Detecting Computer Worms in the Cloud

Sebastian Biedermann and Stefan Katzenbeisser

Security Engineering Group
Department of Computer Science
Technische Universität Darmstadt
{biedermann,katzenbeisser}@seceng.informatik.tu-darmstadt.de

Abstract Computer worms are very active and new sophisticated versions continuously appear. Signature-based detection methods work with a low false-positive rate, but previously knowledge about the threat is needed. Anomaly-based intrusion detection methods are able to detect new and unknown threats, but meaningful information for correct results is necessary. We propose an anomaly-based intrusion detection mechanism for the cloud which directly profits from the virtualization technologies in general. Our proposed anomaly detection system is isolated from spreading computer worm infections and it is able to detect unknown and new appearing computer worms. Using our approach, a spreading computer worm can be detected on the spreading behavior itself without accessing or directly influencing running virtual machines of the cloud.

Keywords: Computer Worms, Anomaly Detection, Cloud Computing

1 Introduction[1]

Cloud computing offers the utilization of IT resources such as computing power and storage as a service through a network on demand. It can save a company the purchase of own data centers or the employment of own IT specialists. This is made possible through virtualization technologies.

The cloud consists of a network of hardware nodes, where each node is able to run several virtualized operating systems in parallel using a virtual machine monitor called hypervisor. A centralized cloud manager summarizes and monitors the resources of all connected hardware nodes and determines which node offers enough free resources to start a new virtual machine if needed. These virtual machines contain an operating system as well as additional software components or requested data.

The virtual machines of the cloud can be easily accessed from outside and their available performance can be flexibly scaled. This architecture results in

[1] The work presented in this paper was performed in the context of the Software-Cluster project EMERGENT (www.software-cluster.org). It was funded by the German Federal Ministry of Education and Research (BMBF) under grant no. "01IC10S01". The authors assume responsibility for the content.

an internal network running many operating systems which are usually able to connect to each other.

Various different definitions of cloud computing are available. Most of the time, cloud computing is considered as a compilation of abstract remote services. Figure 1 shows the cloud from a technical point of view which we use in our work. The cloud network is separated in a back-end and a front-end.

The cloud management software is installed on the front-end. It monitors the resources of the back-end and is also connected to an external network, for example the Internet. Users can make a request on the front-end, usually with a web interface, and define which kind of virtual machine they want to start in the back-end in combination with reserved performance. The requested virtual machine can contain a previously chosen operating system and additional requested software. The core component of the cloud is the back-end network.

It consists of several hardware nodes where each has installed hypervisor software. With the help of the hypervisor software, many virtual machines can be launched in parallel on a single hardware node. Finally, the management component links a connection from the outside user to the started virtual machine in the back-end.

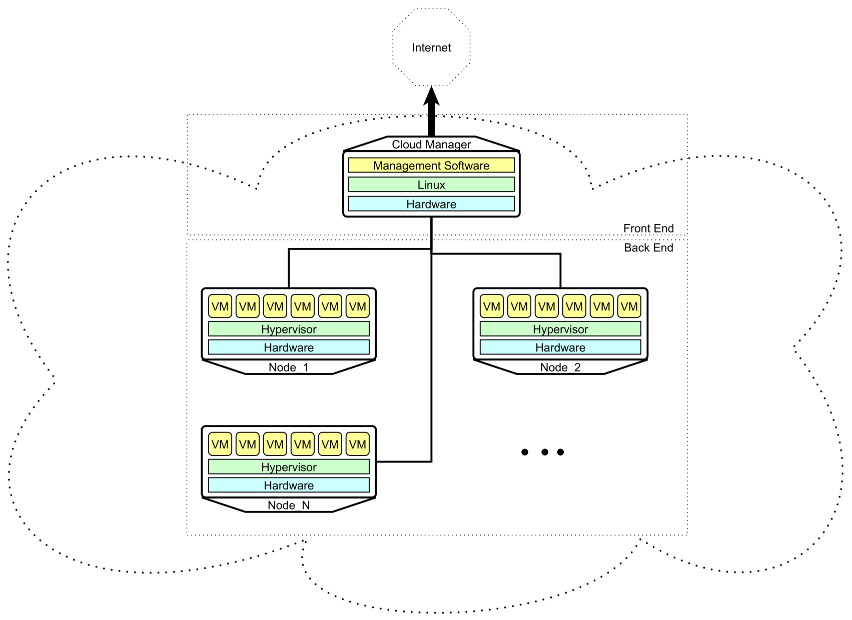

Fig. 1. Front-end and back-end of a cloud computing setup.

Cloud computing networks that offer “Platform as a Service” (“PaaS”) in homogeneous networks with Microsoft Windows operating systems are increasingly used in companies to save hardware resources. Especially in these private clouds, the network is often a homogeneous operating system environment. Employees have full control over these virtual machines which are connected to the internal company network through high-speed links and which can be remotely accessed and used like a normal workplace. The cloud manager can quickly balance and optimize the current needed resources.

To reach this goal, running virtual machines can be live-migrated from one hardware node to another hardware node in the back-end without notifying the logged-in remote cloud user.

Cloud computing offers many new opportunities like cost savings and high flexibility, but this new kind of network is still susceptible to the old open security problems which affect computer networks today.

In this paper, we address the problem of fast spreading malicious software (computer worms) in the cloud back-end network. These computer worms can have a devastating impact on the homogeneous and flexible cloud installation and can cause massive financial losses. The problem of computer worms exists since a long time and it does not seem as if the number of appearing computer worms would decrease. Further, the latest computer worms are becoming more sophisticated and complex and thus detection and containments get more challenging.

Traditional networks offer no means to inspect captures for malicious incidents, unless special intrusion detection systems (IDS) or special monitoring components are installed on the operating systems. On the contrary to this, the virtualized network of the cloud offers new opportunities to monitor internal events of virtual machines without the need of direct access or influence.

In this paper, we discuss the challenge of detecting spreading computer worms as fast as possible in a cloud back-end network. Even though the problem as such is old, we argue in this paper that it can be better addressed in virtualized networks like the cloud back-end.

To this end, we propose a new kind of computer worm detection system especially for the cloud back-end network. Our system is anomaly-based, very flexible and benefits directly from the virtualization technologies. The proposed detection system uses a centralized perspective on the entire back-end of the cloud network and interprets this network as a whole.

A particular advantage is that our proposed computer worm detection system can discover unknown computer worms and that it can not be manipulated by the computer worms because it runs isolated from the virtual machines on the hypervisor layer.

To be able to carry out various test-runs, we implemented a simulation framework which uses our IDS approach and which can run with a very large number of parallel simulated virtual machines. With the help of these simulations, a real spreading epidemic of a computer worm infection in a very large virtual back-end network can be investigated.

2 Open Problem and Approach

In the year 2003, the “Blaster” computer worm infected millions of Windows 2000, XP and Windows Server 2003 systems by remotely exploiting a RPC bug [1]. All infected systems were running a process called “msblast.exe” which started together with the operating system at boot time. The spreading behaviour of the “Blaster” computer worm became an annoying and tremendous costly epidemic.

In early November 2008, the “Conficker” computer worm infected up to 15 million Windows systems by exploiting a remote NetBIOS bug. It injects a randomly named dynamic link library (DLL) into the authentic “svchost.exe” Windows process and tries to hide its presence this way [2]. Version A of the “Conficker” computer worm was especially focused on the infection of other systems that were connected within an internal network which caused very fast spreading within organizations. Conficker B included a brute force attack mechanism to retrieve local passwords and version C even included a P2P protocol for its own distribution [3].

In the year 2010, the “Stuxnet” computer worm heralds the new era of spreading malicious software. This computer worm aims to manipulate industrial control systems, but it is also distributed on Windows systems from version 2000 to 7 using various exploiting technics [4]. The occurrence of the “Stuxnet” computer worm has once again shown that the dangers of spreading malicious software are far away from over.

These computer worms constitute only some examples of computer worms which caused huge economic damage in the last years.

The threat of spreading computer worms is not over: the past showed that continuously new worms arise, despite the deployment of new techniques for detection and abatement. Virus detection software or intrusion detection software can be installed on systems, but this software can also be manipulated by malicious processes or even deactivated from the users themselves by mistake. Until now, containment of fast spreading worms is an open problem to which no satisfying countermeasure is known.

Unlike traditional network subnets, on which one operating system was installed on each hardware node connected directly to the network, todays networking components are virtualized, i.e. multiple systems run in parallel on one hardware node, managed by a hypervisor software component from which the systems are completely isolated. In such virtual computing environments, worms can still spread through traditional methods. However, by utilizing the existing virtualization infrastructure, more useful information of each single running virtual machine can be obtained from the outside in a passive manner:

Virtual machine introspection (VMI) allows to get information on running virtual machines through the hypervisor layer without the need to directly access the machines. This information can contain a list of the current running processes of the operating system, current loaded modules or even an image of the whole random-access memory (RAM). There are flexible libraries available that provide virtual machine introspection without requiring changes to the hypervisor [5].

Based on this technology, intrusion detection systems (IDS) can be developed which monitor virtualized systems from the outside. This architecture has the benefit that this kind of intrusion detection systems can not be manipulated or even detected by malicious software running on the infected virtual machine, because the IDS code is out of reach and thus in a completely separated software environment [6].

The virtual infrastructure of the cloud network offers an elegant way to identify an infectious spread without auxiliary worm signatures. In a virtualized network, information can be obtained from each running virtual machine through VMI. A single centralized monitoring software component can receive all this information of all running virtual machines and interpret the current status of the network.

This way, an IDS can be built based on anomalies detected in the whole virtualized cloud computing network. To demonstrate the feasibility of this approach, we built a centralized anomaly detector which collects information using VMI of all running virtual machines in the cloud back-end. Our approach offers very fast detection of malicious spreading behaviour because of the centralized abstract view on the cloud network. Using our approach, spreading malicious processes can be detected based on their spreading behaviour in the back-end network itself, even without having previous knowledge about the threat like signatures.

After the detection of a computer worm, a signature can be generated and further used in an network traffic based IDS like "Snort" [7] to ban the threat at the gates of the network.

3 Technical Approach

To realize the proposed idea, we first have to define what we consider an anomaly in the cloud computing network. By our definition, an anomaly is a collection of more than one appearing single inconsistencies. In particular, we identify two different inconsistencies that can appear in an operating system in our virtual cloud computing network. An inconsistency arises if one or more of the following events are detected on a running virtual machine in the cloud using VMI or hypervisor information in general:

- A new process is started which is not in a list of known or usual processes.
- A new module is loaded which is not in a list of known or usual modules.

In this context, the execution of a "known" process or module is not uncommon and well known. In contrast, an "unknown" process or module can be a process or module which is not very popular or which execution is very unusual. Of course, one can define more complex event that cause inconsistencies, which may include, for example, unusual high outgoing traffic of a virtual machine or continuous high CPU performance. There are many ways to define triggers for inconsistencies in the cloud computing network. In this paper, we limit our inconsistency to the two listed events above.

Single inconsistencies are not conclusive; however, if a continuous distribution or increasing of exactly the same inconsistency in the cloud back-end network can be discovered, one can infer an anomaly. We define that an anomaly occurs when in successive scans of running virtual machines using VMI at predefined time intervals ΔT a continuous steady increase of an inconsistency in the cloud back-end network exceeds a predefined limit L.

These observations allow to detect computer worms indirectly through their spreading behaviour. These computer worms are unknown processes or modules even themselves or they use other unusual processes or modules during their malicious work on the operating system.

For example, the following steps identify an ongoing threat with the help of observed running processes on virtual machines in the cloud back-end network:

1. Retrieve a list of running processes of a randomly chosen virtual machine using VMI in the cloud back-end network.
2. Find processes which are not in a list of known or common processes and add these information temporarily to a list of unknown processes.
3. After a larger number of scans, identify a potential spreading process, characterized by the fact that this inconsistency occurs on an increasing number of virtual machines.
4. If the occurrence of this identified process exceeds the value of a predefined limit L, take corrective action (e.g. isolate infected virtual machines from the back-end network for further investigations).
5. In contrast, if the occurrence of this process decreases and reaches not the value of the predefined limit L, add its information to the list of known processes and continue. In this case, it is assumed that the process is not a computer worm, but an event occurring in parallel such as e.g. a simultaneously launched update on multiple virtual machines.

An algorithm following these steps is not only able to identify a spreading process in the cloud back-end network, it also improves itself by learning information about unknown harmless processes. This can be helpful to distinguish between the spread of malicious software and regular updates, which can have characteristics of a computer worm if they are installed on virtual machines simultaneously from the Internet.

Using these proposed steps, a computer worm can be identified only by its spreading behavior itself. In this way, no prior knowledge about the computer worm is necessary, such as a signature. This approach is benefiting from the virtualization technologies of the cloud in general by passively observing the running virtual machines with the help of VMI and it is also benefiting from an abstract centralized view on all network nodes of the entire cloud back-end network. New and unknown computer worms can be detected and the triggered anomaly may lead to further more active countermeasures.

4 Experimental Cloud Configuration

Our experimental implementation uses a simple setup. We focused only on monitoring running processes, accordingly we received a current process-list of running virtual machines on the back-end nodes. The implementation consists of a centralized cloud network management component including a "Spreading Process Monitor" component running on Linux. The cloud manager consists of scripts which are able to transfer virtual machine images to connected nodes, launch, stop and destroy them.

Each connected node uses the Xen hypervisor [8] and accordingly each connected node includes an administrative virtual machine which is called in Xen "Domain 0" ("dom0"). This administrative virtual machine can provide information about the CPU usage or the network traffic of the running virtual guest machines ("domU") on the same hardware node. For the implementation, the "dom0" virtual machine additionally uses the XenAccess[2] VMI library package to retrieve a list of the current running processes on each virtual guest machine on this hardware node.

This is done with the help of direct memory access technics and previously defined knowledge about the structure of the RAM dependent on the chosen operating system. Figure 2 illustrates the procedure of virtual machine introspection on a single hardware node in the cloud back-end network.

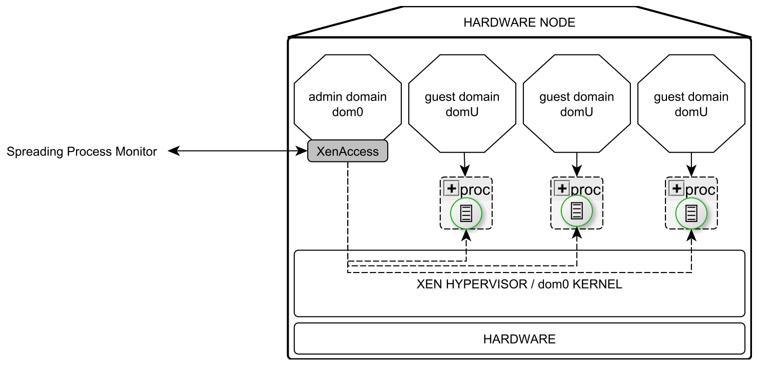

Fig. 2. Illustration of virtual machine introspection (VMI) on a single hardware node in the cloud back-end network.

[2] http://code.google.com/p/xenaccess/

During the runtime, the "Spreading Process Monitor" collects the process-lists of randomly chosen virtual guest machines on different hardware nodes in the network. Continuously collecting and comparing these lists offers the opportunity to detect spreading inconsistencies which increase their appearance on other virtual guest machines in the cloud back-end network. This method is illustrated in Figure 3.

As countermeasures, isolating or freezing infected virtual machines are possible and this can be also controlled by the centralized cloud manager. The network traffic of all virtual guest machines is routed through a bridge which is configured in the administrative "dom0" virtual machine. The current network traffic of each guest machine can be easily scanned, analyzed and also blocked using host-based network filtering software in further steps after an anomaly.

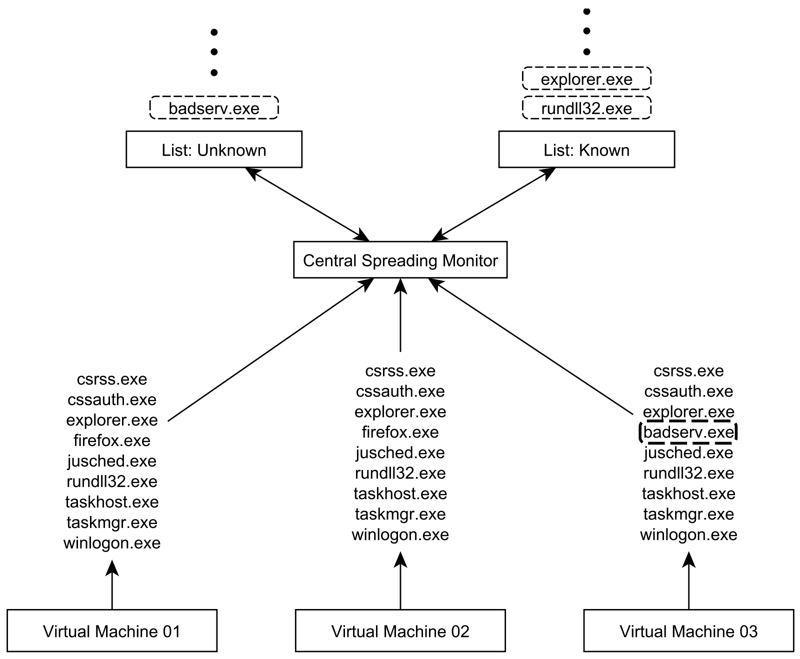

Fig. 3. Retrieving the process lists of virtual guest machines of the back-end network and subdivide this collected information in lists of known and unknown processes.

This way, traffic of infected virtual machines can be isolated or filtered, so that infected virtual machines can be prevented from infecting other uninfected virtual machines inside the internal cloud back-end network. This is a simple but effective approach to get the ongoing spreading threat under control and to get more time for detailed investigations.

5 Simulation

To get a better view on the performance of the approach and practicality on the proposed technique, we developed a simulation framework. Our framework simulates many parallel running virtual machines, each infected one can also infect each other virtual machine with a malicious process. Once infected, the malicious process randomly scans for a new target virtual machine inside the simulated cloud back-end and infects this new victim. The simulation includes a centralized spreading process monitor to identify spreading behaviour on the simulated virtual machines.

Random virtual machines are continuously scanned by this centralized process spreading detector. In the real environment, this scan is done using virtual machine introspection. In such a simulation, time factors play a centralized role. Figure 4 illustrates the amount of infected virtual machines for a simulated back-end network with 256, 512 and 1024 single virtual machines until detection of an anomaly in percent. The time interval for the scans of the centralized process spreading detector is set to $\Delta T = 500ms$ and the limit for the amount of inconsistencies until an anomaly L is set to 6. This simulation shows that the spreading time of the inconsistencies (the unknown process) is a very important factor using our approach. A fast detection can prevent major damage.

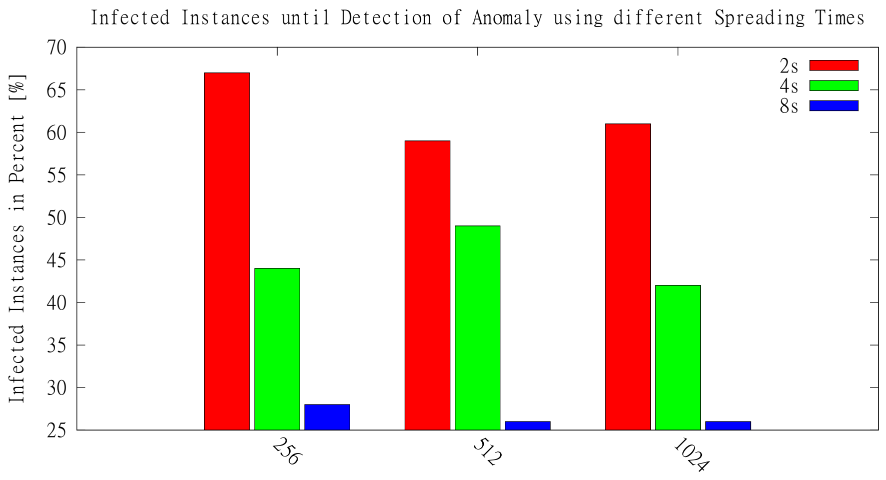

Fig. 4. Percentage amount of infected virtual machines until detection of the anomaly using different spreading times. Three test-runs with a cloud consisting of 256, 512, and 1024 single simulated machines. Average of eight runs.

A malicious process with a spreading time of two seconds easily infects more than 50% of the whole network until it is detected using the chosen parameters. However, the amount of infected virtual machines until detection decreases rapidly if the spreading time of the process is in the order of four or even eight seconds. Thus, feasible and matching selection of parameters ΔT and L is very important.

To defend the cloud back-end network from an entire infection causing a total black-out of the whole network, the spreading process can be disarmed by blocking identified ports which the malicious process uses for spreading, even the infected virtual machines can be completely isolated from the communication in the network.

Figure 5 shows the influence of a countermeasure approach which starts to isolate and remove the malicious process and make eight virtual machines per second immune from the infection after an ongoing anomaly has been detected. Here, we used a scanning interval $\Delta T = 125ms$. The Limit L is still 6 and the cloud back-end network consists of 512 running virtual machines.

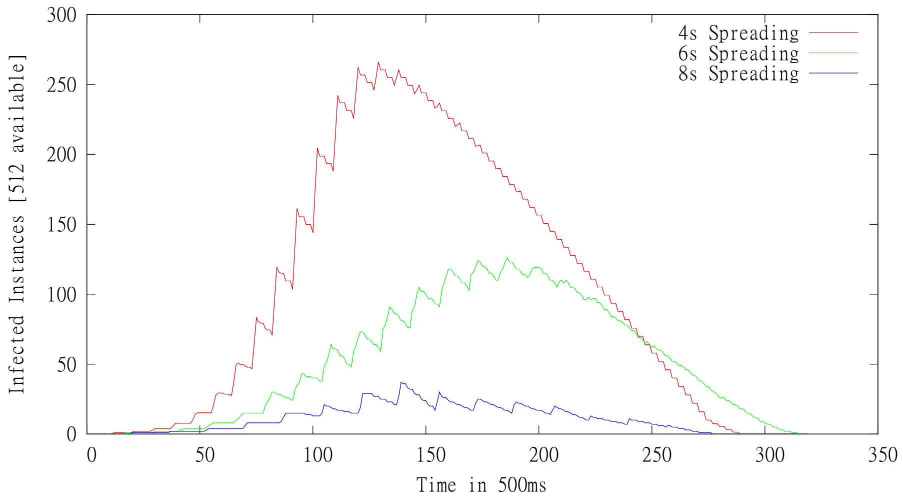

Fig. 5. Simulation of the detection of an anomaly and an immediate countermeasure. The spread is contained by the countermeasure. Different runs with spreading times of 4s, 6s and 8s and with 512 virtual machines. Average of eight runs.

It can be seen that after six seconds around 50% of the virtual machines are infected if the spreading time is four seconds. After the peak, the countermeasure works constantly because all other uninfected virtual machines have already been infected and are immune, which means the computer worm is defeated. The countermeasure works much better if the spreading time is six or eight seconds.

Here, this countermeasure is effective and keeps continuously the most virtual machines of the back-end network under control. The points in time a spreading process can be detected and a counteraction can be started are very important. Using these simple and very fast tactics, a black-out of the whole simulated cloud back-end caused by a total infection can be avoided.

6 Future Work

Our proposed solution benefits directly from the virtualization technologies in the cloud network. Our proposal observes passively the running virtual machines in the back-end and it is not vulnerable to attacks of the spreading computer worms, because our used software runs isolated in "dom0" administrative virtual machines which do not offer services to the network and which usually block unintended communication in general.

Though, there are some ways to improve this approach, with some of them we deal in actual work. Of course, it is not efficient enough to observe only the names of processes or loaded modules, recent computer worms hide mostly in other processes or change their names continuously. In actual work, we generate hashes of each process and module contained in the RAM image of the virtual guest machine and compare these collected integrity measurements.

At first glance, the monitoring of processes and modules do not appear sufficient to detect the new generations of sophisticated computer worms. But it can also be said that with this approach, for example the spreading computer worm "Stuxnet" should be detected as an anomaly. Because not the computer worm itself has to be detected, but the influence and the impact of the worm infection on the operating system in the virtual machine can be discovered. "Stuxnet" continuously loads correctly signed driver modules on each infected machine, because the creators have even stolen the signing keys from a hardware manufacturer. These modules would be added to the unknown list and the continuous spreading of them should be identified as an anomaly.

We propose a security system which uses anomaly detection and which should of course lead to further detailed investigations.

At last, the scanning and identifying progress can be greatly accelerated in future work with the help of parallelized scans and concurrent threads.

7 Conclusion

Cloud computing is changing the IT world and introduces enormous and great improvements. Still, cloud installations are vulnerable to classic open problems such as fast-spreading computer worms. Traditional detection methods, usually based on a signatures, are not able to bring this problem under control, because the amount of new occurring computer worms is steadily growing.

Anomaly detection approaches are more robust than signature based detection methods, but they need meaningful information from the network. The cloud offers new opportunities to monitor the network without directly influencing or accessing single virtual machines using virtual machine introspection.

Therefore, this can be used to have an abstract view on the entire system and to interpret the state of the network with the help of a centralized detector to identify malicious spreading inconsistencies.

In this paper, we showed that it is possible to use features offered by virtual machine introspection to detect and to contain the spreading of computer worms. Our detection method is based on anomalies and works by observing the spreading behaviour of suspicious inconsistencies in the virtualized cloud back-end network. We optimized this detection method using a set of different simulations and we analyzed the influence of different parameters and a countermeasure.

References

1. Microsoft, "Buffer overrun in rpc interface could allow code execution (823980),"
2. T. W. Felix Leder, "Know your enemy: Containing conficker,"
3. C. W. Group, "Lessons learned june 2010," 2011.
4. L. O. M. Nicolas Falliere and E. Chien, "W32.stuxnet dossier," in *Symantec Security Response.*
5. B. D. Payne and W. Lee, "Secure and flexible monitoring of virtual machines," in *Annual Computer Security Applications Conference*, pp. 385–397, 2007.
6. T. Garfinkel and M. Rosenblum, "A virtual machine introspection based architecture for intrusion detection," in *Network and Distributed System Security Symposium*, 2003.
7. M. Roesch, "Snort: Lightweight intrusion detection for networks," in *USENIX Systems Administration Conference*, pp. 229–238, 1999.
8. P. Barham, B. Dragovic, K. Fraser, S. Hand, T. L. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," in *Symposium on Operating Systems Principles*, pp. 164–177, 2003.