Anomaly Detection in Cloud Based Application using System Calls

Marin Aranitasi
Polytechnic University of Tirana, Faculty of Information Technology, Department of Fundamentals of Informatics
Tirana, Albania
Email : maranitasi@fti.edu.al

Mats Neovius
Åbo Akademi University, Faculty of Science and Engineering, Department of Computer Science
Turku, Finland
Email: mneovius@abo.fi

Abstract—**Cloud computing is a rapidly developing computing paradigm. It enables dynamic on-demand resource distribution computing in a cost-effective manner. However, it introduces compelling concerns related to privacy and security of the data. As many of these have been extensively studied and are monitored effectively, this paper proposes a novel solution relying on detecting anomalies in system calls behavior of the system. We use Dempster-Shafer theory of evidence for learning the normality and show how to parametrize this in the method presented. The method is scalable to any set of system calls. Finally, we propose further challenges on this track**.

Keywords- Cloud computing; Security; System calls;.

I. INTRODUCTION

With the advances of (Inter) networking and the advent of the concept cloud computing, computing resources are provided “on-demand” from a virtualized pool. According to NIST [1] essential characteristics of cloud computing include on-demand self-service, network access, pooling, elasticity and measured service. Service models include Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS) with deployment models including private, community, public and hybrid clouds. Common to these are that the hardware is abstracted from the consumer who typically “pay-per-use” of resources whose availability is elastically adjusted by momentary demand [2]. As a consequence, cloud computing encompasses a new means to provide virtually unlimited resources to paying customers whenever needed. Hence, the cloud has revolutionized the way computing infrastructure is used [3] to offer adaptive utility computing.

With novel computing paradigms, concerns on their feasibility arise. In cloud computing this concern relates, among others, to security [4] and privacy. Commonly agreed security requirements include data integrity, confidentiality, access controllability and privacy preservability. These threats do not restrict themselves to point-of-sales activity, but are much more sophisticated extending to data integrity and privacy concerns including espionage, malicious insiders and curious and greedy service providers.These security threats are also the main impediments to wider deployment of cloud computing solutions especiallyin domains operating on sensitive data such as used in the accounting industry [5]. These concerns lend themselves also to emerging cloud-based applications such as the Internet of Things and Big Data [16]. In addition, legislation may add technical impediments for privacy protection such as the Regulation (EU) 2016/679 and EU-U.S. Privacy Shield [6]. The extent to which this affects the Attribute Authority (AA) and the Third Party Auditor (TPA) roles and their functionality is yet to be discovered. Because of these reasons, many solutions are still hosted “in-house”with typically higher initial investment cost, higher maintenance costs and with restrictions on availability.

On the type of attacks on enterprise clouds, Verizon’s security report [7] states that roughly 80% of all breaches are of external origin and 80% have a financial motive, with roughly 20% having espionage as the motive. Moreover, Verizon report that a system is compromised in minutes or seconds and exfiltration of the data happens typically within a day. And despite all the effort by specialists to protect a system from getting compromised, these keep on happening. Research is also directed on this track with searchable and homomorphic encryption research flourishing. Both approaches do, however, rely on a policy and a shared secret lifting the importance of the AA and TPA. Possibly as of this advancements, tactics to perform social attacks granting access to internal attacks develop; with rudimentary known ones including phishing. Other well-known risks the cloud based application faces include data protection risks, system outages including (D)DoS attacks, data loss, vendor lock-in and vendor failure [8].

Common to social and DoS attacks are that no policy-based technological appliance can protect against these. As social attacks typically provide access to restricted data where the attacker would need to know what to search for or how to stir the system up, these are often well targeted with a certain purpose in mind. Alleged examples include Stuxnet, US expelling 35 Russian diplomats at the end of 2016 accused to have tried to influence US presidential elections and the noticed espionage at the Finnish foreign ministry in 2013. With respect to DoS attacks, with the IoT proposal and its envisioned spread, the zombies for botnets are ubiquitous and maintained by ordinary persons lacking maintenance skills. First recorded DDoS attacks with IoT devices are reputed to have occurred in 2015 by CCTV [9] with other larger attacks overloading Twitter, PayPal and Spotify [10] in 2016. Also, if an IoT device operates on private data, its owner's concern is to keep it uncompromised and if compromised, being promptly informed about this. This is forecast as an immense problem with the advances of automated data analysis including image recognition and profiling. Moreover, privacy may be compromised by such a device in a manner enabling identity hijacking.

With respect to these concerns, this paper does not aim to advance on the policies, encryption or similar purely technological advances, but have a main contribution in presenting a method for detecting behavioral anomalies. This method learns a pattern of normality and reacts on events outside this pattern, i.e. anomalies. As an implementation we use system calls of a cloud application, as these are needed whether searchable or homomorphic encryption each time a system accesses the kernel. In this context, we mean by cloud application any cloud based backend software communicating with a set of agents including IoT systems. Thus, compared to policy-based models, this paper takes nearly the reverse view that relies on an agent's past activity to indicate its current activity rather than on Boolean logic and cryptography.

The rest of this paper is organized as follows: Section II present the background and the motivation of the paper. Section III is divided in five subsections and presents our solution to the security issues identified. The first and the second talks about system calls and system call patterns. The other subsections talk about the mathematical method of the proposed solution. Conclusions are presented in section IV.

II. BACKGROUND AND MOTIVATION

In the cloud, main security concerns relate to data privacy or integrity being compromised. Recent infamous ones include Sony PlayStation data breach [11], Dropbox privacy leakage [12] and the alleged major security breach in May 2016 compromising 273M passwords. Typically, the severity of these attacks is measured by the sensitivity of the data being exposed or the harm it causes. It is also speculated that many of the most severe ones do not reach the headlines because of loss of reputation. Moreover, the laws stating the consequences have often not been tested yet, imposing little scrutiny on companies. Moreover, breaches of Service Level Agreements (SLA) dictating the division of responsibilities between the customer and the Cloud Service Provider (CSP) occur frequently. These are seldom made public because of the reputation implications on both parties with an exception of black-hat hackers. Reasons for security breaches may relate to improper configuration, SW bugs, HW errors or power failures [13]. In addition, SW not being up-to-date may contain known exploits. This holds especially for the CSP, where "a single vulnerability or misconfiguration can lead to a compromise across an entire provider's cloud" [14] [17].

With the cloud and its essential characteristics including differences in national legislation, SLA may start to include paragraphs stipulating spatial distribution. This implies that the administrator passwords of the computers used for this application must not have been disclosed to areas not included in the SLA. This concern goes especially for outsourced data services where the owner's exclusive control over their data is compromised when stored on a server whose admin password is known by someone else. For example, Google's privacy policy states that Google reserves the right to review application, project and customer data for compliance with the acceptable use policy [17]. In this case, it comes down to what is "acceptable use policy". Moreover, for personal data, the cloud computing sets a stage of novel problems that need to be dealt with including those who have the right to process the data, what is the level of privacy etc. addressed in e.g. Regulation (EU) 2016/679 and EU-U.S. Privacy Shield [5]. Hence, the cloud used by an application may need to be spatially constrained opposing the principal characteristics of cloud computing.

Means to restrict access and preserve data privacy and integrity includes sophisticated policies on data backup and distribution over nodes. In cases of a cloud application, this is the responsibility of the application service provider and its SLA's with Cloud Service Provider's (CSP). These are often professionally maintained and alternative means to gain access are developed and gaining popularity, e.g. phishing or malwares opening a backdoor or as a key logger. On the CSP, as they share infrastructure, platforms and underlying virtualization software, they form a single point of failure attracting targeted and very sophisticated attacks. If vulnerability is found in any layer, it affects everyone on this cloud. For this, CSA recommend a defense-in-depth strategy. Contemporary attacks are also often more directed and if successful, pose a greater risk.Moreover, common to most attacks are that they often go unnoticed until it is too late, e.g. data exfiltration has already taken place. In such cases, restoring the data from a backup may not suffice as privacy has been compromised.

III. HOW WE MIGHT APPROACH THESE ISSUES

In this paper we take the in-depth approach and present a method that builds the normal behavior of an agent on a cloud system. We construct the normality by analyzing system calls by its user with the aim of detecting system anomalies by monitoring specific system calls of specific applications. This normality would define the way the system works "normally", with any anomaly indicating a situation calling for further attention.

In the next section we present the system calls and we define the set of system calls to use for monitoring and explain the reason why we choose those calls. After that we are going to present the mathematical tool for analyzing and for detecting possible threats in the system.

A. System calls

By definition, a system call is an atomic request in a Unix-like operating system made via a software interrupt by an active process for a service performed by the kernel [16].

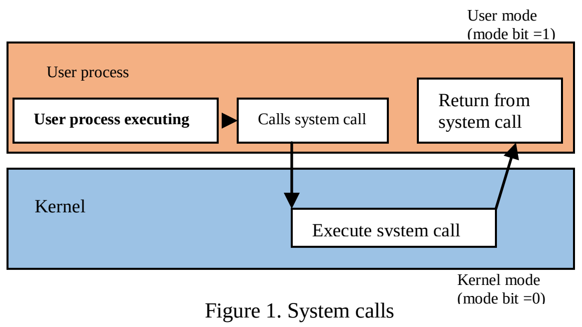

Figure 1. System calls

The system call provides an interface to the operating system's services. Application developers often do not have direct access to the system calls, but can access them through an application programming interface (API). The functions that are included in the API invoke the actual system calls. This is illustrated in Figure 1. By using the API, certain benefits can be gained:

- Portability: as long a system supports an API, any program using that API can compile and run.
- Ease of Use: using the API can be significantly easier than using the actual system call.

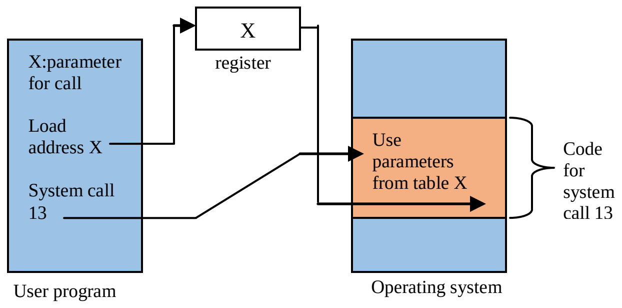

Figure 2. System call parameters

Three general methods exist for passing parameters to the OS as shown in Figure 2:

1. Parameters can be passed in registers.
2. When there are more parameters than registers, parameters can be stored in a block and the block address can be passed as a parameter to a register.
3. Parameters can also be pushed on or popped off the stack by the operating system.

The system calls are plentiful and vary between operating systems, with Linux kernel having 300+ system calls and Windows 7 having close to 700. These can be categorized to 5 different categories: a **process control** is a running program that needs to be able to stop execution either normally or abnormally. When execution is stopped abnormally typically a dump of the memory is taken to be examined by a debugger. The **file management** system calls include create (), delete (), read (), write (), reposition (), or close (). Also, there is a need to determine the file attributes – get and set file attribute. Often the OS provides an API to make these system calls. The **device management** process usually requires several resources to execute, if these resources are available, they will be granted and control returned to the user process. These resources are also thought of as devices. Some are physical, such as a video card, and others are logical, such as a file.User programs *request* the device, and when finished they *release* the device. Similar to files, we can *read*, *write*, and *reposition* the device. The **information management** system call exists purely for transferring information between the user program and the operating system. An example of this is *time*, or *date*. The OS also keeps information about all its processes and provides system calls to report this information. The **communication**system call exists in two models of interprocess communication, the message-passing model and the shared memory model.

- Message-passing uses a common mailbox to pass messages between processes.
- Shared memory use certain system calls to create and gain access to regions of memory owned by other processes. The two processes exchange information by reading and writing in the shared data.

According to the characteristics of the system calls we propose to monitor 2 categories of system calls:

1. Communication
2. File management.

For the communication category we propose to monitor accept (), socket (), connect () system calls, and for the file management category we propose to monitor read (), write (),delete () andcreate () system calls.These are the system calls that rank as the most common threats in the CSA report and are vital for any cloud based application. We think that monitoring the data sent across the network is not a good idea because there is a high overhead tracing those system calls and they do a lot variable invocation for sending and receiving data. Hence, this might not be favorable to do with the method presented below without packet inspection.

B. System call patterns

We assume the system calls monitored to behave in anatomic manner and the set of them to be exclusive and exhaustive. That is, we assume the system calls not to be subject to race conditions hence assuming an atomic part to be executed from the beginning till the end and each and every one is exactly one of the possible. On such a foundation a pattern could be constructed from analyzing the system call log, i.e. by the recorded evidence. Moreover, the pattern could be augmented by contextual bindings by some machine learning method. The outcome could reasonably be a probability of a certain or a sequence of system calls happening. Applying a timed window on this analysis would provide a timed pattern for the system calls, e.g. a diurnal pattern when human behavior is analyzed.

To detect anomalies, a valid approach is to teach a model what normality is by analyzing the past. Yet, the model must consider the possibility of a change in the system or its behavior implying a change in normality. Realistically, this could mean a software update or installation of new software. Hence, the valid system call pattern calls for an adaptive method providing a level of certainty that the system indeed operating normally. In case of anomaly the system could inform the user about the task behaving anomalously prompting the user to authorize the anomaly.

C. Mathematical foundation of the proposed method

On the problem and domain outlined in this paper, we propose to use Dempster-Shafer theory, aka, evidence theory. The evidence theory is a generalization of Bayesian theory of subjective probabilities on a set of exclusive and exhaustive events X, here the system calls. The power set 2^X denotes all combinations of system calls, realistically enabling comparing any category of system calls to discover new domain specific patterns. On this, the mass m is the level of certainty on a set of events with $m : 2^X \rightarrow [0,1]$, $m(\emptyset) = 0$ and $\sum_{2^X} m = 1$. On this, the certainty (belief)bel of a set of outcomes $A \subseteq X$ is $bel(A) = \sum_{x \subseteq A} m(x)$ and plausibility pl is $pl(A) = \sum_{x \cap A \neq \emptyset} m(x)$ as for the possibility of this outcome. This implies that $bel < pl$ whenever $m(X) \neq 0$ and $A \subset X$. The semantics of this is that the difference between *bel* and *pl* denote the uncertainty. Moreover, the complement of a set of events A denoted $\bar{A}$ is the evidence against this event, i.e. $pl(A) = 1 - bel(\bar{A})$. Consequently, the Dempster-Shafer theory

Realistically, in the context of this paper, the *bel* and *pl* would define the uncertainty, i.e. the tolerance between normal (base truth) and anomaly behaviour that initially is 1. The theory provides a foundation for a three-valued logic, whose parameters are: belief as certainty in favour of a proposition bel, uncertainty as for do not know $pl - bel$ and disbelief $\overline{bel}$ as for certainty against this proposition $1 - pl(A)$. They share the property of $pl(A) + \overline{bel} = 1$, i.e. they are additive. In cases when $bel(A) = pl(A)$, the uncertainty is 0 and the theory behaves as traditional probability theory.

D. The adaptive method

Having the Dempster-Shafer theory as a solid foundation with a plethora of extensions that enable calculation with it, the problem of defining the values for the parameters is central. On this, inspired by Krukow's [18] and Teacy et al. [19], Neovius et al. [20, 21, 22] have in previous work presented a method for recording and mapping experiences to Dempster-Shafer theory. They consider an event an experience that in the context of this paper is a system call. Hence, let the set of system calls S and the communication $C = \{accept, socket, connect\}$ and file management $F = \{read, write, delete, create\}$ categories be exclusive and exhaustive, i.e. $S = C \cup F$ and $C \cap F = \emptyset$ and similarly for the elements.

With these system calls, we model an experience as a four tuple $(\delta, \epsilon, \zeta, \eta)$ where δ is the subject system's and application's identification, ϵ the timestamp, ζ the set of system call and η a score $\in \{0, 1\}$. This view can be reduced to that only events that actually took place are recorded, i.e. that the score is always 1. Thus, an experience $(\delta, unixtime, open, 1)$ indicates that on a device and *app* δ at a time called the *open* system call and this was triggered. For an entity, the history of the device's system calls can be modelled as a set of such four tuples, i.e. $\{(\delta, \epsilon, \zeta, \eta)\}$. Projections on this history $Exp(\delta, \epsilon, \{read\}) = \{(\eta)\}$ in case the cardinality *card* of the result indicate the amount of connect system calls that were made at time ϵ. Realistically, δ could be IMEI code augmented by an application, say FB including its version.

With the realistic assumption that recent behaviour weighs heavier, we may apply a decay function on this. Let decay be denoted by λ where $\lambda \in [0,1]$ with semantics of the closer to 1 indicating less decay and 0 being a vacuous view. Then decay at ϵ_m denoted d_{ϵ_m} is defined: $d_{\epsilon_m}\big(Exp(\delta, \epsilon_i, s \subset S)\big) = (\lambda^{\epsilon_m - \epsilon_i} * \eta)$. A cyclic (diurnal) history is an abstracted view of this projection with λ = 1 with ϵ_m denoting the moment and $\epsilon_n \leq \epsilon_m$ the timespan, i.e. $Abs_{\epsilon_m}\big(Exp(\delta, \epsilon_n, s \subset S)\big)$is the abstract score $\sum_{d_{\epsilon_m} Exp(\delta, \epsilon_n, s \subset S)} \eta$. That is, for a comparison view over a 2 hour time span yesterday ϵ_m is set -23hours and ϵ_n to -25hours from this moment. The definition of such views are defined by some contextual predicate constructed by a domain specialist; a fundamental question omitted in this paper.

E. Detecting anomalies with the method

Utilizing the history of events and building a decayed view of the cyclic behavior on each system call provides a basis for normality. For comparison and anomaly detection, the cardinality needs to be put in context. Hence, a projection on the complementary experiences within this category of system calls is motivated. Thus, having $Exp(\delta, \epsilon, \{read\})$, the category's complementary projection is $Exp(\delta, \epsilon, \{write, delete, create\})$, i.e. $Exp\big(\delta, \epsilon, \overline{\{read\}}\big)$. The cardinality of the outcomes provides the relative distribution of these system calls over ϵ on δ. The tolerance is then defined as the a priori weight of uncertainty *W*. The scale of *W* is domain δ and category specific with the property small values risking anomalies with low values; and larger W prolonging the cold start.

As an example, assume the projections over a time span where the system calls cardinality is 95 and the score for projection on $read$ to be 73 and for that on $\overline{read}$ to be 22. However, let the abstracted projections result in 70 and 21 respectively as of decay.Moreover, let for readability $W = 5$, making the example specific cardinality 100. Normalizing these gives the scores 0.70, 0.21 with uncertainty $u$being $0.04 + \frac{W}{cardinality\ +W} = 0.09$. Consider a reference vector $bel(\vec{r}) = (x_i)$ that in this case is $bel(\vec{r}) = (0.7, 0.21)$; much alike the belief and certainty for the two projections. The plausibility of these projections are then $pl(\vec{r}) = (0.79, 0.3)$. With these abstracted values, we propose to define an anomaly behavior as when the current *bel* and *pl* does not overlap with the reference *bel* and *pl* vectors. What actions to perform if this happens is again domain specific.

IV. Conclusion

Cloud computing can lead to numerous business advantages to organizations. As a result of its popularity, many security issues have been exposed by company experts and academic researchers.Numerous of these researches haveproved that security should be a top priority for companies, especially low- to medium-sized enterprises ones. Moreover, the common ground is that security related solutions developed for static client server systems cannot be used in cloud based computing.

In this paper we take a novel view, assuming that a system under attack will behave anomalously. To address this assumption, we presented a soft security means to construct a cloud based solutions' behavioral normality. Knowing the normal behavior, we define the anomalous behavior to be simply anything that is not normal. We stress that the implications of detecting anomalies is domain specific.

As future work, we intend to validate this method with real life data. We will also formalize the normality vs. anomalies more formally. Once having these results, validation on a larger scale is possible.

References

[1] US national institute of standards and technology.http://csrc.nist.gov/, 2016, accessed 15.1.2017

[2] D. Fernandes, L. Soares, J. Gomes, M. Freire, and P. Inacio, "Security issues in cloud environments: aSurvey". International Journal of Information Security, pp. 113–170, 2013

[3] F. Shahzad, "State-of-the-art Survey onCloud Computing Security Challenges, Approaches and Solutions". *Procedia Computer Science*, *37*, 357–362, 2014

[4] I. Iankoulova and M. Daneva, "Cloud computing security requirements: A systematic review". In: *Research Challenges in Information Science RICS,* pp. 1–7, 2012

[5] R. Duncan, "Accounting for stewardship in the cloud" PhD Thesis, University of Aberdeen, 2016.

[6] European Comission. "Comission Implementing Decision of 12.7.2016 pursuant to Directive 95/46/EC of the European Parliament and of the Council on the adequacy of the protection provided by the EU -U.S. Privacy Shield" 2016 http://ec.europa.eu/justice/data-protection/files/privacy-shield-adequacy-decision_en.pdf accessed on 13.01.2017

[7] Verizon "Data Breach Investigations Report" 2016 http://www.verizonenterprise.com/verizon-insights-lab/dbir/2016/ accessed on 13.01.2017

[8] M. Williams, "New Tools for Business,A Quick Start Guide to Cloud Computing", Kogan Page, 2010.

[9] S. Khandelwal, "Hacking CCTV Cameras to Launch DDoS Attacks".The Hacker News, 2015, http://thehackernews.com/2015/10/cctv-camera-hacking.html, accessed on 15.1.2017.

[10] T. Reuters, "Major cyberattack knocks Twitter, Paypal, Spotify offline Friday" CBC news, 2016,http://www.cbc.ca/news/technology/dyn-ddos-attack-websites-down-1.3815417, accesed on 13.01.2017

[11] E. Petterson, "Sony to pay as much as $8 million to settle data breach case", Bloomberg Technology, 2015https://www.bloomberg.com/news/articles/2015-10-20/sony-to-pay-as-much-as-8-million-to-settle-data-breach-claimsaccesed on 13.01.2017

[12] S. Yin, "Dropbox accounts were accessible by anyone fo four hours on Sunday" PCMag UK, 2011.http://uk.pcmag.com/storage-devices-reviews/9092/news/dropbox-accounts-were-accessible-by-anyone-for-four-hours-onaccesed on 13.01.2017

[13] R. Ko and S. Lee "Cloud computing vulnerability incidents: A statisctical overview", Cloud Security Alliance, 2013.

[14] F. Rashid, "The dirty dozen: 12 cloud security threats", Infoworld magazine, 2016, http://www.infoworld.com/article/3041078/security/the-dirty-dozen-12-cloud-security-threats.html, accesed on 13.01.2017.

[15] Linux Information Project "System call definition" 2016 http://www.linfo.org/system_call.html accessed on 13.01.2017

[16] S. De Capitani, S. Foresti, and P. Samarati, "Data Security Issues in Cloud Scenarios", In Proceedings of the 11th International Conference on Information Systems Security, 2015.

[17] Cloud security alliance, "Cloud Computing top threats in 2016".https://downloads.cloudsecurityalliance.org/assets/research/top-threats/Treacherous-12_Cloud-Computing_Top-Threats.pdfaccesed on 13.01.2017

[18] K. Krukow, "Towards a theory of trust for the global ubiquitous computer," PhD thesis, University of Aarhus, Denmark., 2006.

[19] W. Teacy, J. Patel, N. Jennings, and M. Luck, "TRAVOS: Trust and Reputation in the Context of Inaccurate Information Sources," Autonomous Agents and Multi-Agent Systems, vol. 12, no. 2, pp. 183-198. , 2006.

[20] M. Neovius, "Trustworthy Context Dependency in Ubiquitous Systems," TUCS dissertations nr. 151. PhD thesis, Turku, Finland, 2012.

[21] M. Neovius, M. Stocker, M. Rönkkö, and L. Petre, "Trustworthiness Modelling on Continuous Environmental Measurement," in Proc. of the 7th Int. Conf. on Environmental Modelling and Software, 2014.

[22] M. Neovius, "Adaptive Experience-Based Composition of Continuously Changing Quality of Context," in Int. Conf. on Adaptive and Self-Adaptive Systems and Applications, 2015.