconstruction was supported by DARPA from 1998 to 2005. MIT/LL is still researching the development of network data for other uses, which is available to the public. A large network area data was collected from the Air Force network for the generation of network data set. This network dataset is used to evaluate intrusion detection algorithms. The datagram portion of network data packets has been erased to prevent the leakage of confidential information [11].

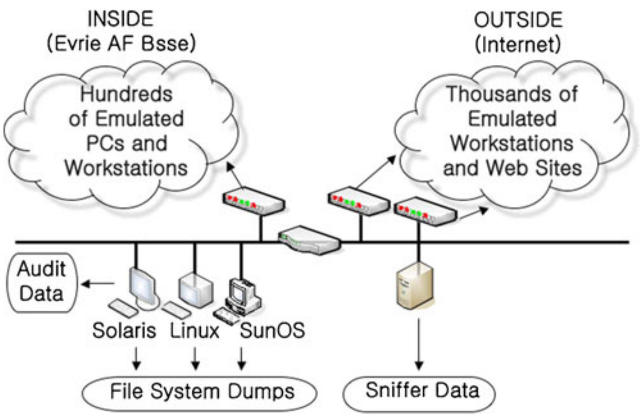


Fig. 1 The experimental network structure used by MIT Lincoln Lab

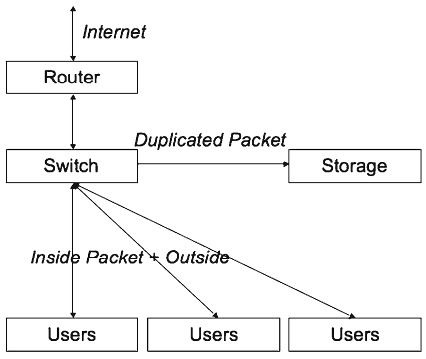


Fig. 2 Small network area data collection

The total experimental process involved in MIT/LL data-set is summarized as follows.

(1). Environmental analysis of the original network: the behavior of the system and users that form the original network.

(2). Collection of network data packets: collection of forwarding data from a specific network and receiving data from another network.

(3). Design of an isolated network testbed: design of an isolated network testbed to resend the collected network data and transform the datagram portion of the data.

(4). An experimental attack on the isolated network testbed: several types of test attacks.

(5). Packet reconstruction (presumption): a random mix of normal and attack data to derive sample data.

The first step was to analyze the original network environment in detail. A system's operating system and other variables had to be confirmed when analyzing the environment. The behavior of users must also be recorded and analyzed. The second step was to capture the packet data from the real network. Complex methods must be used to prevent data loss and collect real network packet data from a huge network in real time. However, MIT/LL's report contained no references to any data loss. Sent and received data related to a user's behavior was also included in the collection. The third step was the reconstruction of the packet data. Datagram

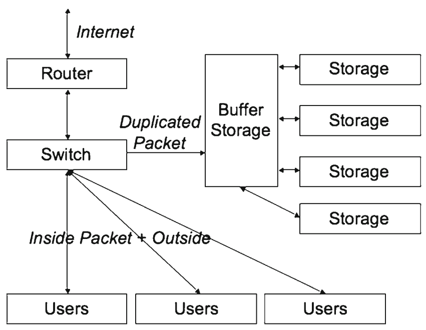


Fig. 3 The data collection system architecture for huge area network

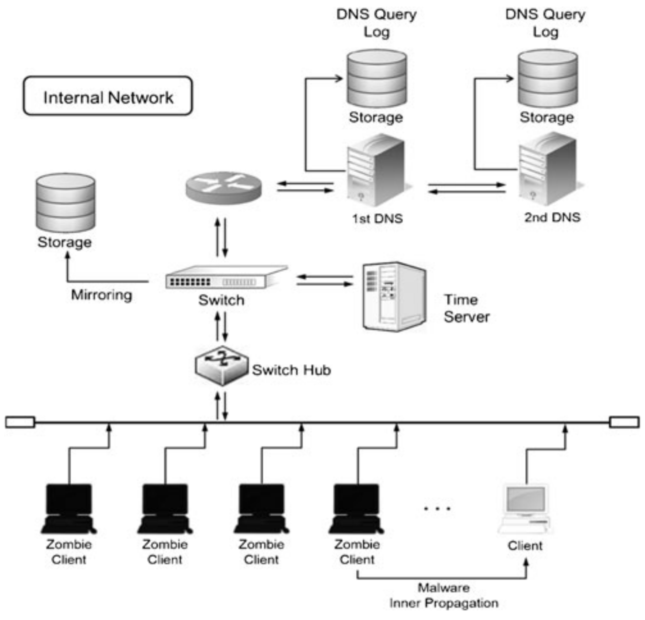


Fig. 4 Virtual isolated malicious software Testbed

transformation was performed after collecting the network packet data, as described above. The fourth step was the collection of attack network packets. Attack data was separated from the Air Force network packets. A further attack experiment was performed after the initial attack and the packets were compared and analyzed.

Thus, we separated the normal data and the attack data from the reconstructed data in our experiments. The normal data was used for transfer and transformation in the testbed while attacks were performed with attack automata, which were made by analyzing the attack data. The last step was the presumption

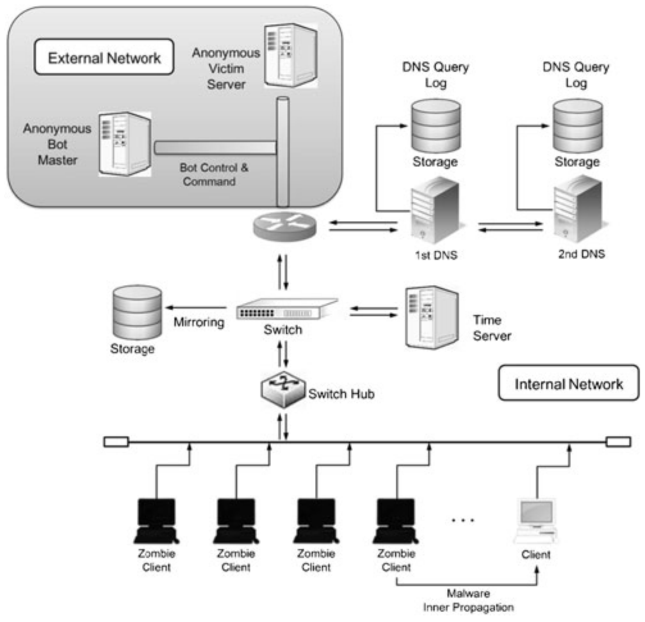


Fig. 5 Open virtual Testbed structure

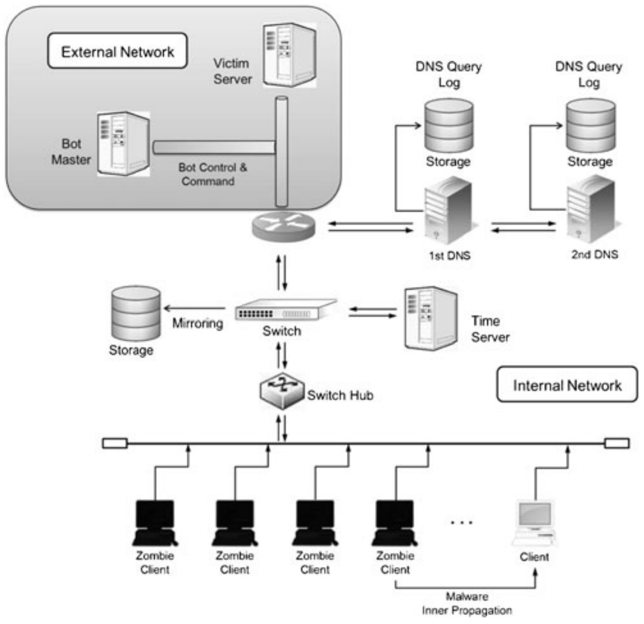


Fig. 6 Open virtual Testbed structure with a specific victim system

stage of this research. The network packet data that is available to the public contains several types of data including: 10 % sampled data, and 10 %:90 % or 30 %:70 % attack to normal data ratios. The algorithm used to achieve this is not available to the public but there must be an appropriate way of mixing the packet data that is provided to the public.

3.2 Reconstruction of the data collected from the air force network

The data collected from the Air Force network was reconstructed using the experimental network shown in Fig. 1.

In this experiment, the data was composed to resemble the hundreds of personal computers and thousands of workstations that actually produced the data. The collection of network packet data was conducted at the front and back layer of routers. The collected network data was classified into sent network data and received network data. The composition of the data set had a mixture of normal and abnormal behavior. Normal behavior and abnormal behavior of the Air Force network was automatically executed according to the personal profile of the user. Furthermore, the automatic execution used the profiles of automata to generate packets for behavior expression.

3.3 Worm propagation testbed at Korea University

Networks can be divided into two categories, i.e., small networks and huge networks area network. This research defines small networks as those containing 100 hosts or less, whereas huge networks contain 1,000 hosts or more. The type of network data also differs at the collection point. In the first method, data is collected the end host. In the second method, data is collected by a router in the middle area. Finally, a Wide Area Network (WAN) gateway has an edge network router. The source network path also divides each collection pointer. The research experiment on network data analysis used a small network and a huge network without specifying the network data collection method.

3.3.1 Collection of small network area data

Small network area data was collected using an edge node router for collection, as mentioned above. The data was collected from a small network below the router. The data was composed of data generated by the small network, response packet data from the small network, and all non-intentional data.

The network structure of the small network area used for data collection was as shown in Fig. 2.

Data collection used Tcpdump, which is open source software that uses a standard Pcaplib.

3.3.2 Collection of huge network area data

Huge network area data was collected using a WAN gateway as the edge network router. Network data was collected

Table 1 Packet collection results from the first experiment

	Packet Count	First packet	Last packet
Mirror	903,661	2011-07-22 15:41:32	2011-07-26 14:33:43
1st DNS(192.168.1.3)	633,170	2011-07-22 14:47:11	2011-07-26 14:29:42
2nd DNS(192.168.1.4)	303,839	2011-07-22 14:47:11	2011-07-25 16:59:15
Time server(192.168.1.2)	242,706	2011-07-22 14:53:19	2011-07-26 14:32:27
Client13(192.168.1.13)	264,150	2011-07-22 14:46:16	2011-07-26 14:35:23
Client14(192.168.1.14)	334,134	2011-07-22 14:44:44	2011-07-26 14:32:21
Client23(192.168.1.23)	263,911	2011-07-22 14:49:01	2011-07-26 14:36:08
Client52(192.168.1.52)	239,423	2011-07-22 15:35:43	2011-07-26 14:38:35
Client53(192.168.1.53)	239,452	2011-07-22 15:34:27	2011-07-26 14:38:02
Client54(192.168.1.54)	239,597	2011-07-22 15:32:56	2011-07-26 14:38:53

Table 2 Packet collection result from the second experiment

	Packet Count	First Packet	Last Packet
Mirror	1,561,462	2011-07-26 17:28:28	2011-07-28 18:19:23
1st DNS(X.X.217.192)	839,718	2011-07-26 17:07:14	2011-07-28 18:45:34
2nd DNS(X.X.217.193)	41,623	2011-07-26 17:04:36	2011-07-28 19:09:10
Time server (X.X.217.191)	784,318	2011-07-26 17:08:57	2011-07-28 18:50:32
Client13 (X.X.217.198)	892,881	2011-07-26 17:32:37	2011-07-28 18:22:37
Client14 (X.X.217.199)	881,972	2011-07-26 17:31:32	2011-07-28 18:21:12
Client23 (X.X.217.204)	884,145	2011-07-26 17:34:05	2011-07-28 18:23:46
Client52(X.X.217.221)	771,807	2011-07-26 17:40:53	2011-07-28 18:23:26
Client53(X.X.217.222)	811,034	2011-07-26 17:40:21	2011-07-28 18:23:12
Client54(X.X.217.223)	771,616	2011-07-26 17:39:17	2011-07-28 18:22:47

by recording 1998 host computer behaviors (Fig. 1), which was different from the method used in the ‘Collection of Small Network Area Data.’

A large capacity router was used for huge network data collection on the WAN gateway. The system contained a fast packet mirroring switch to facilitate that collection of a huge amount of data. Huge storage middleware was used as a buffer when the collection of data was faster than recording allowed, while a large capacity processing system was used for final storage as shown in Fig. 3.

The amount of data collected averaged 2 GB per minute. The total collection time was 2 weeks and the total amount of data collected was 1.5 TB. The header of the network packet was the final recorded data in this research.

4 Cloud-based testbed

Physical resources found at the lowest level are the most difficult to share among multiple users. The various hardware specifications do not need to be visible to users; hence, we need a level of abstraction. These physical resources are actually abstracted first. The function of the Resource Abstraction and Virtualization layer is to convert physical resources into virtual resources. Virtual resources are contained in a resource pool. Resources can be allocated to users from the resource pool, and then released back into the resource pool when they are no longer needed [12].

In this study, we set up a cloud computing environment using the open source-based Xen [13] 4.1.0 and Linux 2.6.18 to collect a packet in the cloud virtual environment Windows XP sp1 was installed on each guest OS. Each of the 40 guest OSs infected a Bot, which a Bot master controlled. A packet was also collected from each guest OS zombie PC using a tcpdump on Dom-0, which generated and controlled a guest OS.

4.1 Virtual testbed capacity

We created virtual environment that could use each computing node. The virtual computing node has to be similar to the real Microsoft Windows XP sp1 environment; hence, it could be infected with the malicious application, which was Zeus and modified version.

The virtual network should also be similar to the real environment; hence, a 1 GB network was simulated in the cloud with 1 router and 1 switch module. The switch and route module model were based on the Cisco Catalyst 3,000 series.

Other virtual nodes were a primary domain name server and a secondary domain name server. Malicious software such as Bots usually try to connect to a C&C server while waiting for a secondary attack order, as with a distributed denial of service attack. To produce these DNS nodes, we used the Microsoft Windows XP sp1 system for all nodes

Table 3 Packet collection results from the third experiment

	Packet count	First packet	Last packet
Mirror	387,619,316	2011-07-29 11:00:03	2011-08-01 11:08:11
1st DNS (X.X.217.192)	2,421,275	2011-07-29 10:59:44	2011-08-01 11:15:34
2nd DNS(X.X.217.193)	2,398,922	2011-07-29 11:00:22	2011-08-01 11:16:46
Bot server (X.X.217.204)	3,685,171	2011-07-29 11:01:39	2011-08-01 06:39:11
Client21 (X.X.217.198)	3,351,240	2011-07-29 11:01:36	2011-08-01 06:07:57
Client22 (X.X.217.199)	28,873,821	2011-07-29 11:01:38	2011-08-01 11:02:18
Client24 (X.X.217.204)	19,444,043	2011-07-29 11:01:38	2011-08-01 11:00:59

except the domain name server nodes. These DNS nodes used an Ubuntu Linux-based kernel 2.6.34.

4.2 Framework

Our basic virtual isolated testbed structure is shown in Fig. 4.

Malicious code infection in the primary system was assumed to be a zombie because it infected 40 nodes and the client distribution with malicious code, i.e., one of the Zeus variant programs that are directed at the Microsoft Windows system. Input and output packet data in the virtual environment were compressed and saved in the pcaplib format. The timeserver was configured to synchronize the virtual node time using a cloud system's standard time. Each node was also assigned a private IP address that was similar to the actual network configuration. The routing module was assigned like an actual network to accept an input, while output sequences changed packets according to the routing module when packet collision occurred in a large network. The switch module mirrored all inputs, output data for the entire network, and provided a node to guarantee a data transfer rate.

4.3 Malicious bot testing steps

The effectiveness of the malicious code when collecting a network's configuration was divided into three steps.

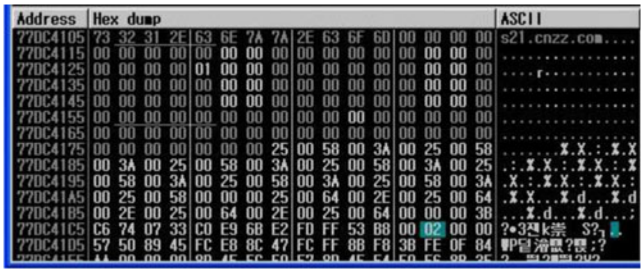
The first is the basic closed network architecture, which is shown in Fig. 5. We collected the actions of a Zeus Bot when trying to connect to C&C during this study.

The second step is organized as shown in Fig. 5.

Second configuration allowed a network to respond when a connection attempt outside C&C was made by infected zombies. An open network allows a DNS node to respond to collected data that differs from the first configuration and it collects C&C communication actions.

The third step is organized as shown in Fig. 6.

The third step collected data using a modified Zeus Bot until it attacked a specific system. It was configured to perform a DDoS attack against a victim system using modified version of the malicious codes from the second stage.



```
Address  |Hex dump                                       |ASCII
77DC4105 |73 32 31 2E|63 6E 7A 7A|2E 63 6F 6D|00 00 00 00|s21.cnzz.com....
77DC4115 |00 00 00 00|00 00 00 00|00 00 00 00|00 00 00 00|................
77DC4125 |00 00 00 00|01 00 00 00|00 00 00 00|00 00 00 00|....r...........
77DC4135 |00 00 00 00|00 00 00 00|00 00 00 00|00 00 00 00|................
77DC4145 |00 00 00 00|00 00 00 00|00 00 00 00|00 00 00 00|................
77DC4155 |00 00 00 00|00 00 00 00|00 00 00 00|00 00 00 00|................
77DC4165 |00 00 00 00|00 00 00 00|00 00 00 00|00 00 00 00|................
77DC4175 |00 00 00 00|00 00 00 25|00 58 00 3A|00 25 00 58|.......%.X.:.%.X
77DC4185 |00 3A 00 25|00 58 00 3A|00 25 00 58|00 3A 00 25|.:.%.X.:.%.X.:.%
77DC4195 |00 58 00 3A|00 25 00 58|00 3A 00 25|00 58 00 3A|.X.:.%.X.:.%.X.:
77DC41A5 |00 25 00 58|00 00 00 25|00 64 00 2E|00 25 00 64|.%.X...%.d...%.d
77DC41B5 |00 2E 00 25|00 64 00 2E|00 25 00 64|00 00 00 3B|...%.d...%.d...;
77DC41C5 |C6 74 07 33|C0 E9 6B E2|FD FF 53 B8|00 02 00 00|?•3젠k崇  S?┐
77DC41D5 |57 50 89 45|FC E8 8C 47|FC FF 8B F8|3B FE 0F 84|WP딜滑跟?瑕;?
```

Fig. 7 Fixed domain addresses in the reversed binary code

The third step collected all of the zombie's actions while it attempted to connect, link, or command C&C, which were different from the previous steps

5 Comparing binary activities and network activity

5.1 Result of the experiments

Collected data from the different experiments based on attacks were configured to a general pcap library format, which was the same as the packet data.

In the first example, the data was collected from a virtual closed network, as shown in Table 1.

The second experiment used an open network, which recorded communications by the zombie client outside the Bot C&C. The results were as shown in Table 2.

In the third experiment (results shown in Table 3), the Victim system was set up on a Linux server that was serviced on an actual network. This included a DDoS attack on the open network.

5.2 Packet analysis results

The results were assembled by reverse engineering the technology to determine whether the network packet data collected on the actual network was similar to the virtual experiment network set up on the cloud network.

We confirmed that the same domain name came out on the packet and binary at the same time during binary analysis, to ensure that the domain name was from the actual

Table 4 Abnormal DNS query from a zombie client

No.	Time	Source	Destination	Protocol	Info
239	2079.385807	163.152.217.201	163.152.217.192	DNS	Standard query A s21.cnzz.com
240	2079.396578	163.152.217.193	163.152.217.192	DNS	Standard query response CNAME cache.cnzz.com CNAME a.cnzz.com.lxdns.com CNAME a.cnzz.z.
243	2088.539236	163.152.217.201	163.152.217.192	DNS	Standard query A zs21.cnzz.com
244	2088.638373	163.152.217.193	163.152.217.192	DNS	Standard query response A 219.232.243.104 A 219.232.241.176

packet data. The code producer had already configured this correctly. Domain s21.cnzz.com was recorded in the second experiment, which was a known malware distribution domain [14]. The domains shown in Table 4 were identified in the packet collection.

If the binary code contained a configuration with a malicious feature, we confirm this domain based on the memory contents at the time of infection. We verified the memory contents as shown in Fig. 7.

6 Conclusion

In this study, we used a cloud system in a malicious code-spreading experiment. Current network experiments that are used to analyze the spread of malicious code require considerable time and computational resources. The proposed cloud system can make significant savings in terms of time and resource costs, while it allows data collection and analysis muck like an actual network. In the future, we aim to configure a more optimal method of packet generation, check packet occurrences in a time sequence, and we will compare an actual network's packet distribution with a virtual network's packet distribution using the virtual network environment presented in this paper. This will facilitate more effective analysis of malicious code distribution.

Acknowledgments This paper is extended research based on an already published MCCTA 2011 conference paper. This version of the extended paper includes further experiments and updated sections, which were the most significant issues.

References

1. Armbrust M, Fox A et al (2009) Above the clouds: a Berkeley view of cloud computing. Tech Report No. UCB/EECS-2009-28, http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html
2. Sharif M, Yegneswaran V et al (2008) Eureka: a framework for enabling static malware analysis, LNCS Volume 5283
3. Mell P, Grance T (2009) The NIST definition of cloud computing. Nat Instit Stand Technol 53:50, http://csrc.nist.gov/groups/SNS/cloud-computing/cloud-def-v15.doc
4. Moreno-Vozmediano R, Montero RS and Llorente IM (2009) Elastic management of cluster-based services in the cloud, In Proceedings of the 1st workshop on Automated control for datacenters and clouds (ACDC '09) ACM
5. Vaquero LM, Rodero-Merino L, Caceres J and Lindner M (2008) A break in the clouds: towards a cloud definition. SIGCOMM Comput Commun Rev 39
6. The Kasumigaseki Cloud Concept. http://www.cloudbook.net/japancloud-gov.
7. Hiroaki H, Kamizuru Y, Honda A et al (2010) Dynamic IP-VPN architecture for cloud computing, Information and Telecommunication Technologies (APSITT)
8. Lippmann RP, Fried DJ, Graf I et al (2000) Evaluating intrusion detection systems: the 1998 DARPA off-line intrusion detection evaluation, DARPA Information Survivability Conference and Exposition
9. Lee W, Stolfo SJ (2000) A framework for constructing features and models for intrusion detection systems. ACM Trans. Inf Syst Secur 3
10. McHugh J (2000) Testing intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory. ACM Trans Inf Syst Secur
11. Mahoney MV, Chan PK (2003) An analysis of the 1999 DARPA/Lincoln laboratory evaluation data for network anomaly detection, LNCS
12. Sotomayor B, Montero RS, Llorente IM, Foster I (2009) Virtual infrastructure management in private and hybrid clouds. IEEE Internet Comput vol.13, no.5
13. Xen Cloud platform. http://xen.org
14. ThreatExpert. http://www.threatexpert.com/report.aspx.