Security Through Software Rejuvenation

Chen-Yu Lee, Krishna M. Kavi, Mahadevan Gomathisankaran, Patrick Kamongi

Department of Computer Science and Engineering
University of North Texas
Email: {Chen-Yu.lee,Krishna.Kavi, Mahadevan.Gomathisankaran}@unt.edu
Email: patrickkamongi@my.unt.edu

Abstract—**Software rejuvenation has been used to improve reliability of systems by periodically checkpointing and restarting them. In this paper, we propose to use rejuvenation as a mechanism to enhance the security of Cloud infrastructure and eliminate malware by continuous and periodic rejuvenation. To evaluate the effectiveness of rejuvenation in eliminating malware, we defined an experimental setup, and utilizing complete system rejuvenation, as well as application level rejuvenation we investigated which malware were eliminated. We also describe a cost model for rejuvenation so that one can determine how often systems and applications should be rejuvenated, trading cost against security. Our experiments and models show that rejuvenation once every 24 hours is cost-effective.**

***Keywords*–Rejuvenation; Malware; Security; Vulnerability.**

I. INTRODUCTION

Computer viruses have been evolving into more complex malware and the detection and elimination of such threats is becoming very expensive in large IT operations. The number of new types of malware detected over the past ten years has increased very rapidly since 2010.

Software Rejuvenation technology was first proposed by Lin in 1993 [1]. The author observed that system performance degrades with time, and failure rates also increase with time. This phenomenon was termed software *aging*. A proactive solution to this problem is to gracefully terminate an application or a system and restart it in a clean internal state, known as *software rejuvenation* [2]. Rejuvenation technology was originally used for software fault tolerance [3] [4]. The most relevant work that applies rejuvenation for protection against security attacks is SWRMS proposed by Aung in 2004 [5]. The authors propose to identify attacks using an intrusion detection system, and then perform software rejuvenation to counteract these attacks, including killing the intruders' processes, halting abuse, shutting down unauthorized connections, and restarting applications. The attacks, however, are not eliminated if the processes are infected. They do not rely on rollback to restart infected processes from a known clean state. Moreover, since the approach is based on detecting intrusions, one should include the cost of detecting attacks along with the cost of rejuvenation, to estimate the total cost of their approach. We do not base rejuvenation on detecting attacks; rejuvenation is applied regularly. Along with rejuvenation, we restart processes from a checkpointed or clean state.

More generally, we believe that rejuvenation can either be used in place of scanning to detect attacks and malware, or in addition to scanning. To evaluate the effectiveness of rejuvenation against malware and viruses, we created a testbed that performs both system level and application level checkpointing and restarting. We then introduced known malware and verified if the malware was eliminated after the restart. The testbed also provides for a realistic evaluation of the cost of rejuvenation and which malware can be eliminated using rejuvenation. In this paper, we also develop a model to compare the cost of rejuvenation with the cost of scanning for malware. We emphasize that rejuvenation is more than just restarting of systems, it also includes checkpointing software applications and systems in clean states, and periodically rolling back the software to known clean states.

The rest of the paper is organized as follows: Section II introduces how rejuvenation can be used to enhance security. This section also introduces a model for estimating the cost of rejuvenation. Section III shows the simulations results of rejuvenation on our web service built by Joomla [6] and OpenStack [7]. Section IV compares rejuvenation with scanning approaches. Section V provides our conclusions and future work.

II. REJUVENATION FOR SECURITY

A. Environment Description

The proposed rejuvenation is applied to Cloud computing [8] environments to enhance security and stability of the systems. Many commercial operations rely on Cloud computing and in such applications, maintaining low Mean Time To Repair (MTTR) and the cost of repair are essential to the profitability of the operations. Therefore, they normally use software patches to fix problems, instead of completely overhauling their systems. The patches include system patches, software patches, malware/virus signatures, firewall rules, etc.

B. Work Flow of Rejuvenation for Security

We propose to use periodic rejuvenation (i.e.,checkpoint, rollback, recover and restart) to improve security and reliability of components. The rejuvenation can be applied modularly to minimize the downtime of the system. Each module is restored (or rejuvenated) to a clean checkpoint and reconnected with other related modules. Patches can also be applied to modules during a rejuvenation to reduce their vulnerabilities and to eliminate detected malware. The patches should be verified as clean and distributed by authorized providers, to assure that patched modules are clean. The work flows are shown in Figure 1, and the main processes are described below:

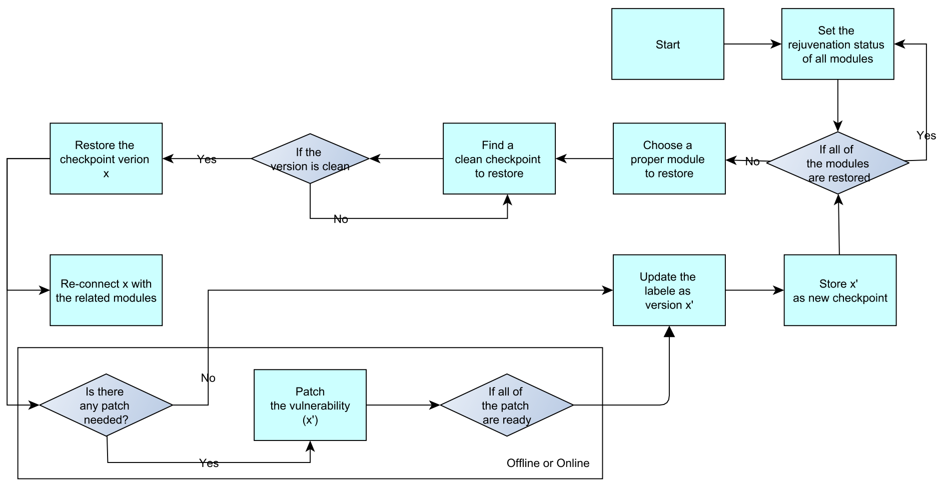
Start
Set the rejuvenation status of all modules
Yes
If all of the modules are restored
No
Choose a proper module to restore
Find a clean checkpoint to restore
If the version is clean
Yes
No
Restore the checkpoint verion x
Re-connect x with the related modules
Update the labele as version x'
Store x' as new checkpoint
Is there any patch needed?
No
Yes
Patch the vulnerability (x')
If all of the patch are ready
Offline or Online

Figure 1. The workflow of secure rejuvenation mechanism

- Checkpoint: When a new software module is tested, verified, and ready to go online, it is assumed to be in a clean state and a checkpoint of the module is taken. Periodically, the module is rolled back to the clean checkpoint to scrub the module of any infections. If any design fixes or other patches are made available to the module since its original release (and the patches are verified as trusted and clean), the module is upgraded during the rejuvenation period, and the checkpoint image is also updated to the new clean state.
- Recover: All modules of a system go through a rejuvenation process (checkpoint-recovery) periodically, where the periodicity is determined based on the cost of rejuvenation and the frequency of new malware introductions. The process eliminates not only software aging and soft or intermittent faults, but also some malware. The rejuvenation may also be performed when an abnormal condition or a suspected security threat is detected.
- Restart: The module always restarts after each recovery. This eliminates software aging and some common security threats, including denial of service (DoS) and others.

III. Simulation Results

A. Simulation Settings

To understand how rejuvenation can eliminate malware, we built a Joomla content management service on our private Cloud environment supported by OpenStack [7]. The specification of our system is shown in Table I.

B. Results and Analysis

The rejuvenation is divided into two types: a complete system rejuvenation and component rejuvenation. In our simulation, the complete system rejuvenation is provided by OpenStack which creates an instance snapshot stored in the snapshot repository, and restores it while launching the complete rejuvenation. It supports live snapshotting, which allows for taking snapshots of the running virtual machines without pausing them. In the second case, we only take checkpoints of an application or components of applications and periodically restore them.

In the experiment, we tested some malware and vulnerabilities listed in Table II. Most of the attacks are at the Operating Systems (OS) level: they create backdoors for bot or other attacks by using rootkit or other related technologies. Under the attack, the malware gains root privileges, hides itself, and deletes itself from the log. We perform our experiment in two phases.

Phase 1. Inject malware and scan the complete system to make sure the malware is in the system and detectable by anti-malware software. NOD32 [9] and ClamAV [10] are applied in our experiments.

Phase 2. Restore the checkpointed version and then scan to find if the malware is eliminated by rejuvenation. If the malware is not eliminated by the rejuvenation, it should be detected by anti-malware software.

The experimental result shows that the complete system rejuvenation eliminates all the malware we introduced and recovers all of the infected files. We also simulated changes to the integrity of MySQL [11] files using the known vulnerabilities of MySQL. After restoring, the modified files are recovered correctly. The rejuvenation can only recover the

TABLE I. textscSimulation Environment Specification

Platform	Version
Cloud platform	OpenStack
Flavor	m1.small
RAM	2GB
Processor	QEMU Virtual 1.0@2.33GHz(1Core)
Instance operation system	Ubuntu 12.04
Instance Size	20 GB
Application service	Joomla 3.3
Database	MySQL 5.5.36
Compiler	PHP 5.4.27
Web service	Apache 2.4.9
Anti-malware software	ClamAV 0.98.3
Anti-malware software	F-prot 6
Anti-malware software	Nod32 4

infected files, but the vulnerabilities still exist making the system vulnerable for repeated attacks. Vulnerabilities can only be fixed with appropriate patches. We also tested to show that rejuvenation can rescue the system from Denial of Service (DoS) [12] or low-rate DoS attacks [13].

TABLE II. TEST MALWARE AND VULNERABILITIES

Malware	Scope	Rejuvenation, Result
Backdoor.Linux.Ovason	Operation System	Restore, Eliminated
Backdoor.Linux.Phobi.l	Operation System	Restore, Eliminated
Backdoor.Linux.Rst.a	Operation System	Restore, Eliminated
Exploit.Linux.Da2.a	Operation System	Restore, Eliminated
Exploit.Linux.Race.l	Operation System	Restore, Eliminated
Net-Worm.Linux.Scalper.b	Operation System	Restore, Eliminated
Rootkit.Linux.Agent.sm	Operation System	Restore, Eliminated
Trojan.Linux.Rootkit.n	Operation System	Restore, Eliminated
Trojan.Tsunami.B	Operation System	Restore, Eliminated
VirTool.Linux.Mhttpd	Operation System	Restore, Eliminated
Virus.Linux.Osf.8759	Operation System	Restore, Eliminated
Virus.Linux.Radix	Operation System	Restore, Eliminated
Virus.Linux.Silvio.b	Operation System	Restore, Eliminated
Virus.Linux.Snoopy.c	Operation System	Restore, Eliminated
Vulnerability	**Scope**	**Rejuvenation, Result**
CVE-2013-1636	Joomla	Restore, Recovered
CVE-2014-2440	MySQL	Restore, Recovered
CVE-2014-2436	MySQL	Restore, Recovered
Attack	**Scope**	**Result**
Denial of Service (DoS)	Apache	Reboot, Recovered
Low-rate Dos	Apache	Reboot, Recovered

C. *Performance and Cost*

Performance of rejuvenation (the time spent for rejuvenation) is important because it relates to the unavailability or downtime of the system or service. In this section, we report the performance of our simulations in two parts: time spent for rejuvenation and the storage space required for storing checkpointed information.

1) Time spent and storage space costs: For the complete rejuvenation experiment, we checkpointed and restored the working instance. But, checkpointing can be performed without any downtime; restoring, however causes 54 seconds of downtime. For component rejuvenation, we set up a checkpoint on Apache and MySQL database software in our experiments with 850 MB of data without pausing; the chekpoint images used 327MB of memory. Application level rejuvenation required 187.51 seconds (includes the time to retrieve checkpointed images, stopping the application and restarting the application using the checkpointed image).

2) Cost: Globalscape found that in 60% of Fortune 500 companies, a single hour without critical systems costs their company between $250,000 and $500,000 and one in six companies reported that one hour of downtime can cost $1 million or more [14].

Assuming that on average $500,000 of loss per hour of downtime, our experiments show a cost of $7500 for complete system rejuvenation, $26,043 for rejuvenating Apache and MySQL software. The cost of the storage needed for checkpoint images are $2.4 and $0.12 respectively based on Amazon's prices. The cost model is described in Section IV-B.

IV. ANALYSIS AND COMPARISON OF REJUVENATION FOR SECURITY

In this section, we compare the capabilities in term of defense against various security threats and cost associated with rejuvenation and malware scanning techniques.

A. *Characteristics Comparison*

Rejuvenation has been used as a fault-tolerant/fault-avoidance approach in software systems. In a similar manner, rejuvenation can be applied as a defense against security threats. By restoring components to clean or healthy states, rejuvenation can make the system less prone to catastrophic failures. In Table III, we compare the capabilities of rejuvenation with malware scanning when applied to survivable systems. In Section III, our experiments have shown that rejuvenation can eliminate or mitigate the effects of several types of malware. Some weaknesses that cannot be eliminated using rejuvenation include trapdoors, which are eliminated by compiler-based code checkers and detected by resource monitors. But, anti-malware software need to monitor and scan entire memory and file systems to detect malware and eliminate or quarantine the infected files.

TABLE III. THE COMPARISON OF THE FEATURES, AND THE ABILITIES OF THREAT ELIMINATION BETWEEN REJUVENATION AND MALWARE SCANNING FOR SECURITY.

Feature	Rejuvenation	Malware Scanning
Fault avoidance	Partial	No
Fault tolerance	Yes	Yes
Denial of Service(DoS) or Low-rate DoS	Reboot	Log analysis
Virus elimination	Restore to checkpoint	Scanning
Trojan horse elimination	Restore to checkpoint	Scanning
Trapdoors elimination	No	No
Automated software-patching	Yes	Yes
Intrusion dection	No	Yes

B. *Cost Model*

This section discusses the cost of performing rejuvenation compared with malware scanning more formally.

Malware scanning software (e.g., anti-malware software) is usually performed as a daemon, scanning all the stored files, executing processes, system kernel and other system software continuously. Scanning may detect more security threats than that can be eliminated using only rejuvenation. However, scanning for malware consumes computational resources and thus the following model can be used for estimating the cost of malware scanning ($CoMS$).

- Instance size(V): The cost of scanning is proportional to the size of the system being scanned. In addition to scanning of the system at startup, malware scanning occurs continuously and is invoked when changes to the system are detected (such as file updates, internet downloads, mail attachments or other changes to the system state, such as changes to page tables). In this paper, we relate the cost of scanning to the average volume of the new information that must be scanned over a given period of time. The period of time and the volume of data scanned are compared with the rejuvenation period and the volume of information involved in the rejuvenation process.
- Scan speed(SS): This is the rate at which a system can be scanned to detect malware or virus signatures.
- Cloud computing fee (CCF): The fee charged by Cloud providers (whether the computing is used for scanning or for providing services).

The total cost involved with malware scanning C_{MS} for size V over a chosen time period T is

$$C_{MS}(V, T) = \frac{V \times CCF}{SS} \tag{1}$$

As an example, if it is assumed that the scanning speed is 26.58 MB/sec [15] and the computing fee charged by Amazon EC2 is \$0.176, \$0.351, \$0.702, and \$1.404 per hour for instance size 4, 32, 40, and 80GB [16], the cost of performing one malware scan on a Cloud environment with 10 GB size data would be \$0.007, \$0.117, \$0.293, and \$1.173 respectively.

In this paper, we assume two types of rejuvenation: a regular, periodic rejuvenation at fixed periods, and ad hoc rejuvenations when anomalies or threats are detected. Thus, factors that contribute to the cost of rejuvenation are divided into two parts. One is the cost involved with rejuvenation (CoR), and the other is involved with monitoring (CoM) to detect anomalies. Rejuvenation makes some modules unavailable (downtime) during the process of restoration. The following are the factors that influence the cost of rejuvenation.

- Downtime (DT): While performing the rejuvenation, some modules will be unavailable and the downtime can range from a few seconds to a few minutes.
- Number of transactions lost (TL): The number of transactions lost during the downtime.
- Potential loss of revenue associated with each transaction (PR) that could not be completed during downtime.
- Version storage fee (SF): Since clean modules and checkpointed states must be saved, we include the cost of storage with rejuvenation. In some cases, we may need to save m snapshots or checkpoints to fully recover the system to a clean state. Thus we include the total cost of storage needed for checkpointing. This can be compared with the volume scanned by malware scanners.
- Data transfer fee (TF): We assume that the checkpoints are stored in a backup or archival facility and this information has to be transferred to executing environments during restoration. We include the data transfer costs for transferring n bytes of data transferred between an execution environment and backup facility.

The total cost involved with rejuvenation $CoR_{periodic}$ for size V, with a rejuvenation period of T, is

$$CoR_{periodic}(V, T) = DT \times TL \times PR + mV \times SF + V \times TF \tag{2}$$

In addition to periodic rejuvenation, ad hoc rejuvenation (rollback to clean checkpoint and restart) is also applied when an abnormal condition or a security violation is detected. The detection may be based on monitoring system performance or other indicators. For example, performance indicators, including memory allocations, CPU usage, network traffic, disk writes, may indicate abnormal behavior of applications. We will include the cost of monitoring ($CoM(V, t)$) the system to identify abnormal conditions in the cost of rejuvenation. The cost depends on the volume V of information monitored.

$$CoR_{adhoc}(V, t) = CoM(V, t) + CoR_{periodic}(V, t) \tag{3}$$

Since the ad hoc rejuvenation can take place at any time between scheduled periodic rejuvenations, we will use a probability distribution that associates the probability of detecting an abnormal condition over this period of time. We can now compute the expected cost of rejuvenation that includes both ad hoc and periodic rejuvenation as follows.

$$CoR_{total}(V, T) = \int_0^T f(t)(CoM(V, t) + CoR_{periodic}(V, t))\,\mathrm{d}t \tag{4}$$

Here, $f(t)$ is the probability density function that reflects the probabilities of detecting abnormal behaviors. $f(t)$ varies depending on the security environment of the institution. If the systems are not protected, the probability of detecting an abnormal system may be higher. T is the scheduled rejuvenation period.

Consider for example that it takes 17 seconds to rejuvenate a system (i.e., the downtime is 17 seconds [17]), the average number of transactions lost in a year is 355.72 [17], the average potential revenue of a transaction lost is \$100,000, and the storage fee charged by Amazon is \$0.095 per GB-month and data transfer fee is \$0.12 per GB, the cost of performing each periodic rejuvenation is \$19.1756 for the 10 GB cloud instance. If we assume that in addition to hourly scheduled rejuvenation, ad hoc rejuvenations are warranted with a probability of 10% in between scheduled rejuvenations and if we assume that monitoring consumes 0.1% of CPU time, the total cost of rejuvenation can be estimated as

$$CoR_{total}(10GB, 1hr) = 2.03756 + 19.1756 \tag{5}$$

C. Cost Comparison

Figure 2 shows the cost of rejuvenation performed periodically for different frequencies: four hours, six hours, 12 hours, and 24 hours over a year. The cost of rejuvenation over a year depends on the frequency of rejuvenation and the cost of each rejuvenation. In Figure 2, we did not include monitoring and ad hoc rejuvenation costs, since these costs depend on the probability of detecting an abnormal condition. The figure also shows the cost of scanning for malware. The cost of scanning depends on the size of the system being scanned. The red horizontal lines represent the cost of scanning continuously ($Frequency = 0$). We also show the cost of malware scanning when scanning takes place at four, six, 12 and 24 hour periods - similar to rejuvenation. Systems need only to scan new information generated during the period and we assume that the amount of new information generated is proportional to the length of the period. It can be seen that the cost of rejuvenation decreases with the decrease in frequency (less frequent rejuvenation). The cost of continuously scanning for malware (see the three dash lines) is higher than rejuvenation at certain rejuvenation periodicities. If one assumes that

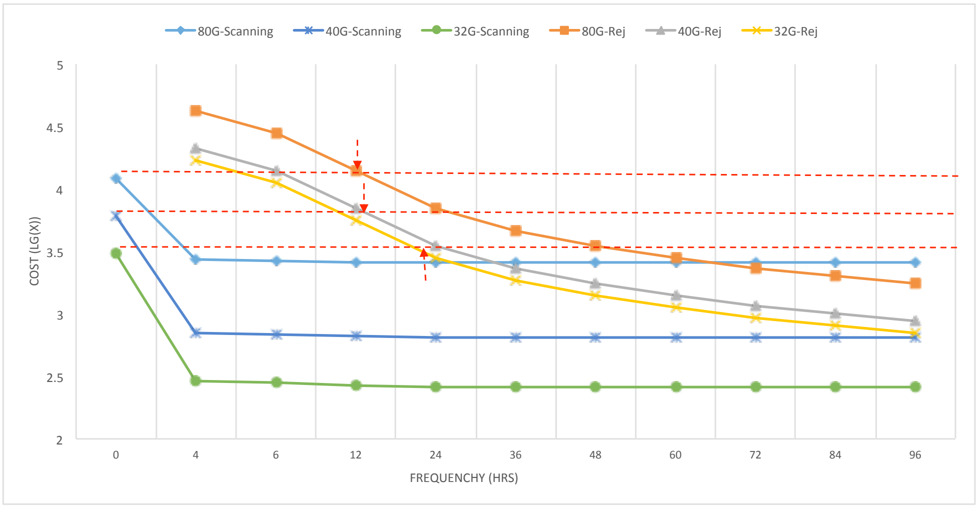

Figure 2. Cost comparison of secure rejuvenation versus malware scanning

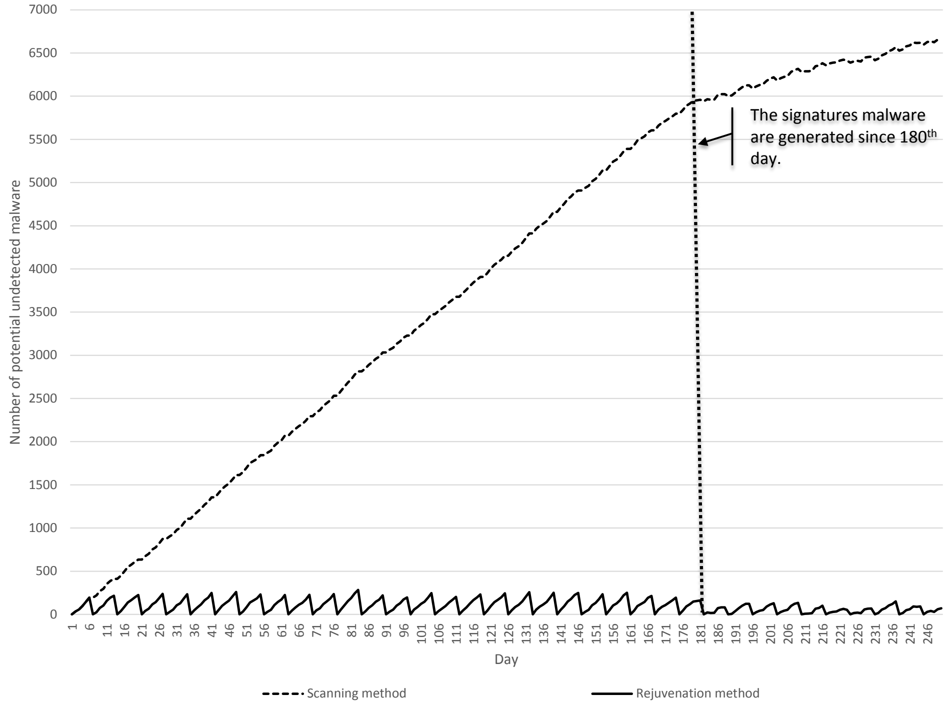

Figure 3. The potential number of malware remaining in a system after use of scanning versus rejuvenation

systems scan for malware at fixed internals (such as every four hours), rejuvenation costs are higher except when it takes place once every 80-100 hours. Based on our assumptions and cost models, rejuvenation once every 24 hours appears to be a reasonable choice for different system sizes.

D. Undetectable Malware Elimination

Malware is getting more sophisticated and the sophistication is increasing in recent years. McAfee's report shows that there are over 100,000 new malware instances detected in a given day [18]. There are three phases in the detection and elimination of malware. The first is the undetected phase in which the malware strain was not detected in the system. The second is the identification phase in which the malware strain is detected as a malicious code pattern and its signature is generated. Finally, the malware strain enters the detected phase after its signature is updated. A study by Damballa demonstrated that the typical gap between malware release and detection using anti-malware is 54 days, almost 8 weeks [19]. Nearly half of the 100,000 malware samples go undetected on the first testing day, and there were at least 15% of the samples remaining undetected even after 180 days. This means that the system may suffer from undetected malware for long periods of time.

Suppose a system component is infected with an average of 30 malware strains every day since it is released, the number of potential malware strains hidden may increase over the next several weeks before some strains are detected. On average it will be 9 weeks before detected malware signatures are released, and the number of hidden malware will be reduced as shown in Figure 3.

By contrast, the proposed rejuvenation mechanism periodically restores the component to a ”clean” version (checkpoint); thus, the exposure of the system to new malware introduction is the time between rejuvenations. Assuming that the component is rejuvenated once a day, it remains in ”clean” status at the beginning of each day. After the 9th week, some malware strains are eliminated because of the signatures, thus the potential malware strains lurking may decrease as long as the backup version is not infected.

E. Complete Rejuvenation and Component Rejuvenation

The services can be rejuvenated one service at a time such that the impact of rejuvenation is not felt by the entire system. The performance of rejuvenation depends on the instance’s capability. By taking our experiment as an example, the instance works with flavor m1.small, thus the ability of checkpointing smaller size files is slower than checkpointing of complete instance performed by the host. Furthermore, any patches or upgrades to services can be done separately from a running system.

F. Application of Mobile Device

Kaspersky Lab’s report shows that approximately 10,000,000 unique mobile malicious installation packages were detected in 2012-2013 [20]. Sometimes mobile malware resists the anti-malware protection because of Android vulnerabilities. Malware uses the vulnerabilities to bypass the code check, enhance the privilege to extend their capabilities, and make it more difficult to be removed, like Trojan-SMS.AndroidOS.Svpeng.a. Therefore, it is difficult for normal users to remove malware, since most of the malware is embedded in the legitimate software and acquires administrator privilege during the installation. There are only two options for users. One is to reset the system to factory settings, but some malware could obstruct this reset. The other is to apply anti-malware software to continuously scan, analyze, and eliminate it; but this consumes processing and thus the battery life of the device.

Our rejuvenation mechanism can be applied on Virtual Machine-based environments, such as cloud services, as well as mobile devices (e.g., Android). Our rejuvenation mechanism restores the checkpointed image from either the storage of the device or from some external storage in the Cloud, or may rely on trusted zones to bring the system to a clean or consistent state. If the rejuvenation is performed while the device is connected to a power source, the battery life will not be a consideration. Rejuvenation can be performed on a regular basis, similar to checking periodically for software patches and upgrades. A rejuvenation mechanism, therefore, is more suitable in a mobile environment, than malware scanning techniques.

V. Conclusion

The cybersecurity of Cloud-based computing systems are becoming critical to modern society as we are becoming ever more dependent on information infrastructures. Balancing system reliability, availability and security is complex. Malware and other security threats are becoming more sophisticated. Thus a multipronged approach is necessary to improve security as well as system survivability. We feel that software rejuvenation, which has been successfully employed as a fault-tolerance mechanism, can also be used as a defense against security threats. We conducted experiments in a controlled environment to show that rejuvenation does eliminate some malware. We will extend our experiments to more thoroughly evaluate which types of malware can and cannot be eliminated with rejuvenation only. In this paper, we also introduced a model that can be used to compare the costs associated with rejuvenation and malware scanning so that one can determine the rejuvenation frequencies that lead to cost-effective defense against hidden threats. While we compared rejuvenation as an alternative to scanning in this paper, they should be used together.

Acknowledgment

This research is supported in part by the NSF Net-centric and Cloud Software and Systems Industry/University Cooperative Research Center and NSF award 1128344 and 1332035. The authors also acknowledge David Struble’s help in making the paper more readable.

References

[1] F. Lin, “Re-engineering option analysis for managing software rejuvenation,” Information and Software Technology, vol. 35, no. 8, 1993, pp. 462–467.

[2] Y. Huang, C. Kintala, N. Kolettis, and N. D. Fulton, “Software rejuvenation: analysis, module and applications,” in Proceedings of the 25th International Symposium on Fault-Tolerant Computing (FTCS-25), 1995, pp. 381–390.

[3] R. Agepati, N. Gundala, and S. V. Amari, “Optimal software rejuvenation policies,” in Proceedings of the Reliability and Maintainability Symposium (RAMS), 2013, pp. 1–7.

[4] S. Oikawa, “Independent Kernel/Process Checkpointing on Non-Volatile Main Memory for Quick Kernel Rejuvenation,” in Proceedings of the 27th International Conference on Architecture of Computing Systems (ARCS), ser. LNCS. Springer, 2014, vol. 8350, pp. 233–244.

[5] K. M. M. Aung and J. S. Park, “Software Rejuvenation Approach to Security Engineering,” in Proceedings of the International Conference on Computational Science and Its Applications (ICCSA), ser. LNCS. Springer, 2004, vol. 3046, pp. 574–583.

[6] “Joomla,” URL: http://www.joomla.org [accessed: 2014-07-28].

[7] “OpenStack,” URL: http://www.openstack.org [accessed: 2014-07-28].

[8] P. Mell and T. Grance, “The NIST Definition of Cloud Computing,” NIST, Tech. Rep. SP800-145, Sep. 2011.

[9] NOD32, URL: http://www.eset.com [accessed: 2014-07-28].

[10] ClamAV, URL: http://www.clamav.net/ [accessed: 2014-07-28].

[11] MySQL, URL: http://www.mysql.com/ [accessed: 2014-07-28].

[12] Z. Tan, A. Jamdagni, X. He, P. Nanda, and R. P. Liu, “A System for Denial-of-Service Attack Detection Based on Multivariate Correlation Analysis,” IEEE Transactions on Parallel and Distributed Systems, vol. 25, no. 2, 2014, pp. 447–456.

[13] Y. Tang, X. Luo, Q. Hui, and R. Chang, “Modeling the Vulnerability of Feedback-Control Based Internet Services to Low-Rate DoS Attacks,” IEEE Transactions on Information Forensics and Security, vol. 9, no. 3, 2014, pp. 339–353.

[14] “Three Ways System Downtime Affects Companies and Four Methods to Minimize It,” Globalscape, Tech. Rep., 2014.

[15] “Scan Speeds for 2011/2012 AntiVirus Software,” Antivirus Ware, 2011, URL: http://www.antivirusware.com/testing/scan-speed/ [accessed: 2014-07-28].

[16] “Amazon EC2 Price,” Amazon Web Services, 2013, URL: http://aws.amazon.com/ec2/pricing/ [accessed: 2014-07-28].

[17] F. Machidaa, D. S. Kim, and K. S. Trivedi, “Modeling and analysis of software rejuvenation in a server virtualized system with live VM migration,” Performance Evaluation, vol. 70, 2013, pp. 212–230.

[18] “Infographic: The State of Malware 2013,” McAfee, Inc., Tech. Rep., Apr. 2013, URL: http://www.mcafee.com/us/security-awareness/articles/state-of-malware-2013.aspx [accessed: 2014-07-28].

[19] “3% to 5% of Enterprise Assets Are Compromised by Bot-driven Targeted Attack Malware,” Damballa, Inc., Tech. Rep., Mar. 2008, URL: http://www.prnewswire.com/news-releases/3-to-5-of-enterprise-assets-are-compromised-by-bot-driven-targeted-attack-malware-61634867.html [accessed: 2014-07-28].

[20] V. Chebyshev and R. Unuchek, “Mobile Malware Evolution: 2013,” Kaspersky Lab ZAO, 2013, URL: http://www.securelist.com/en/analysis/204792326/Mobile_Malware_Evolution_2013 [accessed: 2014-07-28].