CReW: Cloud Resilience for Windows Guests through Monitored Virtualization

Flavio Lombardi
Sistemi Informativi - DCSPI
Consiglio Nazionale delle Ricerche
Rome, Italy
Email: flavio.lombardi@cnr.it

Roberto Di Pietro
Dipartimento di Matematica
Universita Roma Tre
Rome, Italy
Email: dipietro@mat.uniroma3.it

Claudio Soriente
DLSIIS
Universidad Politecnica de Madrid
Madrid, Spain
Email: csoriente@fi.upm.es

***Abstract*—Clouds are complex systems subject to an increasing number of anomalies and threats. In this paper we briefly revisit the issues related to Windows guest cloud service resilience and later provide some preliminary results on the resilience of Windows cloud guests via virtualization. In particular, we propose an architecture, Cloud Resilience for Windows (CReW). CReW can transparently monitor guest Windows VMs and can also react to both security breaches and system integrity violation, improving the dependability of cloudified Windows systems. CReW can also improve resilience from software misconfiguration by restoring the guest latest safe state. Effectiveness and performance of a CReW prototype have been evaluated; obtained results show the feasibility of such a system.**

***Keywords*-Resilience management and security issues in clouds; Resilience measurement studies; Operating systems, file and storage systems; Secure and intrusion tolerant systems**

I. Introduction

Resilience [6] is the ability of a system to recover and provide an acceptable level of service in situations in which a component (HW/SW) is either missing (e.g. a crashed service or a burned component) or it is not behaving correctly, where this latter case can be originated by configuration changes, upgrades or attacks, to name a few.

In particular, the Windows kernel and services are constantly exposed to a number of new threats; new vulnerabilities are discovered that malware and especially rootkits can exploit to gain administrator privileges and hide their presence from spyware blockers, antiviruses, and system utilities. In fact, rootkits can intercept calls to the Windows libraries and modify the returned values in order to remain hidden [14]. Kernelspace rootkits can also intercept calls to execution ring 0 where they can alter kernel data structures. In addition, software upgrades and deployment of new components can reduce guest service functionality or open the way to new vulnerabilities and threats.

Windows-based services are often deployed on Linux-based Clouds as Virtual Machines (VMs) [1], leveraging the reliability of the underlying OS and the isolation benefits virtualized platforms can provide. Linux offers a securer hosting platform, offering at least two different hypervisor technologies, Xen and KVM [13]. The latter has been preferred to Xen in this work due to its tight integration with the vanilla Linux kernel and due to the better support from the majority of Linux vendors and developers who are slowly phasing-out Xen.

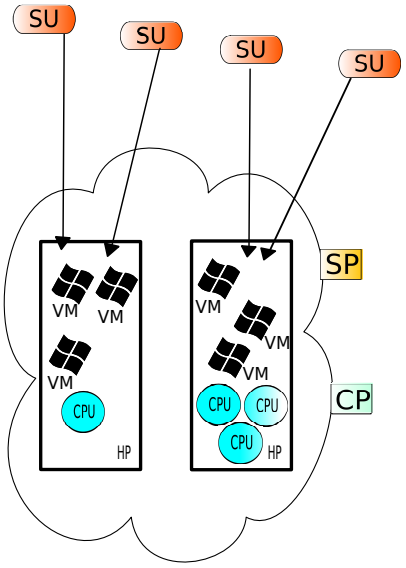

Figure 1. Windows Server VMs in the Cloud: Service Users (SU) access Windows server VMs managed by Service Providers (SP) on Hosting Platforms (HP) managed by Cloud Providers (CP)

This paper discusses the Windows guest cloud service resilience problems (see Figure 1) and provides some contributions. First, it describes an architecture, Cloud Resilience for Windows (CReW) for increased security of cloud resources and services. CReW can effectively improve the trustworthiness in such computing systems. CReW can transparently monitor guest Windows VMs and can also react to security breaches and to unauthorized system configuration changes, improving the persistence of the dependability of cloudified Windows systems. A CReW prototype has been implemented on a widely deployed cloud host software (Eucalyptus [10]). Effectiveness and performance of a CReW prototype have been evaluated; obtained results show the feasibility and effectiveness of such a system.

The remainder of this document is organized as follows: Section II summarizes state of the art solutions applied to

cloud resilience; Section III describes CReW architecture and gives some implementation details; Section IV presents preliminary performance figures; finally, in Section V conclusions are drawn.

II. Related Work

Cloud resilience and particularly security issues have been the subject of much recent research efforts, since undetected attacks can be extremely dangerous over time.

In *Self-Cleansing Intrusion Tolerance* (SCIT) [4] all servers are considered potentially compromised. SCIT periodically restores servers from secure snapshots in order to make nodes resilient against long-lasting attacks. Similarly, *VM-FIT* [3] creates redundant server copies which can periodically be refreshed to increase the resilience of the server. Such systems do not support long-lasting sessions required by most complex server applications. Sousa's approach [19] leverages proactive recovery: In the case when the probability that services have been compromised exceeds a given threshold, clean service replicas react and substitute compromised ones. Similarly to the above-mentioned solutions, the Sitar [20] system relies on redundancy to obtain intrusion-tolerance. Sitar requires an IDS to successfully detect an attack for the intrusion tolerance mechanisms to operate. It remains to be seen what happens to sitar when the intrusion attempt is successful and goes undetected by the IDS. Also LoGrid [16] is an interesting example of resilience tool that adopts autonomic reaction mechanisms to protect from attacks.

Co-location attacks have been proven to be able to extract information from a target VM on the same hardware [15]. Placing a VM on the same host as the target VM can be easily obtained by an attacker, provided information on the cloud auto-scaling allocation algorithm is known.

Most recent approaches leverage CPU hardware virtualization support [11] to monitor Virtual Machines. *SecVisor* [17] and *KVM-L4* [12] leverage virtualization to monitor guest kernel code integrity from a privileged VM or hypervisor. Such proposals have limitations that prevent them from being used in distributed computing scenarios (e.g.. SecVisor only supports one guest per each host) or do not scale up to large distributed systems such as clouds. KVM-L4 shares the same underlying technology as [7] but the additional context switching overhead in the 64-bit scenario (as in most cloud hosts) is not clear.

Among other solutions, SVGrid [21] creates a secure Xen-based virtual environment aimed at monitoring file and network access requests from a virtual machine where security policies are enforced. SVGrid provides isolation among applications by running multiple Grid Virtual Machines each running one grid application at a time. Also Smith [18] proposes a separation of grid components currently running on one machine into separate virtual operating systems. This prevents users from installing malicious software and driving inter-users attacks, since each user is confined to his or her own virtual environment.

Virtualization also introduces technological challenges that deserve special attention. They include an increase in the complexity of digital forensics and questions regarding additional vulnerabilities of the host system.

III. CReW Architecture and Implementation

The proposed Cloud Resilience for Windows (CReW) is intended to actively monitor guest Windows VM and cloud middleware activity and integrity. Our proposal extends the KvmSec [7] and ACPS [8] approaches in order to protect guest Windows components from misconfiguration-enabled damage and against intruders and attacks such as worms and viruses. CReW is entirely contained in the host side and leverages semantic introspection [5]. This allows deploying and running unmodified guest VM images.

As shown in Figure 2, in CReW, the host-side database *Checks DB* contains computed checksums for selected host infrastructure and guest kernel code, data, and files. In fact, the storage is virtualized and accesses are intercepted by the *Interceptor/Actuator* component. Further, off-line integrity cross-checks are periodically run over guest system configuration files, in order to detect differences between guest-returned file content and actual files on disk (cross-view detection [14]). The runtime *HashC* daemon repeatedly recomputes hash values of monitored objects and submits warnings to the *AlertDB*. The *Evaluator* daemon asynchronously examines such warnings and decides whether anomalies in the system behavior are present or the system is under attack. In such a case the *Interceptor/Actuator* is instructed to act according to a specified security policy. In fact, CReW can directly react to security breaches or delegate reaction to an external networked configuration/security management layer. CReW can also replace a misbehaving server on-the-fly by reverting back to the latest VM safe state. Alternatively the guest can be restarted from a clean backup image. CReW is integrated in the virtualization software and leverages Qemu [13] to access guest VM memory and status. Access to cloud middleware components is not mediated (see Figure 2) and allows to collect information over load level and functionality of such components. It is worth noting that no system_call is ever blocked or delayed by CReW to check for permission violation and the kind of reaction our monitoring system can perform (freezing/restarting/reinitializing) cannot be distinguished from normal system maintenance tasks.

CReW actively intercepts and monitors guest Windows VM activity by leveraging virtualization-supporting extensions of recent CPUs [11]. CReW has been implemented entirely on open source software and has been deployed over Eucalyptus. In fact, it can be tailored and applied to other distributed systems and guests. CReW can inspect Windows kernel guest state thanks to the information learnt from the

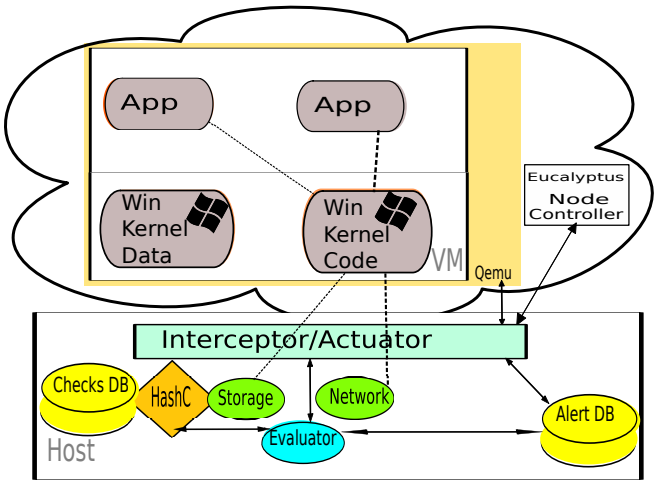

Figure 2. CReW combined with Eucalyptus - Architecture

Identifier	Category	Example
AN1	guest server misconfig.	failed web server update
AN2	kernel rootkit	see table II
AN3	colocation-based attack	multiple alloc. requests

Table I
MOST COMMON CLOUD SERVICE ANOMALIES AND THREATS

Windows research kernel [9] source code. However, CReW services can be applied to regular Windows kernels as well.

CReW enjoys the following features: it is transparent to guest machines; it leverages CPU hardware virtualization, which renders the system less detectable on guest side, and it can be deployed on Linux-hosted cloud computing platforms.

IV. EVALUATION

In this section we present the results of our experiments aimed at evaluating CReW effectiveness and performance in real-world environments, facing configuration changes, events popping-up, and threats. Tests have been conducted using Athlon 64 X2 4400+ CPUs equipped with 4GB RAM; Ubuntu 10_04-based Eucalyptus ran on the hosts, KVM version 83. Each guest was given 1 virtual CPU and 1GB RAM. All guests ran 32-bit OSes while hosts ran 64-bit OSes. The guest operating systems were Windows Server 2003 SP1.

A. CReW Effectiveness

In this section we show how CReW copes with anomalies and attacks cloud services can be subject to. In particular, we report on practical experiments performed to measure the resilience of the proposed architecture. The detection and reaction capabilities of our system are assessed against the set of anomalies and attack techniques summarized in Table I.

Attack Type	Detection Reason	Reaction
AFX	hooks native Win API	Revert to safe snapshot
Vanquish	DLL injection	Restore Libraries
Mebroot	alters disk MBR	Reinstall MBR
HackerDefender	SSDT replacement	Restore SSDT
Rustock	alters Sysenter handler	Revert to safe snapshot

Table II
CREW DETECTION/REACTION CAPABILITY

CReW has been proven to detect and to react to anomalies and attacks belonging to the above mentioned categories. In particular, we took from the current literature some relevant attacks (see Table II) that actual Windows networked architectures can be subject to and we showed the added protection provided by CReW to guest VMs when the system is exposed to such events.

In particular, we simulated an anomaly of type AN1 by causing a partial update of an IIS Server hosted on a Windows VM. CReW notices the lack of response to request network packets targeted at the HTTP service. Once the anomaly is detected, CReW restarts the compromised server from a verified executable and re-establishes its configuration files.

Then, we implemented an attack of type AN2 by inserting a Rustock rootkit [2] in a guest VM. Rustock alters the syscall handler in order to change the execution flow to execute malicious code. CReW detects both the alteration of the handler and the change in the library files checksum on virtual storage. The reaction includes reverting back to the latest safe state of the virtual machine.

Finally, we implemented attacks of type AN3 by using the techniques cited by Ristenpart in [15]. First of all, both external (outside the cloud) and internal (from sibling VMs) network probing via port scanning is intercepted by CReW rules that raise an alarm that is inserted into the Alert DB. Secondly, in order to reduce co-location time, CReW stimulates and induces transparent migration of guest Virtual Machines from one host to another over time. As regards keystroke timing [15], given that the attacker resorts to co-residence load measurements to analyze the time between keystrokes and collect sensitive information, CReW renders such attack less feasible since CReW itself is running on the CPU under attack, thus rendering times measurement results much less reliable for the attacker.

B. Performance

In order to test different kinds of workload we adopted commonly available tools on both Windows and Linux. In particular, we tested the performance of the following workloads: (1) CPU-bound; and, (2) I/O bound. We measured the time it takes a Windows cloud guest on Eucalyptus to perform two different kinds of operations:

T1 CPU stress tests: the Lame audio software is used to

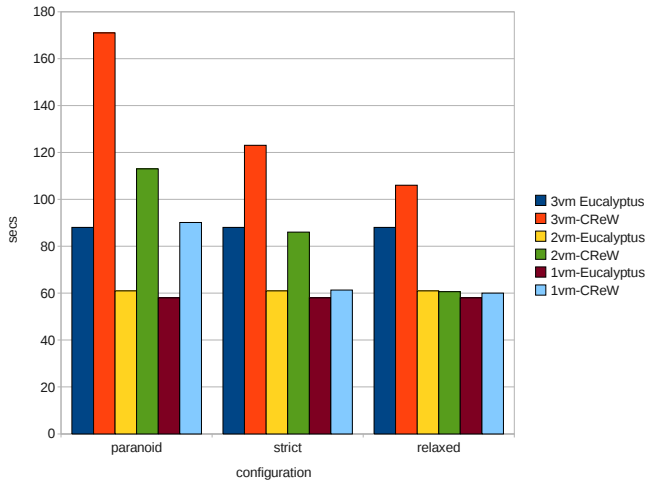

Figure 3. CReW impact on execution times - T1 - CPU-bound workload

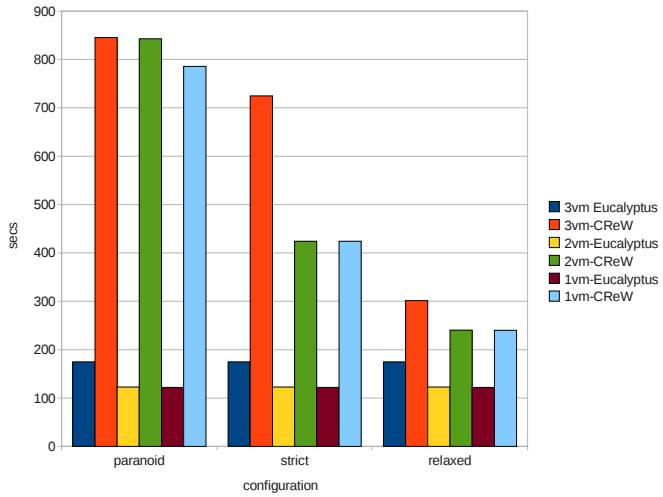

Figure 4. CReW impact on execution times - T2 - I/O-bound workload

test CPU performance when encoding WAV files to the mp3 format;

T2 I/O stress tests: the Tar archive file management software is used to test read/write disk operations and filesystem performance.

Three different approaches to monitoring have been tested for each benchmark:

(1) *paranoid*, where the monitoring checks are repeated at every interaction with the virtual machine;
(2) *strict*, where the frequency of monitoring checks is half of that for the paranoid mode;
(3) *relaxed*, where the frequency of monitoring checks is half of that for the strict mode.

Test results are shown in groups composed of six bars each. The six bars represent the result of the same test under six different scenarios:

1) first bar represents the scenario where 3 VM are running on the Eucalyptus host and CReW is inactive;
2) second bar represents the scenario where 3 VM are running on the Eucalyptus host and CReW is active;
3) third bar represents the scenario where 2 VM are running on the Eucalyptus host and CReW is inactive;
4) fourth bar represents the scenario where 2 VM are running on the Eucalyptus host and CReW is active;
5) fifth bar represents the scenario where only 1 VM is running on the Eucalyptus host and CReW is inactive;
6) sixth bar represents the scenario where only 1 VM is running on the Eucalyptus host and CReW is active;

The results of the CPU-bound tests are reported in Figure 3 where bars represent execution times (lower is better). The same set of tests has been run on an unprotected Windows guest machine on the same Eucalyptus host. Values are averaged over the tested CPUs and show that the overhead introduced by CReW is proportional to the degree of protection/control frequency, and can be made quite small in relaxed mode.

The results of the I/O-bound tests are reported in Figure 4 where, as above, bars represent execution times (lower is better). The same set of tests has been run on an unprotected Windows guest on the same Eucalyptus host. Values are averaged over the tested CPUs and show that the overhead introduced by CReW is quite large here, even though the overhead is still proportional to the degree of protection/control frequency, and can be reduced in relaxed mode.

These first results are interesting, and encourage us to perform further investigation aimed at reducing such overhead. Nevertheless, the impact on performance can be tuned as needed in order to obtain the required performance-protection trade-off. It is worth noting that even though overall performance is degraded by the monitoring system itself, such performance penalty cannot be distinguished by the attacker from regular CPU load, since no system call is ever blocked or delayed by CReW.

V. Conclusion

In this paper we provide some preliminary results on the resilience of Windows cloud guests via virtualization. In particular, we introduce a novel architecture (CReW) for transparent monitoring of Windows guests and services. CReW can effectively improve the trustworthiness in such computing systems. CReW can also react to security breaches and to system misconfiguration, improving the dependability of cloudified Windows systems. The proposed architecture has been implemented entirely on open source software. Effectiveness and performance of a first CReW prototype have been evaluated; results show the feasibility and effectiveness of such a system. The bottom line is that virtualization can increase the resilience of cloudified Windows systems and services by leveraging smart monitoring

of core components. As for further research directions, we aim to further investigate the support the CReW system can provide in securing Windows platforms.

Acknowledgments

The authors would like to especially thank Matteo Signorini and Ing. Maurizio Lancia for their support. Claudio Soriente's research was partially supported by the Spanish National Science Foundation (MICINN) under grant TIN 2010-19077, the Madrid Regional Research Council (CAM) under the CLOUDS project (S2009/TIC-1692) and EU structural funds, and the European Commission under the MASSIF project (FP7-257475).

References

[1] Amazon.com. Amazon ec2 running microsoft windows server. http://aws.amazon.com/windows, 2009.

[2] Ken Chiang and Levi Lloyd. A case study of the rustock rootkit and spam bot. In *HotBots'07: Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, pages 10–18, Berkeley, CA, USA, 2007. USENIX Association.

[3] Tobias Distler, Rüdiger Kapitza, and Hans P. Reiser. Efficient state transfer for hypervisor-based proactive recovery. In *WRAITS '08: Proceedings of the 2nd workshop on Recent advances on intrusiton-tolerant systems*, pages 1–6, New York, NY, USA, 2008. ACM.

[4] Yih Huang, David Arsenault, and Arun Sood. Closing cluster attack windows through server redundancy and rotations. In *CCGRID*, pages 21–33, 2006.

[5] Xuxian Jiang, Xinyuan Wang, and Dongyan Xu. Stealthy malware detection through vmm-based "out-of-the-box" semantic view reconstruction. In *CCS '07: Proceedings of the 14th ACM conference on Computer and communications security*, pages 128–138, New York, NY, USA, 2007. ACM.

[6] Jean-Claude Laprie. Resilience for the scalability of dependability. In *NCA '05: Proceedings of the Fourth IEEE International Symposium on Network Computing and Applications*, pages 5–6, Washington, DC, USA, 2005. IEEE Computer Society.

[7] Flavio Lombardi and Roberto Di Pietro. Kvmsec: a security extension for linux kernel virtual machines. In *SAC '09: Proceedings of the 2009 ACM symposium on Applied Computing*, pages 2029–2034, New York, NY, USA, 2009. ACM.

[8] Flavio Lombardi and Roberto Di Pietro. Secure virtualization for cloud computing. *Journal of Network and Computer Applications*, In Press, Accepted Manuscript, DOI: 10.1016/j.jnca.2010.06.008, 2010.

[9] Microsoft. Windows Research Kernel. http://www.microsoft.com/resources/sharedsource/Licensing/researchkernel.mspx, 2006.

[10] Daniel Nurmi, Rich Wolski, Chris Grzegorczyk, and al. The Eucalyptus open-source cloud-computing system. In *CCGRID '09: Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, pages 124–131, Washington, DC, USA, 2009. IEEE Computer Society.

[11] Ronald Perez, Leendert van Doorn, and Reiner Sailer. Virtualization and hardware-based security. *IEEE Security and Privacy*, 6(5):24–31, 2008.

[12] Michael Peter, Henning Schild, Adam Lackorzynski, and Alexander Warg. Virtual machines jailed: virtualization in systems with small trusted computing bases. In *VDTS '09: Proceedings of the 1st EuroSys Workshop on Virtualization Technology for Dependable Systems*, pages 18–23, New York, NY, USA, 2009. ACM.

[13] Qumranet. Linux kernel virtual machine. http://kvm.qumranet.com.

[14] Nguyen Anh Quynh and Yoshiyasu Takefuji. Towards a tamper-resistant kernel rootkit detector. In *SAC '07: Proceedings of the 2007 ACM symposium on Applied computing*, pages 276–283, New York, NY, USA, 2007. ACM.

[15] Thomas Ristenpart, Eran Tromert, Hovav Shacham, and al. Hey, you, get off of my cloud: Exploring information leakage in third-party compute clouds. In *CCS '09: Proceedings of the 14th ACM conference on Computer and communications security*, pages 103–115, New York, NY, USA, 2009. ACM.

[16] Silvio Salza, Yuri DiCarlo, Flavio Lombardi, and Roberto Puccinelli. Leveraging the grid for the autonomic management of complex infrastructures. In *GCA Grid Computing and Applications Conference Proceedings*, pages 32–37, 2006.

[17] Arvind Seshadri, Mark Luk, Ning Qu, and Adrian Perrig. Secvisor: a tiny hypervisor to provide lifetime kernel code integrity for commodity oses. In *SOSP '07: Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles*, pages 335–350, New York, NY, USA, 2007. ACM.

[18] Matthew Smith, Christian Schridde, and Bernd Freisleben. Securing stateful grid servers through virtual server rotation. In *HPDC '08: Proceedings of the 17th international symposium on High performance distributed computing*, pages 11–22, New York, NY, USA, 2008. ACM.

[19] Paulo Sousa, Alysson Neves Bessani, Miguel Correia, Nuno Ferreira Neves, and Paulo Verissimo. Resilient intrusion tolerance through proactive and reactive recovery. *Pacific Rim International Symposium on Dependable Computing, IEEE*, 0:373–380, 2007.

[20] Dazhi Wang, Bharat B. Madan, and Kishor S. Trivedi. Security analysis of sitar intrusion tolerance system. In *SSRS '03: Proceedings of the 2003 ACM workshop on Survivable and self-regenerative systems*, pages 23–32, New York, NY, USA, 2003. ACM.

[21] Xin Zhao, Kevin Borders, and Atul Prakash. Svgrid: a secure virtual environment for untrusted grid applications. In *MGC '05: Proceedings of the 3rd international workshop on Middleware for grid computing*, pages 1–6, New York, NY, USA, 2005. ACM.