A Virtualized Network Testbed for Zero-Day Worm Analysis and Countermeasure Testing

Khurram Shahzad, Steve Woodhead, and Panos Bakalis

Internet Security Research Laboratory, University of Greenwich, Chatham, Kent, UK
{K.Shahzad,S.R.Woodhead,P.Bakalis}@gre.ac.uk

Abstract. Computer network worms are one of the most significant malware threats and have gained wide attention due to their increased virulence, speed and sophistication in successive Internet-wide outbreaks. In order to detect and defend against network worms, a safe and convenient environment is required to closely observe their infection and propagation behaviour. The same facility can also be employed in testing candidate worm countermeasures. This paper presents the design, implementation and commissioning of a novel virtualized malware testing environment, based on virtualization technologies provided by VMware and open source software. The novelty of this environment is its scalability of running virtualised hosts, high fidelity, confinement, realistic traffic generation, and efficient log file creation. This paper also presents the results of an experiment involving the launch of a Slammer-like worm on the testbed to show its propagation behaviour.

Keywords: Worms, malware, Slammer, testbed, virtualization, VMware.

1 Introduction

Computer worms are a serious potential threat to network security. The high rate of propagation of worms and their ability to self-replicate make them highly infectious. A zero-day worm is a type of worm that uses a zero-day exploit; a publically unknown and un-patched vulnerability in network daemon software [1]. SQL Slammer is considered to be the fastest zero-day random scanning worm in history as it infected more than 75K hosts in less than 10 minutes [2]. The Stuxnet worm is a recent addition to this class of malware that spies on and subverts supervisory control and data acquisition (SCADA) systems and was the first network worm to include a programmable logic controller (PLC) rootkit [3].

Whilst other experimental malware testbeds have been reported, further improvements in this area will allow greater effort to be exerted in the development of malware defense techniques, such as worm countermeasures. Physical network setup [4, 5, 6, 7], simulation [8, 9, 10, 11], emulation [12, 13, 14] and virtualization [15, 16, 17, 18, 19, 20, 21] are some of key techniques previously reported for creating such experimental testbeds. The major challenges in implementing such a test environment are fidelity, scalability, confinement, realistic benign and malicious traffic generation, efficient log file creation, rebuild and configuration time, analysis and visualization, multi platform support, portability to different physically distributed testing environments, and flexibility in adjusting to experimental needs [17]. These diverse requirements of network and security experimental research are not well met by any single existing testbed. Competing methods remain popular because each tries to cover some portion of these requirements. Hence there is a need to design, implement and evaluate a novel virtual testing environment which incorporates increased granularity and instrumentation functionality.

With the aim of addressing these points, this paper presents the design, implementation and commissioning of a novel virtualized malware testbed, which employs VMware virtualisation technology and a range of open source software. We refer to the testbed as the Virtualized Malware Testbed (VMT). The novelty of this environment is its scalability, high fidelity, confinement, realistic traffic generation, and efficient log file creation. The paper also presents the results of an experiment involving the launch of a Slammer-like worm within VMT, to show the propagation behavior of the worm, and to validate the operation of the testbed.

The remainder of paper is presented as follows: Section 2 summarises the relevant previous work; Section 3 details the design, implementation and commissioning of VMT; Section 4 presents the experimental methodology and results of launching the Slammer-like pseudo-worm; and finally Section 5 concludes the paper with a discussion summarizing the findings and identifying any limitations, as well as summarising potential future work in this area.

2 Relevant Previous Work

Various network and malware testing environments have been built and proposed in the past which can be classified into the following categories:

- Physical machine testbeds
- Simulation testbeds
- Emulation testbeds
- Virtual machine testbeds
- Full system virtualization testbeds

2.1 Physical Machine Testbeds

Physical machine testbeds employ real physical hosts and network hardware for conducting research experiments. Emulab [4] was a distributed physical network setup, implemented for conducting research experiments. It consists of 218 physical nodes distributed between two US universities. Netbed [4] is a simulation environment implemented on Emulab that provides time and space sharing and employs ns-2 [11] for research and development. Emulab evolved into DETER [5], which is a cluster based testbed, consisting of high end workstations and a control software. It uses high-performance VLAN-capable switches to dynamically create nearly arbitrary topologies among the nodes. It was the first testbed to be remotely accessible through the public internet infrastructure. The 1998 DARPA off-line intrusion detection evaluation [6] and LARIAT [7] are also two physical machine testbeds sponsored by US Air Force and developed at the Lincoln Laboratory, MIT.

2.2 Simulation Testbeds

Simulation testbeds employ simulation tools to conduct network experiments. PDNS and GTNetS [8] were two network simulators for developing packet level worm models. These simulators allow an arbitrary subject network configuration to be specified consisting of scan rate, topology and background traffic. On the basis of defined input parameters, various types of outputs such as number of infected hosts in any given instance, sub-millisecond granularity of network event statistics or a global snapshot of the entire system are produced. Ediger reported the development of the Network Worm Simulator (NWS) [9], which implements a finite sate machine concept to simulate network worm behavior. Tidy et al [10] have reported a large scale network worm simulator aimed at the investigation of fast scanning network worms and candidate countermeasures.

2.3 Emulation Testbeds

Emulation testbeds provide a compromise between simulation and real world testing. ModelNet [12] and PlanetLab [13] are two emulated testbeds, implemented for general networking and distributed system experiments. In ModelNet, unmodified applications run on edge nodes, configured to route all their packets through a scalable core dedicated server cluster, by emulating the characteristics of a special target topology. PlanetLab was developed for the purpose of creating world-wide distributed systems, and has a dual nature of being used by developers and clients. Honeypots such as Honeyd [14] can also be classified as an emulation system as it has been used in many recent security systems for malware detection and capture.

2.4 Virtual Machines Testbeds

Virtual machine testbeds employ virtualization technologies as their main building block to conduct security and network experiments. ReVirt [15] is an advanced VM-based forensic platform which enhances individual virtual machines with efficient logging and replay capabilities, by redirecting log files from the guest OS to the host OS, for intrusion analysis purposes, thereby making it possible for malware analysis researchers to replay the malware exploitation process in an intrusion by intrusion fashion. Based on ReVirt [15] research, another platform VMWatcher [16] was developed that places the anti-virus system in the hypervisor layer, in order to be unreachable by the attacker. Research in SINTEFF ICT [17] has examined the effect of malicious software on a Windows XP workstation by utilizing Nessus [22] as the attacker, Wireshark [23] as a sniffer, Snort [24] as a NIDS and Sysinternals[25] to provide HIDS functionality.

2.5 Full System Virtualization Testbeds

Full system virtualization testbeds employ full virtualization; a technique that provides a type of virtual machine environment with complete simulation of the underlying hardware. vGround [18] has extended UML's virtual networking capabilities by supporting a VM-create-VM approach to automatically extend the network size. It uses Snort [24] and Bro [26] as NIDS and Kernort [27] as a HIDS to monitor worm target discovery and propagation. ViSe [19] provides a virtualization platform where malware exploits can be tested against the entire range of x86 based operating systems under controlled conditions, while being monitored by a NIDS. V-NetLab [20] has implemented a model based on DETER's [5] remote access capability by utilizing data link layer virtualization and packet encapsulation, thereby providing a more secure means of remote access to security related Testbeds. Golath [21] is a virtual network based on a Java Virtual Machine (JVM) and the Ultra light-weight abstraction level (ULAL). It provides a virtual environment to run any application written in Java, independent of the type of host operating system. System behavior can be monitored in this environment by adding different Java plug-in extensions.

2.6 Motivation

As far as the authors of this paper are aware, no previous virtualized malware testing environment has provided a scalable solution with a large number of virtual machines for security experiments by using VMware technologies. Isolation of the test environment from the management network with remote access also seems to be a problem. It is also noted that no previous reported work has produced infection and propagation analysis of any fast random scanning worm such as SQL Slammer in a real network with real world slammer exploitable conditions.

3 Virtualized Malware Testbed (VMT) Design and Capabilities

The Virtualized Malware Testbed (VMT) was designed with the intention of employing it in an investigation of exiting and hypothetical zero-day worms; and testing candidate worm countermeasures. Our goals of implementing VMT were experimental scalability, fidelity, repeatability, programmability, remote access and efficient log file creation.

3.1 Architecture, Design and Implementation

VMT uses VMware ESXi [28] as the core virtualization technology and Damn Small Linux (DSL) [29] as the main virtualised operating system. It also uses Quagga [30] to provide a software routing suite. VMware vCenter Server [31] provides a graphical user interface to manage VMware ESXi servers remotely.

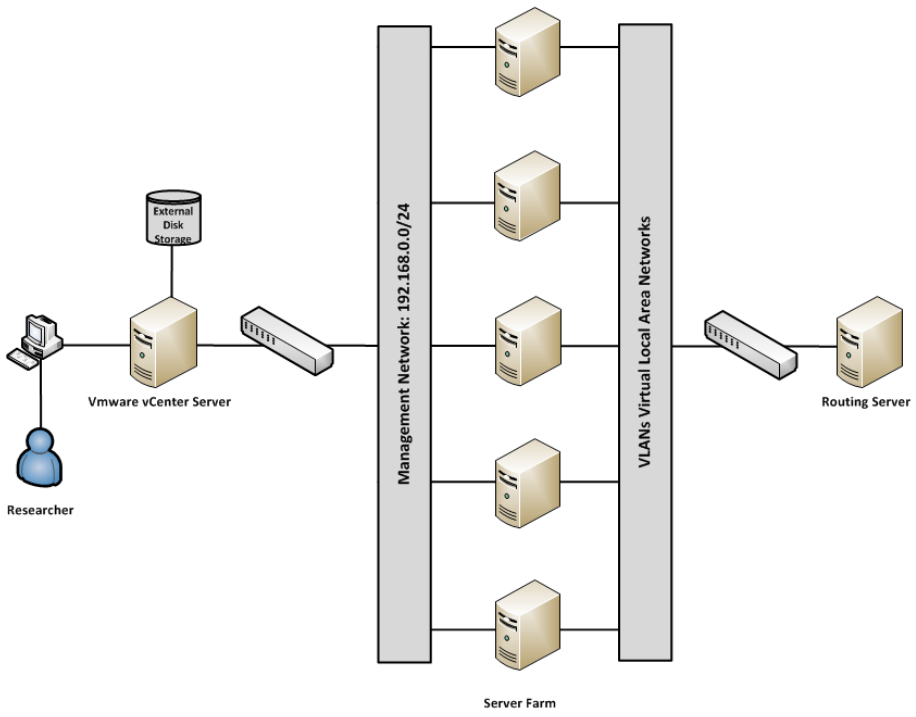

Fig. 1. Physical Network Setup

Figure 1 illustrates the physical architecture of VMT. It consists of a server farm with five servers, a management server and Ethernet switches. Each server in the server farm is running ESXi while the management server is running VMWare vCenter Server. One network interface card in each server farm machine is connected to a logically isolated management network along with the management server; thereby allowing access to all resources from one interface. Multiple virtual topologies can be created within the server farm by using virtual local area networks (VLANs) and Quagga. Each DSL virtual machine image is installed with 32 MB of memory and 1 GB hard disk. Table 1 summarizes the hardware and operating systems which make up the VMT infrastructure.

Table 1. VMT Hardware and Operating System Infrastructure

	Processors	**No of cores**	**Operating System**	**Memory**	**Storage**	**VMs**
Server 1	i7	6	ESXi 5.1	64 GB	1 TB	DSL, Ubuntu
Server 2	i7	4	ESXi 4.1	24 GB	1 TB	DSL, Ubuntu
Server 3	i7	4	ESXi 4.1	24 GB	1 TB	DSL, Ubuntu
Server 4	Xeon	4	ESXi 4.1	8 GB	512GB	DSL, Ubuntu
Server 5	Xeon	4	ESXi 5.1	8 GB	512GB	DSL, Ubuntu
Management Server	i7	4	Windows Server 2003 R2	8 GB	2 TB	N.A
Routing Server	i5	2	Ubuntu Quagga	4 GB	512GB	N.A

A minimum rebuild and configuration time are key goals of any security testing environment. VMware vCenter Server provides PowerCLI [32]; a command line interface tool that allows administrators to create simple and robust scripts to automate the main tasks, including virtual machines cloning.

4 Experimentation

4.1 Slammer-Like Pseudo Worm

In order to analyze the behavior of a SQL Slammer-like worm; we developed a network daemon which implements a Slammer-like pseudo-worm. This daemon listens on UDP port 1434 and upon receiving a datagram with an appropriate authentication string (included for safety reasons), it begins generating UDP datagrams addressed to port 1434 and to random IP addresses. The speed of datagram generation per second, and the pool from which the random destination IP addresses are chosen are configurable parameters. We have also implemented a logging server. At the point of "infection", the pseudo-worm daemon sends an infected time message to the central logging server.

4.2 Experimental Setup

We have setup a virtual test network comprising of a single Class A address space 10.0.0.0/8 but divided into four subnets; 10.0.0.0/10, 10.64.0.0/10, 10.128.0.0/10 and 10.192.0.0/10 as shown in Figure 2. These four subnets are connected through a central router by using RIP, configured on Quagga. Four further Quagga based routers are implemented (one for each subnet). One Linux based virtual machine is running in each subnet to provide a DHCP service. DSL is installed with the pseudo-worm daemon on each of the susceptible virtualised hosts. All hosts in the network are time synchronized by using the Network Time Protocol (NTP).

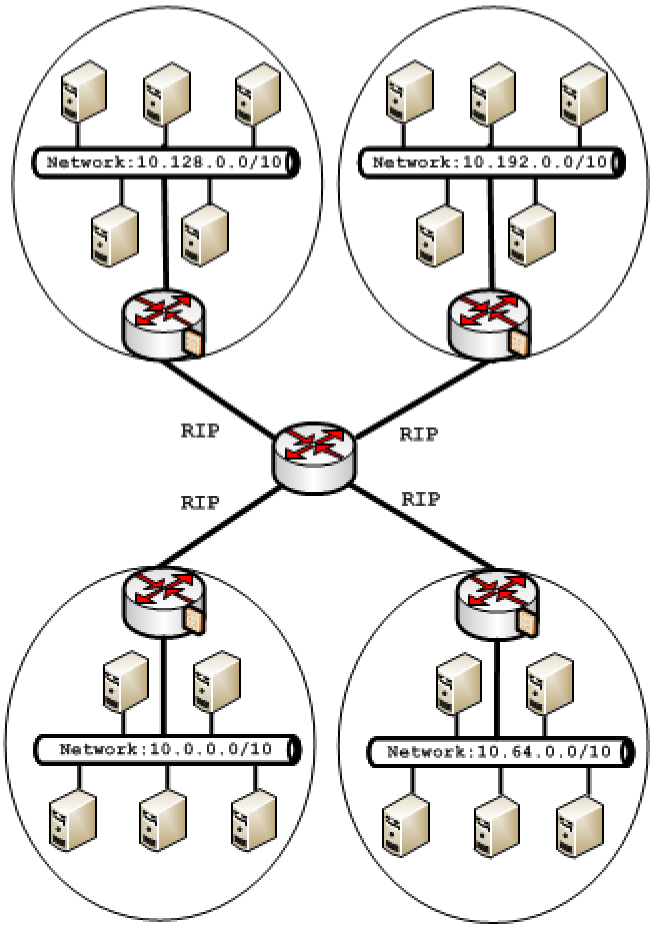

Fig. 2. Virtual Experimental setup for Slammer- like Pseudo Worm Behavior Analysis

4.3 Experimental Methodology

Moore et al. [2] reported a set of key characteristics of the Slammer worm outbreak in 2003 and these were used to set up the experimental parameters. Moore et al. reported that 18 hosts per million of the entire IPv4 addresses space were susceptible to infection. They also observed that the Slammer worm exhibited an average scan rate of 4,000 datagrams per infected host per second.

A single class A network has 2^{24} hosts, and so will contain 2^{24} * 0.000018 = 302 susceptible hosts. On this basis, 302 virtual machines with the Slammer like pseudo-worm daemon were deployed across the four subnets. Each worm daemon was configured to scan within a single class A network (10.0.0.0/8). In order to avoid overloading the server farm hardware (in which case we would have been measuring the effect of the hardware restrictions, rather than the properties of the worm) we scaled back the average worm scanning rate by a factor of 80. Therefore, based on an average scan rate reported by Moore et al of 4000 scans per second, we configured the Slammer-like network daemons to scan at 50 scans per seconds in our experiment.

4.4 Experimental Results

Figure 3 shows the results of the experiment, with the time axis scaled down by a factor of 80, to make the results comparable with the real infection event of 2003, reported by Moore et al [2].

In order to provide a baseline comparison, we have also plotted a susceptible/infected analytical model, based on the Logistic Equation, reported by Ediger [9].

The VMT experiment achieved infection of 99% of vulnerable hosts within approximately 14 minutes. This time is directly comparable with that reported in [2] for the real Slammer event of 2003. We have also plotted available data from [2] for the 2003 event, in Figure 3 (empirical data is only available for the first 4 minutes of infection), and it can be seen that the VMT experimental results are again, broadly comparable.

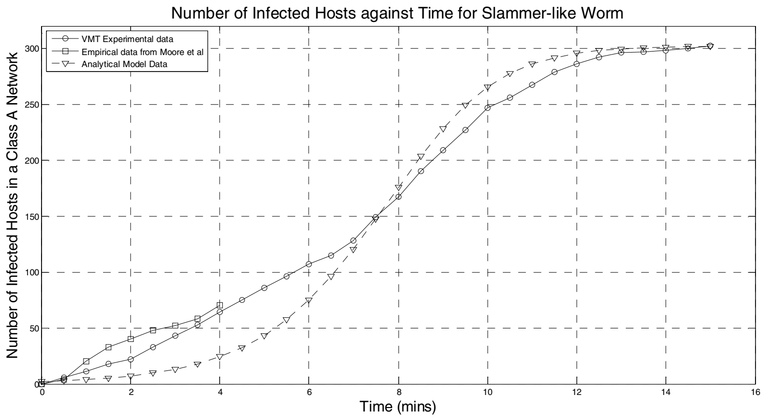

Fig. 3. Experimental Results for Slammer-Like Worm Infection on VMT

5 Discussion

The cyber-epidemiological analysis of zero-day internet worms remains a significant challenge and use of virtualized testbeds remains a viable tool for such research. This paper has presented a novel Virtualized Malware Testbed (VMT) for worm testing based on VMware ESXi and open source software. We have also demonstrated its feasibility for epidemiological experimentation for a Slammer–like pseudo-worm.

In comparison with other network and security testing environments, VMT provides an effective, scalable, remotely manageable and isolated environment, which also incorporates efficient log creation. It is expected that VMT will be a useful experimentation environment for epidemiological investigations of existing and hypothetical zero-day worms, as well as the investigation and evaluation of candidate countermeasures.

5.1 Limitations and Future Work

This paper has reported the design, implementation and initial testing of VMT with a single network worm type. The experimentation has also been limited to the scale of a single class A network (circa 16M hosts).

In terms of future work, we shall be exploring the use of VMT to explore the stochastic properties of worms, as well as its ability to investigate other types of network worm. We also expect to experiment with a range of candidate worm countermeasures, and to explore the applicability of VMT for charactering the epidemiology of more sophisticated malware threats, such as Stuxnet.

Acknowledgment. VMware ESXi, VMware VCenter servers are provided as part of VMware Academic Program. (http://www.vmware.com/partners/academic/program-overview.html).

References

1. Weaver, N., Paxson, V., Staniford, S., Cunningham, R.: A taxonomy of computer worms. In: Proceedings of 2003 ACM Workshop on Rapid Malcode, pp. 11–18. ACM Press, New York (2003)
2. Moore, D., Paxson, V., Savage, S., Shannon, C., Staniford, S., Weaver, N.: Inside the Slammer worm. IEEE Security and Privacy 1(4), 33–39 (2003)
3. Langner, R.: Stuxnet: Dissecting a cyberwarfare weapon. IEEE Security & Privacy 9(3), 49–51 (2011)
4. White, B., Lepreau, J., Stoller, L., Ricci, R., Guruprasad, S.: An integrated experimental environment for distributed systems and networks. In: Proceedings of 5th Symposium on Operating Systems Design and Implementation, Boston, MA, USA, pp. 265–270. USENIX (2002)
5. Benzel, T., Braden, R., Kim, D., Neuman, C.: Design, deployment and use of the DETER testbed. In: Proceedings of DETER Community Workshop on Cyber Security Experimentation and Test 2007, Berkeley, CA, USA, pp. 1–8. USENIX (2007)
6. Lippmann, R.P., Fried, D.J., Graf, I., Haines, J.W., Kendall, K.R., McClung, D., Weber, D., Webster, S.E., Wyschogrod, D., Cunningham, R.K., Zissman, M.A.: Evaluating intrusion detection systems: the 1998 DARPA off-line intrusion detection evaluation. In: Proceedings of the 2000 DARPA Information Survivability Conference and Exposition (DISCEX), vol. 2, pp. 12–26. IEEE Press, New York (2000)
7. Rossey, L.M., Cunningham, R.K., Fried, D.J., Rabek, J.C., Lippmann, R.P.: LARIAT: Lincoln Adaptable Real Time Information Assurance Testbed. In: Proceedings of IEEE Aerospace Conference, Big Sky, Montana, USA, vol. 6, pp. 2671–2682. IEEE (2002)

8. Perumalla, K.S., Sundaragopalan, S.: High fidelity modeling of computer network worms. In: Proceedings of the 20th Annual Computer Security Applications Conference (ACSAC), Tucson, AZ, USA, pp. 126–135. ACSA (2004)
9. Ediger, B.: Simulating Network Worms, `http://www.stratigery.com/nws/`
10. Tidy, L., Woodhead, S.R., Wetherall, J.C.: A Large-scale Zero-day Worm Simulator for Cyber-Epidemiological Analysis. UACEE International Journal of Advances in Computer Networks and Security 3(2), 69–73 (2013)
11. ns (network simulator), `http://www.isi.edu/nsnam/ns`
12. Vahdat, A., Yocum, K., Walsh, K., Mahadevan, P.: Scalability and accuracy in a large-scale network emulator. In: Proceedings of USENIX 5th Symposium on Operating Systems Design and Implementation (OSDI), Boston, MA, USA, pp. 271–284. USENIX (2002)
13. Peterson, L., Anderson, T., Culler, D., Roscoe, T.: A blue print for introducing disruptive technology into the internet. SIGCOMM Computer Communication Review 33(1), 59–64 (2003)
14. Provos, N.: A virtual Honeypot framework. In: Proceeding of USENIX 13th Security Symposium, San Diego, USA, pp. 1–14. USENIX (2004)
15. Dunlap, G., King, S., Cinar, S., Basrai, M., Chen, P.: ReVirt: enabling intrusion analysis through virtual machine logging and replay. In: Proceeding of USENIX 5th Symposium on Operating Systems Design and Implementation (OSDI), Boston, MA, pp. 208–223. USENIX (2002)
16. Jiang, X., Wang, X.: Stealthy malware detection through VMM-Based "out-of-the-box" semantic view reconstruction. In: Proceedings of 14th ACM Conference on Computer and Communication Society (CCS), Alexandria, VA, USA, pp. 128–138. ACM (2007)
17. Jenson, J.: A novel testbed for detection of malicious software functionality. In: Proceeding of Third International Conference on Availability, Security and Reliability, Barcelona, Spain, pp. 292–301. IEEE (2008)
18. Jiang, X., Xu, D., Wang, H.J., Spafford, E.H.: Virtual Playgrounds for Worm Behavior Investigation. In: Valdes, A., Zamboni, D. (eds.) RAID 2005. LNCS, vol. 3858, pp. 1–21. Springer, Heidelberg (2006)
19. Årnes, A., Haas, P., Vigna, G., Kemmerer, R.A.: Digital Forensic Reconstruction and the Virtual Security Testbed ViSe. In: Büschkes, R., Laskov, P. (eds.) DIMVA 2006. LNCS, vol. 4064, pp. 144–163. Springer, Heidelberg (2006)
20. Sun, W., Katta, V., Krishna, K., Sekar, R.: V-netlab: an approach for realizing logically isolated networks for security experiments. In: CSET 2008: Proceedings of the Conference on Cyber Security Experimentation and Test, Berkeley, CA, USA, pp. 1–6. USENIX (2008)
21. Fagen, W., Cangussu, J., Dantu, R.: A virtual environment for network testing. Journal of Network and Computer Applications Archive 32(1), 184–214 (2009)
22. Nessus Vulnerability Scanner, `http://www.tenable.com/products/nessus`
23. Wireshark, `http://www.wireshark.org/`
24. Snort, `http://www.snort.org/`
25. Windows Sysinternals, `http://technet.microsoft.com/en-US/sysinternals`
26. The Bro Network Security Monitor, `http://www.bro.org/`
27. Jiang, X., Xu, D., Eigenmann, R.: Protection mechanisms for application service hosting platforms. In: Proceedings of 4th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2004), Chicago, Illinois, USA, pp. 633–639. IEEE Computer Society (2004)

28. VMware ESXi, `http://www.vmware.com/products/vsphere/esxi-and-esx/overview.html`
29. Damn Small Linux (DSL), `http://www.damnsmalllinux.org`
30. Quagga Software Routing Suite, `http://www.nongnu.org/quagga`
31. VMware vCenter Server, `http://www.vmware.com/products/vcenter-server/overview.html`
32. VMware vSphere PowerCLI, `http://communities.vmware.com/community/vmtn/server/vsphere/automationtools/powercli?view=overview`