A performance model of the Trusted Cloud Computing Platform VM Launch Protocol

Said Naser Said Kamil
School of Computing Science
Newcastle University, UK.
said.kamil@ncl.ac.uk

Nigel Thomas
School of Computing Science
Newcastle University, UK.
nigel.thomas@ncl.ac.uk

ABSTRACT

The adoption of the cloud computing paradigm is associated with increasing security concerns. Cloud computing service models (SaaS, PaaS and IaaS) are exposed to different security threats in each level of services. The Trusted Cloud Computing Platform (TCCP) proposes a security model that protects customers VMs at the IaaS level. In this paper, we investigate methodologies for the specification and scalability of a performance model and evaluation of the VM Launch security protocol within the TCCP model. The Markovian process algebra PEPA has been used to specify and analyse the VM Launch protocol.

CCS Concepts

•**Computer systems organization** → **Cloud computing;** •**Security and privacy** → *Security protocols;* •**General and reference** → **Performance;** *Evaluation;*

Keywords

TCCP security protocol, Cloud Computing, PEPA.

1. INTRODUCTION

Utility cloud computing has attracted a considerable number of companies and individuals to handle and manage large data and scalable business services. However, confidentiality and integrity of data remain significant concerns. Therefore, numerous security improvements have been applied by cloud providers, in order to guarantee the trustworthiness of their offered services. Undoubtedly, several security challenges are impacting the use of the cloud computing, particularly security risks in Virtual Machines (VMs), for example hypervisor rootkits (BluePill and VMM Rootkit Framework) [2, 4, 7].

One of the solutions to this problem is the Trusted Cloud Computing Platform (TCCP), proposed by Santos *et al* [8], which aims to offer a secure execution environment for customers through a set of security protocols working at the Infrastructure as a Service (IaaS) level that prevent the admin at the backend of the cloud provider from inspecting or tracing customers VMs. In addition, TCCP allows the user to determine whether the platform that will host the VM is secure before the real launch of the VM, using the Trusted Coordinator (TC) hosted by an external third party.

In our previous work [6] the Node Registration security protocol, which is the initial part of the TCCP system, has been modelled and evaluated using PEPA [5]. This paper aims to investigate methodologies to evaluate the performance of the VM Launch protocol, which is the second part of the TCCP system. The PEPA Eclipse Plug-in tool [10] will be used to analyse the behaviour of the protocol. Additionally, an investigation will be carried out for the scalability and the validity of the proposed PEPA model. The paper is organized as follows. Starting with an overview of the architecture of the TCCP system, then illustrating the design and mechanism of the VM Launch protocol. This is followed by a description of the PEPA model of the VM Launch protocol. Next the results of a variety of scenarios that have been considered are shown and discussed. The paper is closed with a conclusion and further work.

2. TRUSTED CLOUD COMPUTING PLATFORM

With the aim of enabling the building of trusted platforms the Trusted Computing Group (TCG) [1] proposed a design of the Trusted Platform Module (TPM) chip; where the TPM is identified uniquely by an endorsement private key (EK) and unmodifiable cryptographic functions. TPM features are leveraged by a number of trusted platforms such as Terra [3] with the aim of enabling remote attestation. Although it is a valuable associated security mechanism, there is a limitation in this approach where the cloud manager can inspect or tamper the customers' virtual machines at the backend of IaaS. As such, the TCCP method extends the approach of the trusted platform to include the whole backend of IaaS.

The Trusted Cloud Computing Platform (TCCP) [8] is designed based on trusted computing technologies. The proposed design of TCCP provides cloud service providers (IaaS), for instance, Amazon EC2, with a closed box mechanism that ensures confidentiality and integrity of hosted VMs. This enables customers to decide if the service pro-

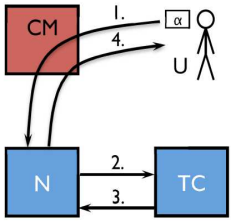
VALUETOOLS 2015, December 14-16, Berlin, Germany

DOI 10.4108/eai.14-12-2015.2262693

vided via the IaaS provider is secure or not, before the real launching of the VMs operation, through a secure remote attestation. Furthermore, the TCCP system uses the Trusted Virtual Machine Monitor (TVMM) to certify that the untrusted Cloud Manager (i.e. administrator) at the backend of the IaaS provider is not allowed to access or modify the VMs of customers, where the trusted nodes will be managed by the Trusted Coordinator (TC) which is hosted and maintained by a third party External Trusted Entity (ETE). The cloud manager (CM) has no privileges in the ETE.

2.1 Virtual Machine Launch

Before launching the VM the TCCP needs to ensure that a set of requirements, including: the VM can only be launched on a trusted node, and the administrator at the backend cannot inspect or modify the customers VM. Furthermore, the initial VM state α contains the VM image (VMI) and the public key of the user, whereas the VMI can be personalized and contain secret data; and the public key used for ssh login purposes. Moreover, the TCCP allows the user to decide whether to use a VMI provided or not. Figure 1 illustrates the VM launch protocol and the involved parties, which required to fulfil the mentioned requirements.



1. $\{\alpha, \#\alpha\}_{K_{VM}}\{n_U, K_{VM}\}_{TK^P_{TC}}$
2. $\{\{\{n_U, K_{VM}\}_{TK^P_{TC}}, n_N\}_{TK^p_N}, N\}_{TK^P_{TC}}$
3. $\{\{n_N, n_U, K_{VM}\}_{TK^P_N}\}_{TK^p_{TC}}$
4. $\{n_U, N\}_{K_{VM}}$

Figure 1: TCCP message exchange during VM launch [8].

The VM launch protocol is designed as follows: firstly, the user starts by generating a key session K_{VM}, then sends the first message to the CM, which contains: α and hash α encrypted with the K_{VM}; and the session key encrypted with the TC trusted public key TK^P_{TC}, to ensure only the TC can authorise someone to access α. Once the CM receives the request from a user, the CM will designate the node N from a cluster of trusted nodes to host the VM, and the request will be forwarded to N. As the node cannot access α, it will send the second message to the TC encrypted by TK^p_N, which means that the TC will be able to verify whether the node is trusted, or not. The request will be denied in the case of the corresponding public key not found in the TC's trusted node database. On the other hand, the node is trusted and the TC will decrypt the K_{VM} and sends it back to the node N in the third message, encrypted by TK^p_{TC}. Accordingly, N is able to decrypt α and boot the VM. As a final confirmation, the identity of N running the VM will be sent to the user by the node in the fourth message via the CM.

3. VM LAUNCH PEPA MODEL

The Performance Evaluation Process Algebra (PEPA) modelling paradigm has been used to create the VM Launch Model, with the purpose simulating the behaviour of the VM launch protocol. PEPA is a Markovian process algebra, meaning that the system described gives rise to a continuous time Markov chain (CTMC) which can be analysed to find performance metrics of interest. A formal definition of PEPA is given by Hillston [5]. The parties involved in the VM launch operation (i.e. *User, CM, N,* and *TC*) have been modelled in PEPA as components and each component has several independent actions along with shared actions representing the message passing. Furthermore, each action has a rate which is the reciprocal of the average duration, or delay, to undertake by the action. The components have each been modelled as a set of sequential actions to simulate the protocol behaviour.

The actions of *User* component which interacts with the *CM* are shown below.

$$
\begin{aligned}
User &\overset{def}{=} (generate\_Kvm, r1).User0\\
User0 &\overset{def}{=} (sendAlphaHashAlphaEncreptedBy\_Kvm, r2).User0a\\
User0a &\overset{def}{=} (key\_KvmEncreptedBy\_TKP\_TC, r3).User1\\
User1 &\overset{def}{=} (receiveRequest, r4).User2\\
User2 &\overset{def}{=} (forward\_N\_id\_To\_User, r17).User3\\
&\quad + \ (end, r13).User\\
User3 &\overset{def}{=} (useVM, r18).User
\end{aligned}
$$

The *CM* actions that organize the requests received from users and controls the nodes are as follows:

$$
\begin{aligned}
CM &\overset{def}{=} (receiveRequest, r4).CM1\\
CM1 &\overset{def}{=} (designate\_N, r5).CM2\\
CM2 &\overset{def}{=} (forwardRequestTo\_N, r6).CM3\\
CM3 &\overset{def}{=} (send\_N\_id\_To\_User\_EncreptedBy\_Kvm, r16).CM4 + (end, r13).CM\\
CM4 &\overset{def}{=} (forward\_N\_id\_To\_User, r17).CM\\
&\quad + \ (end, r13).CM
\end{aligned}
$$

Below are the actions of the Node (N) component which is responsible for receiving the requests from the *CM* and communicating with the *TC* in order to launch the VM.

$$
\begin{aligned}
N &\overset{def}{=} (forwardRequestTo\_N, r6).N0\\
N0 &\overset{def}{=} (sendMessageTo\_TC\_EncrecptedBy\_TKp\_N, r7).N1\\
N1 &\overset{def}{=} (sendDecrypted\_Kvm\_To\_N, r11).N2\\
&\quad + \ (end, r13).N\\
N2 &\overset{def}{=} (decrypt\_alpha\_bootVM, r14).N3\\
N3 &\overset{def}{=} (send\_N\_id\_To\_User\_EncreptedBy\_Kvm, r16).N
\end{aligned}
$$

The following component is the *TC* that manages a set of trusted nodes and control the authority of nodes.

$$
\begin{aligned}
TC &\overset{def}{=} (sendMessageTo\_TC\_EncrecptedBy\_TKp\_N, r7).TC0\\
TC0 &\overset{def}{=} (searchInDatabaseFor\_TKP\_N, r8).TC1\\
TC1 &\overset{def}{=} (isTrusted\_N, p * r9).TC2\\
&\quad + \ (untrusted\_N, (1-p) * r9).TC4\\
TC2 &\overset{def}{=} (decrypte\_Kvm, r10).TC3\\
TC3 &\overset{def}{=} (sendDecrypted\_Kvm\_To\_N, r11).TC\\
TC4 &\overset{def}{=} (request\_isDenied, r12).TC5\\
TC5 &\overset{def}{=} (end, r13).TC
\end{aligned}
$$

The cooperation between the components of the model is denoted in the system equation, where (Q) represents the number of *User* component instances that cooperate in parallel with *CM* over the set *L*. The cloud manager, *CM*, co-

operates over the set M with the node N, which cooperates with the TC over the set O.

$$System \stackrel{def}{=} (User[Q] \underset{L}{\bowtie} CM[C]) \underset{M}{\bowtie} N \underset{O}{\bowtie} TC$$

Where $L = \{receiveRequest, forward\_N\_id\_To\_User, end\}$ and the $M = \{forwardRequestTo\_N, send\_N\_id\_To\_User\_EncreptedBy\_Kvm, end\}$, then $O = \{sendMessageTo\_TC\_EncrecptedBy\_TKp\_N, sendDecrypted\_Kvm\_To\_N, end\}$. As well, $C = 1$ unless otherwise stated. As there is no current implementation of the TCCP which we could use for measurement, we have made assumptions about the rates of actions which are used. In a full performance study one would obviously need to vary these rates to understand the effect of the assumptions made (i.e. perform a sensitivity analysis). However, due to the limitations of space, we will focus only on a small number of factors in the following evaluation.

4. EXPERIMENTS AND RESULTS

4.1 Varying the Probability of Trust

In this section a number of experiments have been applied to the VM Launch Model, in order to examine the impact of varying the probability of trust on the model behaviour. Probability of trust represents the probability of successful VMs launch requests that have been sent to the *TC*. A selection of the most informative graphs is shown as a sample of the obtained outcomes. As a consequence of the structure of the model and the rates chosen, some different actions will display exactly the same throughput in the model, therefore only the most significant actions are shown, which will make the graphs easier to follow. Figure 2 displays the throughput of the VM Launch model, when the probability of trust is high ($p = 0.8$). The actions throughput are increased steadily in Figure 2 as the number of users increased. Clearly, the subsequent actions that occur after the *isTrusted_N* action will have higher probability to occur when p is higher.

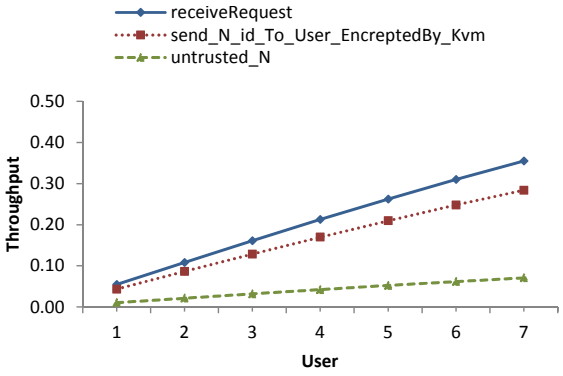


Figure 2: The throughput of the VM Launch Model, where the probability of trust ($p = 0.8$)**.**

Figure 3 shows the population of the VM Launch Model, when the probability of trust is equal (0.2). It can be noticed that, (*CM*, *N* and *TC*) started in an idle state, then decreased as the number of users increased. When the system has one user the rate of which that user interacts with the *CM* is fixed (i.e. It is a coupled system). In addition the population of *User3* increases gradually, as the *useVM* action has a relatively slow rate, which allows to observe the behaviour of the other users, which is saturated at two users in the system. Furthermore, *User2* waiting for a response from the *CM* that make the population of *User2* is raised.

The more users we introduce in the system the more likely that *User1* will have to wait to interact the *CM* (as the users effectively block each other). That is to say, while the probability of *CM* is high, the *CM* is more likely to be idle when the system has one user (obviously) because *User* will spend time doing its independent actions and there are no other users waiting to access the *CM*. Therefore, making the rates of *User* actions faster, or adding more users in the system, will make it less likely that the *CM* is idle. In addition, Figure 3 shows that 70% of the time a single *User* is doing its *use* action (in which case the *CM* is idle) and when the system has two users then the mean time of those two users is 50% of the time exchange and interact.

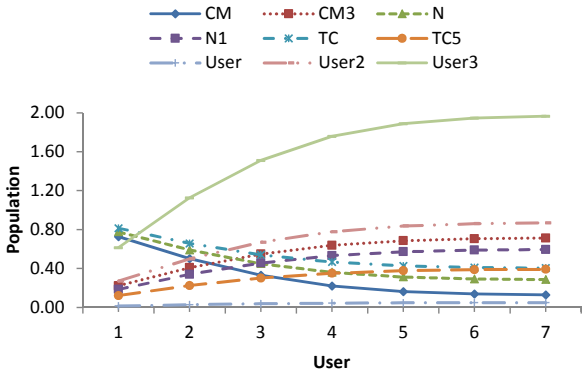


Figure 3: The population of the VM Launch Model, where the probability of trust ($p = 0.2$)**.**

4.2 Scalability of the CM

4.2.1 *Increasing the number of CM instances*

Figure 4 shows the throughput of the VM Launch Model; where the number of *CM* instances $C = 2$ and the probability of trust is 0.8. Besides, the maximum number of users can be derived using CTMC analysis is 5. Figure 4 shows that when $C = 2$, the *CM* can accommodate two users at the same time; then the waiting time is increased when the number of users growing. Although, different scenarios have been used through varying the number of *CM* instances, the results show that there are no any differences in the overall performance. That is because the rate of *CM* is somehow fast while the capacity of the *TC* in the system can only process one request at a time, thus each time there will be one user in use state and the other in the wait state regardless of increasing the number of users. As a result, in this case increasing the *CM* instances will not improve the system performance and increase the cost of using more infrastructures.

Moreover, the ODE analysis has been applied for large number of users in this scenario, where the number of users have been varied from 100 to 1000. However, these results have not shown any kind of variation from increasing the number of *CM* instances ($C = 1$, 2 and 10). Thus it has been decided to exclude these graphs.

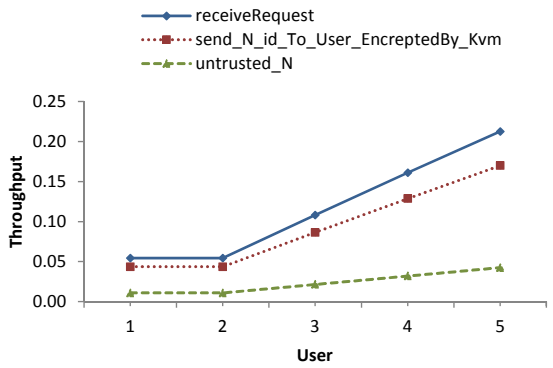


Figure 4: The throughput of the VM Launch Model, where number of *CM* ($C = 2$) and probability of trust ($p = 0.8$).

4.2.2 *Decreasing the rates of CM*

The previous section showed the system performance is not improved by increasing the number of *CM* instances. As a consequence it has been decided to apply a new scenario that allows to scale up the behaviour of the model. To achieve this objective, it is decided to make the rates of both node *N* and *TC* much faster by multiplying them by 10 and makes *CM* rate much slower (i.e. $r5 = 0.06$) and observe how that will affect the overall performance in terms of scalability. The results show only a very small impact on the overall performance of the system.

In summary, the main purpose of these experiments is the scalability of the system; and the point is that by inventing more infrastructure, that will allow scaling the *CM*. One limitation is that, increasing the number of *CM* instances will give rise to a very small improvement in the behaviour of the system which means ineffective and costly choice. Furthermore, as the mechanism of the TCCP security protocol stated that there is only one *TC*, thus it cannot scale the *TC* by replication within this architecture. Therefore, as a future point we are looking at how can can we achieve better scaling through threading in the *TC*, to allow the *TC* to respond to multiple requests simultaneously.

5. CONCLUSION

In this paper we have shown the performance evaluation of the VM launch security protocol within the TCCP system by means of PEPA Eclipse Plug-in tool. To simulate the behaviour of the protocol, a PEPA model has been created. The model and analysis here represent the first stages of an investigation into secure cloud deployment and there are many improvements and extensions which remain to be explored. The motivation of this work is providing a methodology to evaluate the performance of the VM launch security protocol taking into account, providing an accepted level of service while maintaining the confidentiality and the integrity of transmission and processing data. Specifically, investigating methods for the specification of the performance problem and the analysis of performance problem for the VM launch protocol.

Analysis techniques such as CTMC and ODE have been applied, allowing to capture the behaviour of the protocol and deriving measures that appropriate for this type of system. The outcomes of the initial experiments on the model have shown that the *CM* is the initial bottleneck, but replicating the *CM* instances rapidly causes the *TC* to become saturated, causing the throughput of the actions within the *TC* will be at the maximum throughput the model can have. Accordingly, this limitation in the model behaviour which is in the fact, simulated the architecture of the TCCP system as the security protocol centralizes all the processes to be managed by means of the *TC*. Indeed, theses findings agree with the claim made by [11, 9], where they have stated that the *TC* becomes the bottleneck for the reason that all transactions are controlled by the *TC*. As a result, considering a multiple threaded *TC*, will be the next obvious point of research, in order to provide the scalability while maintaining the confidentiality and integrity.

6. REFERENCES

[1] Trusted platform module (TPM). `http://www.trustedcomputinggroup.org/resources/trusted_platform_module_tpm_summary/`, 2015. [Online; accessed 24-Apr-2015].

[2] A. Celesti, M. Fazio, M. Villari, A. Puliafito, and D. Mulfari. Remote and deep attestations to mitigate threats in cloud mash-up services. In *Computer and Information Technology (WCCIT), 2013 World Congress on*, pages 1–6.

[3] T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh. Terra: a virtual machine-based platform for trusted computing. *SIGOPS Oper. Syst. Rev.*, 37(5):193–206, 2003.

[4] K. Hashizume, D. Rosado, E. FernÃąndez-Medina, and E. Fernandez. An analysis of security issues for cloud computing. *Journal of Internet Services and Applications*, 4(1):1–13, 2013.

[5] J. Hillston. *A compositional approach to performance modelling*. Cambridge University Press, Cambridge; New York, 1996.

[6] S. N. S. Kamil and N. Thomas. Performance analysis of the trusted cloud computing platform. In *31st UK Performance Engineering Workshop*. School of Computing, University of Leeds, 2015.

[7] J. Rutkowska. Introducing stealth malware taxonomy. *COSEINC Advanced Malware Labs*, pages 1–9, 2006.

[8] N. Santos, K. P. Gummadi, and R. Rodrigues. Towards trusted cloud computing, 2009.

[9] P. Sen, P. Saha, and S. Khatua. A distributed approach towards trusted cloud computing platform. In *Applications and Innovations in Mobile Computing (AIMoC), 2015*, pages 146–151.

[10] M. Tribastone, A. Duguid, and S. Gilmore. The PEPA eclipse plugin. *SIGMETRICS Perform. Eval. Rev.*, 36(4):28–33, 2009.

[11] H.-z. Wang and L.-s. Huang. An improved trusted cloud computing platform model based on DAA and privacy CA scheme. In *Computer Application and System Modeling (ICCASM), 2010 International Conference on*, volume 13, pages V13–33–V13–39.