LARX: Large-scale Anti-phishing by Retrospective Data-Exploring Based on a Cloud Computing Platform

Tianyang Li*, Fuye Han*, Shuai Ding† and Zhen Chen‡
*Department of Automation
Email: {lty09, hfy10}@mails.tsinghua.edu.cn
†Department of Computer Science and Technologies
Email: dingshuai@csnet1.cs.tsinghua.edu.cn
‡Research Institute of Information Technology (RIIT)
Email: zhenchen@tsinghua.edu.cn
Tsinghua University
Beijing, China

***Abstract*—Anti-phishing is an intriguing challenge for Internet users especially for online business or e-pay users. Tracking phishing is quite difficult because most victims are not instantly aware of phishing attacks until their accounts are compromised, and monetary losses occur. Most web browsers provide plug-ins to protect users from phishing websites, but a client side solution cannot provide detailed forensic information on phishing attacks. In this paper, we propose an offline phishing detection system named LARX (acronym for *L*arge-scale *A*nti-phishing by *R*etrospective data-e*X*ploration). LARX uses network traffic data archived at a vantage point and analyzes these data for phishing attacks. All of LARX's phishing filtering operations use cloud computing platforms and work in parallel. As an offline solution for phishing attack detection, LARX can be effectively scaled up to analyze a large volume of trace data when enough computing power and storage capacity are provided.**

***Index Terms*—computer forensics; anti-phishing; cloud computing; Amazon Web Services; Eucalyptus;**

I. Introduction

As the Internet plays a more and more important role as our main information infrastructure, online businesses and e-pay services on the Internet are booming due to their convenience and benefits for users. The underground economy based on Internet scam and fraud is also booming. Those attackers who benefit from such scam and fraud activities commit more and more e-crime, such as spam and phishing attacks, etc.

Phishing attack is an intriguing yet practical problem because sensitive information may be stolen (e.g. usernames, passwords, and credit card numbers). It is estimated that phishing attacks cause roughly one billion US dollars worth of damage annually. Not only the users but also the financial institutions such as banks and e-pay systems have been impaired by phishing attacks.

Much work has been done on phishing attack protection [1]–[3]. In order to protect web browser users from phishing attacks, plug-ins comparing visited URL with blacklisted URLs are already provided by main-stream web browsers. Google also provides Google Safe Browsing API [4] for developers to check a URL using Google's phishing website blacklists.

An example of research on the life cycle of phishing websites is given in [5], and the results show that URLs associated with the phishing websites are quite ephemeral, thus making the collection of phishing attack forensics difficult [6]–[11]. Moreover, the concealed nature of such attacks makes it even harder to detect and prevent.

Gregor Maier et al. [12] propose a traffic archiving technology for post-attack analysis in Bro. Using Time Machine, network trace data is archived and can be fed back to the IDS with knowledge of current attacks to detect attacks more effectively.

With a similar idea, we propose an offline phishing attack forensic collection and analysis system. This system is targeted to solve the following challenging problems:

(1) How to collect the original data and search for the phishing attacks within the data.

(2) How to handle the large volume data in a reasonably short period of time.

In this paper, we propose LARX, acronym for *L*arge-scale *A*nti-phishing by *R*etrospective data-e*X*ploration, an offline phishing attack forensics collection and analysis system. First, we use traffic archiving in a vantage point to collect network trace data. Secondly, we leverage cloud computing technology [13]–[15] to analyze the experimental data in a way similar to the "divide and conquer" scheme. We used two existing cloud platforms, Amazon Web Services [16], [17] and Eucalyptus [18]. A physical server is also used for comparison. All of LARX's phishing filtering operations are based on a cloud computing platform and work in parallel. Finally, as an offline solution, we conclude that LARX can be effectively scaled up to analyze a large volume of network trace data for phishing attack detection.

Compared with LARX's design and implementation, it is interesting that Monarch [19], a similar system for real-time

978-1-4577-0638-7 /11/$26.00 ©2011 IEEE

URL spam filtering for tweets, is proposed by K. Thomas et al. at ICSI and UC Berkeley. Compared with Monarch, LARX focuses on offline exploration of a large volume of network trace data.

The main contribution of our paper is two-fold. First, we propose the idea of archiving network trace data for further collection and analysis of phishing attack forensics. Secondly, we implement a practical solution to collect data trace and analyze these data parallel on a cloud computing platform to search phishing attack in a reasonable short time interval. Because service provided by cloud computing platforms can be effectively scaled up in a rather easy manner, LARX can be effectively scaled up to to collect and analyze phishing attack forensics in a large amount of network trace data by leveraging the scalability provided cloud computing platforms. The results show that our solution is economical for large scale forensic examination of phishing attacks in network trace data.

II. System Design

A. Computing Platform

Most of our work focuses on the network trace collection and data exploration. The underlying computing platform is cloud computing. Here we will introduce our cloud computing platform based on Amazon Web Services.

1) Cloud Computing Based on Amazon Web Services: Amazon's EC2 and S3 are used in LARX. The main purpose of using Amazon Web Service is to test the feasibility of using cloud computing platforms in LARX. Taking into account user privacy and legal issues, we anonymized the network trace data before uploading the data to Amazon's S3 service.

2) Cloud Computing Based on Eucalyptus: In this section, we introduce our cloud computing platform using on Eucalyptus.

Fig. 1 shows the basic architecture of Eucalyptus. It consists of several web services components - the Cloud Controller (CLC), Walrus, Storage Controller (SC), Cluster Controller (CC) and the Node Controller (NC). In our single cluster setup using Eucalyptus, a single physical server runs all of the front end web services (all services except the NCs). The Node Controller is put on another physical server, which is the compute component in our cluster. Fig. 2 shows the topology we used in our single cluster setup.

Our Eucalyptus cluster consists of two identical physical servers. The physical server has the following hardware configuration: Intel Core 2 Quad Processor (2MB cache, 3GHz, 1.333 GHz FSB), double channel 4GB DDR3, Intel G41 and ICH7R chipset, and Intel 82574L network chipset.

B. Data Processing

1) Network Trace Data Collection: Our network trace data is collected during a period of about half a year at a vantage point. A modified version of Time Machine [12] is used in network trace data collection. The key strategy for efficiently recording the contents of high network traffic is to exploit the heavy-tailed nature of network traffic - most network connections are quite short, with the exception of

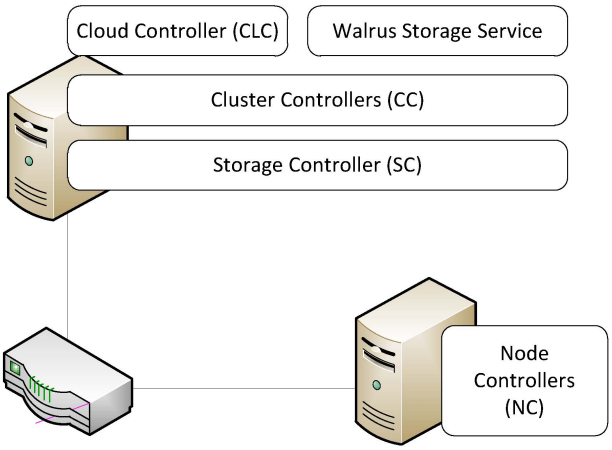

Fig. 2: Basic architecture of Eucalyptus

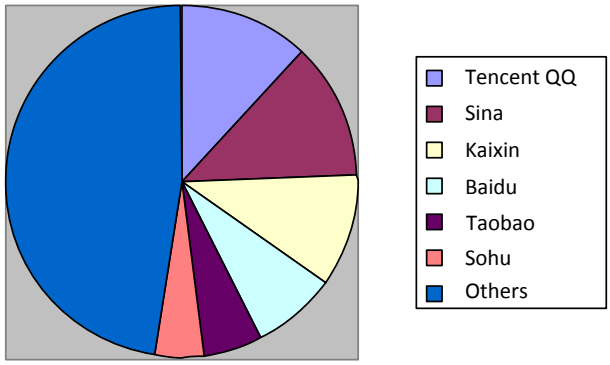

Fig. 3: URL distribution in a typical 512M network trace data block.

a small number of elephant connections (the heavy tailed connections) accounting for the bulk of total network traffic volume [12]. Thus, by recording only the first N bytes of each connection (the cutoff is 15 kilobytes in our case), we can record most connections in their entirety, while still greatly reducing the volume of network trace data that we must retain. For elephant connections, we keep only the beginning portion of such connections; however, in practical cases the beginning portion of such connections is the most interesting part because the beginning portion of such connections contain protocol handshakes, authentication dialogs, data item names, etc.

At the vantage point where network trace data is collected, approximately 2TB of traffic passes through each week. The amount of network trace data archived each week is roughly 100GB.

Network trace data archived by Time Machine consists of 512MB network trace data blocks. Typically, a 512M network trace data block contains of about 40K URLs. An example of URL distribution is shown in Fig. 3. Network traffic analysis using ntop [20] at the vantage point shows that roughly 35% of all network traffic passing through the vantage point is HTTP. Network traffic of different TCP protocols (mostly TCP based protocols and only a few UDP based protocols) at the vantage point is shown in Fig. 4.

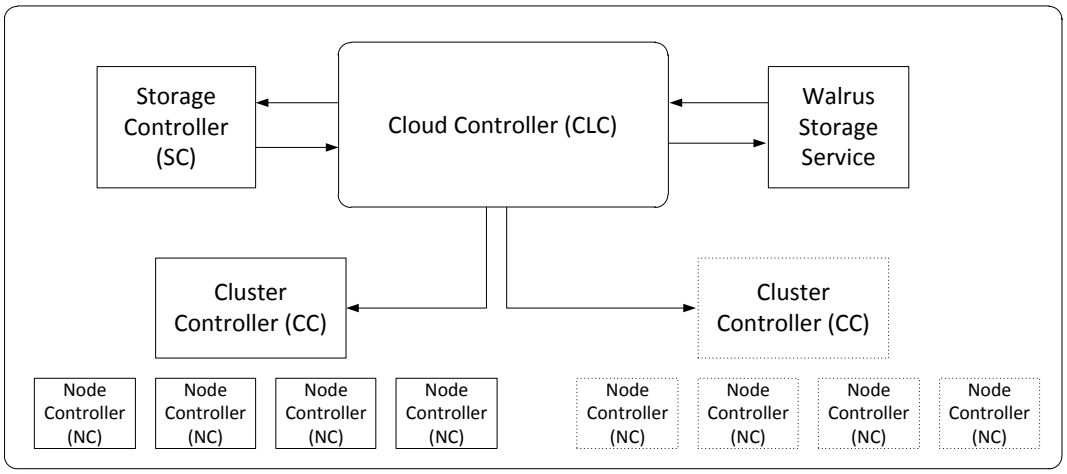

Fig. 1: Basic architecture of Eucalyptus

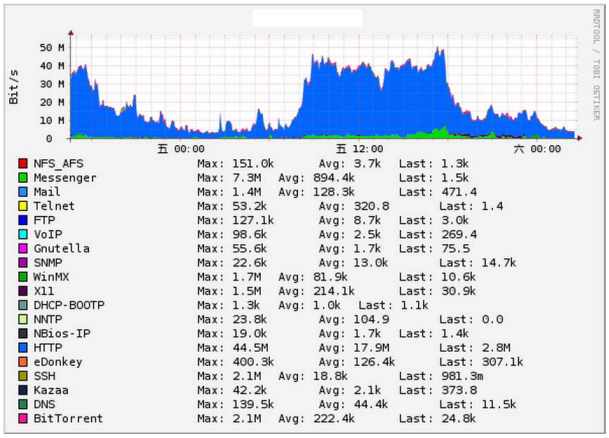

Fig. 4: Network traffic of different protocols (mostly TCP based protocols and only a few UDP based protocols).

2) Data Anonymization: To protect user's privacy and avoid legal issues in the research, the network trace data is anonymized by replacing IP addresses and other user information before processing the network trace data using Amazon Web Services.

3) Data Processing in LARX: The network trace data processing procedure is divided into different phases which are described as follows:

(1) File splitting: The size of each network trace data file created by Time Machine is 512 MB, and each network trace data is further divided into smaller parts for processing by using tcpdump [21], so that the amount of memory used during the extraction of data from TCP streams will not exceed the amount of physical memory.

(2) TCP stream reassembly: During this stage, TCP streams in the network trace data are reassembled using tcptrace [22].

(3) URL extraction: After reassembling TCP streams in the network trace data, grep [23] is used to find all URLs contained in the network trace data by searching for lines starting with "Referer: http://".

(4) URL matching: Extracted URLs are checked for phishing websites by using Google Safe Browsing API [4]. In order to check URLs for phishing websites, we use phishing website blacklists provided by Google. Google provides the first 32 bits of phishing websites' URLs' SHA-256 values for users to use. If a URL's SHA-256 value matches Google's blacklists , the full 256 bits hash value is sent to Google to to determine whether that URL points to a phishing website. During the process of matching URLs' SHA-256 hash values, a prefix tree is used because blacklists provided by Google is only 32 bits long and a prefix tree can efficiently match URLs' SHA-256 values in $O(n)$ time (where n is the number of URLs).

(5) Result reporter During this stage the final results on different machines are collected, and aggregated into a final report.

III. Experiment Results

We conducted our experiment using two existing cloud computing platforms - Amazon Web Services and Eucalyptus. A physical server is also used for comparison purpose. In our experiments, we analyzed 10GB of network trace data.

A. Amazon Web Services

Network trace data processing executes on an AWS EC2 small instance (m1.small) running Ubuntu Linux 10.04. As reported by the instance's operating system , the CPU is Intel(R) Xeon(R) E5430 (6MB cache, 2.66GHz), and the instance has 1.7GB of memory (with HighTotal: 982MB, and LowTotal: 734MB). Time spent in different process stages using Amazon Web Services are measured and concluded as shown in Table I.

It seems that the speed of prefix tree matching is quite fast and this time spending can be almost ignored. But before URL matching, it needs some time to download Google Safe Browsing's blacklists, the time spent is quite undetermined due to changing network conditions and Google servers response latencies.

It is worth mentioning that the m1.small instance in EC2 is memory constrained without swap partition support. It will

cause problems when the program consumes a large amount of memory (exceeding the memory usage limit) during network trace data analysis.

B. Eucalyptus

We used a c1.xlarge instance on our Eucalyptus platform running Ubuntu Linux 10.04. This instance has 4 CPUs and 2040MB of memory. Time spent in different process stages using Eucalyptus are measured and concluded as shown in Table I.

TCP stream reassembly takes a very long time in the instance on our Eucalyptus platform. The storage component and the compute component in our Eucalyptus cluster are located on different physical servers, and these different physical servers uses Ethernet for communication, thus TCP stream reassembly, which is disk I/O intensive because we use tcptrace, is greatly slowed down.

C. Physical Server

We also ran the network trace data block processing task on our own physical server platform with Intel Core 2 Quad Processor (2MB cache, 3GHz, 1.333GHz FSB), double channel 4GB DDR3 , Intel G41 and ICH7R chipset, and Intel 82574L network chipset. Time spent in different process stages using a physical server are measured and concluded as shown in Table I.

D. Comparison of Using Different Platforms

Our data show that the cloud computing platforms has low disk I/O performance compared to physical servers, and this is currently the greatest bottleneck in LARX. Also, when our Eucalyptus cluster case and physical server case are compared with the Amazon Web Services case, it seems that the CPU used in Amazon instance has better performance than QX9400 quad core CPU than our physical server.

E. Estimated Number of Instances

Assuming that time needed for a m1.small instance to handle a k bytes network trace data block in stage (2), stage (3), and stage (4) of data processing are t_1, t_2, t_3 (in seconds) respectively, the average traffic throughput is f bytes/s during the last 24 hours, and the traffic cut-off factor is h.

The number of total instances N in parallel needed to handle all of last 24 hours traffic is calculated as follows:

$$T = t_1 + t_2 + t_3$$

$$N = \frac{fTh}{k}$$

It is important to note that f and T are affected by several factors such as the percentage of HTTP stream in network traffic, the number of URLs in HTTP streams, users' behavior on the web, etc.

In the case of our physical server, assume that $f = 100\text{MB/s}$ (800Mbps) on 1 Gbps link, $h = 0.2$ (meaning that only 20% traffic is captured), each block is 200MB, $T = 40\text{s}$, then the number of physical servers in parallel is calculated as follows:

$$N = \frac{fTh}{k} = \frac{100 \times 40 \times 0.2}{200} = 4$$

In the case of Amazon EC2 (m1.small) instances, assume that $f = 100\text{MB/s}$ (800Mbps) on 1 Gbps link, $h = 0.2$ (meaning that only 20% traffic is captured), each block is 200MB, $T = 330\text{s}$, then the number of physical servers in parallel is calculated as follows:

$$N = \frac{fTh}{k} = \frac{100 \times 330 \times 0.2}{200} = 33$$

In the case of Eucalyptus instances, assume that $f = 100\text{MB/s}$ (800Mbps) on 1 Gbps link, $h = 0.2$ (meaning that only 20% traffic is captured), each block is 200MB, $T = 330\text{s}$, then the number of Eucalyptus instances in parallel is calculated as follows:

$$N = \frac{fTh}{k} = \frac{100 \times 2800 \times 0.2}{200} = 280$$

The number of Eucalyptus instances needed is too high, thus using LARX based on Eucalyptus is not practical.

Because of the cheap cost of using Amazon Web Services for data processing, by comparing these two results we can see that using cloud computing based LARX is a cost effective way to collect and analyze phishing attack forensics in a large volume of raw network trace data. Cloud computing platforms such as Amazon Web Services, Eucalyptus, and OpenStack [24] can provide scalability either remotely or locally, LARX can be effectively scaled up in cost effective ways by using cloud computing platforms rather than scaling up by using more physical servers.

F. Estimated Cost of Using Amazon Web Services for Network Trace Data Analysis

The cost of using Amazon Web Services consists of two parts - uploading network trace data to S3, and using instances in EC2 for network trace data processing.

In our real case, approximately 130GB of network trace data are collected on each day, therefore 4 Amazon EC2 m1.small instances can finish processing these network trace data on each day. The estimated monthly cost of using LARX on Amazon Web Services for our real case is approximately $700. Therefore, LARX based on Amazon Web Services is a cost-effective solution to the collection and analysis of phishing attack forensics in a large volume of raw network trace data.

G. Phishing Attack Results

After processing 10GB of network trace data, we failed to find any URL of a phishing website. There are several possible reasons: (1) Blacklists provided by Google Safe Browsing is incomplete. When K. Thomas et al examined large amounts of URLs in tweets [19], there is evidence showing that blacklists provided by different anti-phishing services are somewhat different in their coverage. (2) It is also possible that blacklists provided by Google Safe Browsing changes

TABLE I: Time spent in different stages of processing using different platforms. The size of network trace data block used is 200MB.

Platform	TCP stream reassembly	URL extraction	URL matching
Amazon Web Services	~ 290s	~ 47s	$1 \sim 2$s
Eucalyptus	~ 2750s	~ 45s	~ 5s
Physical server	$15 \sim 20$s	$15 \sim 20$s	~ 5s

very quickly, thus real-time blacklists might not contain the URLs of phishing sites in our network trace data. (3) Current browsers which have incorporated anti-phishing protection provide reliable anti-phishing protection. (4) Current computer security softwares provide reliable anti-phishing protection. (4) Users are aware of links that may lead to phishing websites, thus such links are not clicked by users.

IV. Conclusion

In this paper, we design and implement the LARX system to explore the large volume of network trace data to track phishing attacks. Traffic archiving is used at a vantage point to collect network trace data and cloud computing technology is leveraged to analyze the experimental data in parallel. Two existing cloud computing platforms, Amazon Web Services and Eucalyptus, are used. A hardware platform is also used for comparison purpose. All of LARX's phishing filtering operations are based on a cloud computing platform and operate in parallel, and the processing procedure is also evaluated. The results show that LARX can be a practical solution to the collection and analysis of phishing attack forensics in a large volume of raw network trace data by effectively leveraging the scalability provided by cloud computing platforms.

V. Acknowledgement

This work is supported by Ministry of Science and Technology of China under National 973 Basic Research Program Grant No. 2011CB302805, National High-tech Program under Grant No. 2007AA01Z468, Natural Science Foundation of China under Grant No. 90718040, and Tsinghua National Laboratory for Information Science and Technology (TNList) Cross-discipline Foundation.

References

[1] B. Wardman, G. Shukla, and G. Warner, "Identifying vulnerable websites by analysis of common strings in phishing URLs," in *eCrime Researchers Summit, 2009. eCRIME '09.*, 20 2009-Oct. 21 2009, pp. 1 –13.

[2] S. Li and R. Schmitz, "A novel anti-phishing framework based on honeypots," in *eCrime Researchers Summit, 2009. eCRIME '09.*, 20 2009-Oct. 21 2009, pp. 1 –13.

[3] R. Layton, P. Watters, and R. Dazeley, "Automatically determining phishing campaigns using the USCAP methodology," in *eCrime Researchers Summit (eCrime), 2010*, Oct. 2010, pp. 1 –8.

[4] "Google Safe Browsing API - Google Code," *http://code.google.com/apis/safebrowsing/*.

[5] S. Sheng, B. Wardman, G. Warner, L. Cranor, J. Hong, and C. Zhang, "An empirical analysis of phishing blacklists," in *CEAS 2009 - Sixth Conference on Email and Anti-Spam*, Jul. 2009.

[6] W. H. Allen, "Computer forensics," *IEEE Security and Privacy*, vol. 3, pp. 59–62, 2005.

[7] M. A. Caloyannides, N. Memon, and W. Venema, "Digital forensics," *Security Privacy, IEEE*, vol. 7, no. 2, pp. 16 –17, March-April 2009.

[8] F. Raynal, Y. Berthier, P. Biondi, and D. Kaminsky, "Honeypot forensics part 1: analyzing the network," *Security Privacy, IEEE*, vol. 2, no. 4, pp. 72 – 78, July-Aug 2004.

[9] ——, "Honeypot forensics, part ii: analyzing the compromised host," *Security Privacy, IEEE*, vol. 2, no. 5, pp. 77 –80, Sept-Oct 2004.

[10] N. Sklavos, N. Modovyan, V. Grorodetsky, and O. Koufopavlou, "Computer network security: report from MMM-ACNS," *Security Privacy, IEEE*, vol. 2, no. 1, pp. 49 – 52, Jan-Feb 2004.

[11] B. Carrier, "Digital forensics works," *Security Privacy, IEEE*, vol. 7, no. 2, pp. 26 –29, March-April 2009.

[12] G. Maier, R. Sommer, H. Dreger, A. Feldmann, V. Paxson, and F. Schneider, "Enriching network security analysis with time travel," in *Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, ser. SIGCOMM '08. New York, NY, USA: ACM, 2008, pp. 183–194. [Online]. Available: http://doi.acm.org/10.1145/1402958.1402980

[13] L. Barroso, J. Dean, and U. Holzle, "Web search for a planet: The Google cluster architecture," *Micro, IEEE*, vol. 23, no. 2, pp. 22 – 28, March-April 2003.

[14] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google file system," in *SOSP*, 2003, pp. 29–43.

[15] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," in *OSDI*, 2004, pp. 137–150.

[16] S. L. Garfinkel, "An evaluation of Amazons grid computing services: EC2, S3 and SQS," Center for Research on Computation and Society School for Engineering and Applied Sciences, Harvard University, Tech. Rep., 2007.

[17] "Amazon Web Services," *http://aws.amazon.com/*.

[18] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The Eucalyptus Open-Source Cloud-Computing System," in *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, ser. CCGRID '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 124–131. [Online]. Available: http://dx.doi.org/10.1109/CCGRID.2009.93

[19] K. Thomas, C. Grier, J. Ma, V. Paxson, and D. Song, "Design and Evaluation of a Real-Time URL Spam Filtering Service," in *Proceedings of the IEEE Symposium on Security and Privacy*, May 2011.

[20] L. Deri and S. Suin, "Effective traffic measurement using ntop," *Communications Magazine, IEEE*, vol. 38, no. 5, pp. 138 –143, May 2000.

[21] "TCPDUMP LIBPCAP public repository," *http://www.tcpdump.org/*.

[22] "tcptrace - Official Homepage," *http://www.tcptrace.org/*.

[23] "grep - GNU Project - Free Software Foundation (FSF)," *http://www.gnu.org/software/grep/*.

[24] "OpenStack Open Source Cloud Computing Software," *http://www.openstack.org/*.