Deriving a Distributed Cloud Proxy Architecture for Managed Cloud Service Consumption

Dirk Thatmann, Mathias Slawik, Sebastian Zickau, and Axel Küpper
Service-centric Networking, Department of Telecommunication Systems
Technische Universität Berlin and Telekom Innovation Laboratories
Berlin, Germany
{*d.thatmann|mathias.slawik|sebastian.zickau|axel.kuepper*}*@tu-berlin.de*

***Abstract*—Businesses adopting Cloud Computing often have to comply with strict constraints, such as enterprise policies and legal regulations. From these compliance issues arise the need to enable managed cloud service consumption as a prerequisite for adoption. As we have shown before, the proposed *TRusted Ecosystem for Standardized and Open cloud-based Resources (TRESOR)* cloud ecosystem can achieve management of cloud service consumption [1]. In this paper we motivate and derive the architecture of the distributed *TRESOR* cloud proxy from technical, business and legal requirements within the context of the *TRESOR* project. We apply a derivation method where we evaluate the impact of each incremental architecture decision separately. This process enables researchers with supplementary requirements to adapt the intermediate derivations within other contexts in flexible ways.**

***Keywords*-Cloud Computing, Cloud Ecosystem, Distributed Cloud Proxy, Architecture, Cloud Management, REST**

I. Introduction

Adapting cloud services can be difficult for certain business sectors. As introduced by Thatmann et al. [1] a scheme for overcoming the loss of management capabilities, legal compliance uncertainties and integration limitations do exist. Hindered by the high demand for data privacy, monitoring, data storing, SLA and compliance to complex regulations by law, the health care industry is barely taking advantage of cloud services. A goal of the *TRusted Ecosystem for Standardized and Open cloud-based Resources (TRESOR)* cloud ecosystem is to regain management capabilities in order to lower the entrance barrier for sensitive sectors to use cloud computing.

The *TRESOR* cloud ecosystem provides a modern, secure, and legal compliant way in consuming and trading cloud services and focuses especially on sensitive sectors, such as the health care industry while minimizing lock-in effects. The cloud ecosystem consists of different components: the cloud broker, the cloud proxy, and an open PaaS platform. However, in this paper we focus on the derivation of the cloud proxy architecture.

The role of the cloud proxy is to link cloud service customers and cloud services in a secure and legal compliant way. This includes real time communication monitoring, logging and access control.

In the following chapters we motivate the cloud proxy architecture through an iterative step-by-step procedure based on principle design constraints, considerations of the state of the art technologies and concepts and previously identified requirements. Section II reflects the motivation for the cloud proxy, identifies important requirements imposed on it, introduces its basic design principles, and describes the Trusted Cloud Transfer Protocol (TCTP). Section III presents the development of the cloud proxy step-by-step, while applying the aforementioned design principles and requirements. In Section IV the previously derived architecture is evaluated based on a real world scenario. Section V concludes the paper.

II. Motivation

Solutions, which address arising shortcomings, need to be found for an ever-changing industry sector, such as the IT sector on the verge of an always-online world. The disadvantages and risks of using cloud computing in sensitive business environments collected in [1] led to proposed solutions in order to address these limitations. This preliminary work forms the basis for specifying basic design principles and identifying additional requirements of a cloud proxy architecture. One primary goal of this work is to achieve new fundamental possibilities in cloud computing service consumption. To motivate the overall solution we describe the basic design choices and five fundamental requirements which form the basis for the presented approach.

Today, many modern cloud services follow RESTful design principles [2], [3], [4] and leverage from its simplicity (e.g., for monitoring RESTful requests) or performance advantages [5]. Based on these findings, we define our basic design principles. These principles are:

1.) Supporting modern HTTP-based, RESTful web technologies.
2.) Addressing management capabilities by following RESTful principles.
2.1) Separating the HTTP header and body, thus handling management and confidential data separately.
2.2) Following common principles of cloud computing, such as SOA and utility computing.

The proxy shall control compliance to SLAs and contracts and monitor and audit all communication during run-time. Furthermore, the end-user authentication and authorization shall be done with support of already existing services on customers premises. A high level role model is depicted in Figure 1.

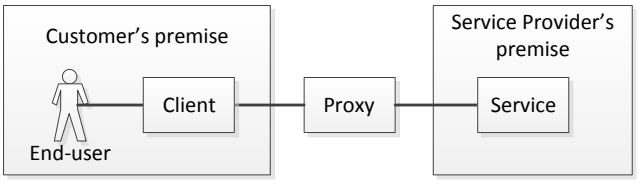

Figure 1. High level role model

Taking a simple service consumption as the basis in a provider/consumer environment, the notion of trust and management capabilities arise, if the provider and consumer are different parties. With just two parties involved it would be a stalemate situation if it comes to a disagreement. An independent evaluation of the service consumption which might monitor the promised SLAs is one solution that can increase the trust level. Any company has certain rules which need to be manageable from the consumer side, e.g. the possibility to add or remove users in a system or to grant certain access rights to service consumers. These are notions in the area of authentication and authorization. In an environment with two independent parties there is the need of privacy and security of data storage and service consumption because the consumer side does not want to reveal any confidential information about its business or partners. In a service consumption environment, in this case a cloud computing ecosystem, interoperability of all components in a heterogeneous environment is a critical factor of success. The following five requirements sum up the motivation:

A. Cloud Proxy Requirements

The cloud proxy requirements, which have to be considered in every derivation step separately, are stated next:

1) Management: Companies shall have the opportunity to define and maintain specific access rules for employees including already existing role models, in order to control access to cloud services (e.g. SaaS) offered by 3rd parties over the Internet. This requires a policy decision point. Configuration options for compliance to enterprise policies and legal regulations shall exist.

2) Independent Monitoring: Service Level Agreements (SLAs) have to be monitored and audited independently from the cloud service consumers and providers. This increases the quality and the management capabilities of SLAs, which is especially important for the acceptance of cloud services in sensitive sectors such as the health care sector.

3) Privacy: Only the customer and the assigned data processing cloud service should have access to sensitive utility data. Precondition can be a legal contract for commissioned data processing.

4) Authentication & Authorization integration: From a customers perspective, already existing authentication and authorization systems shall be reused in order to reduce costs arising from additional systems and software while adopting cloud services.

5) Interoperability: From an interoperability perspective, web browsers, without any plugins, shall become the common denominator in cloud consumption scenarios. This reduces lock-in effects and allows common desktop and even mobile web browsers to access cloud services. However, this does not exclude other applications to be integrated and used for accessing cloud services.

B. Trusted Cloud Transfer Protocol

As introduced in [1], the Trusted Cloud Transfer Protocol (TCTP) enables a 3rd party HTTP proxy entity to monitor and control HTTP compliant communication. Therefore, HTTP headers and RESTful URIs can be accessed, but sensitive and secret data, contained in the HTTP body, cannot. To achieve this goal, sender and receiver have to adopt to TCTP encryption scheme. In contrast to classic one-to-one communication (sender / receiver), a 3rd independent party is involved in the TCTP-based communication.
TCTP behaves as follows: At first, a connection from sender A to proxy B and from proxy B to receiver C, each secured independently by TLS, is established. This lets the 3rd party become a intermediate link. Second, the sender A encrypts the HTTP body separately, using a different symmetric encryption key, received by a second TLS handshake with the receiver C. Afterwards the HTTP message (header and body) is send through the proxy B to the receiver C. The TCTP stack, compared to HTTPS, is shown in Figure 2.

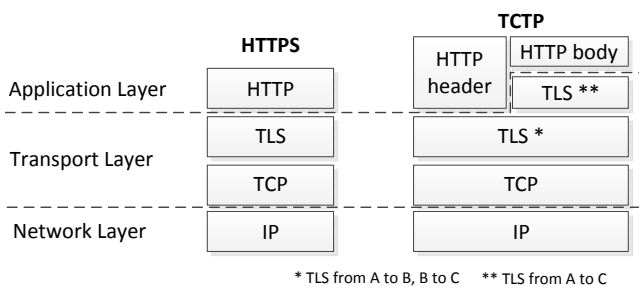

Figure 2. Stack comparison: HTTPS and TCTP

III. Deriving The Cloud Proxy Architecture

In this section the proxy architecture is derived from previously defined general design principles and derived requirements, introduced in chapter II-A.

A. Zero State: Cloud Usage Today

From a customer's perspective most cloud services can be accessed by web browsers. Hereby, authentication mechanisms are used to identify the customer. This often requires to type in credentials manually or to increase QoE the integration of web technologies like OpenID and OAuth for a Single Sign On (SSO). Especially SSO solutions, widely used in enterprise domains, such as Microsoft Active Directory, Kerberos or LDAP, are not designed to interoperate by design with OpenID and OAuth used in the Internet. A shortcoming that can be mentioned is Kerberos' strict requirement to synchronized clocks of involved hosts, which is hardly achieved in non-managed networks such as the Internet, hence *Authentication & Authorization* as defined is not met. Integrated *Management* functionality is not generally given. Customers have to adopt existing role models and access rules manually and individually to each 3rd party cloud service.

In terms of *independent monitoring*, many problems on different layers exist [6] [7]. Amazon provides its own service CloudWatch [8] for its own resources, hence it not independent. Independent 3rd party services, able to monitor different cloud providers (such as Gomez [9]), try to fill this gap. However, independent monitoring is not available by default, therefore cloud service customer have to take care of monitoring themselves.

Privacy is met, since a direct and end-to-end encrypted communication between customer and cloud service provider (e.g. SaaS) is supported by HTTPS.

Interoperability is met, given that common web browsers allow access to common web services out of the box. However, accessing cloud services built with technologies such as Adobe Flash, Microsoft Silverlight or Java FX, always require in general a run-time provided as a plugin.

Under consideration of requirements presented in section II-A, Table I presents an overview of supported requirements for a common cloud service usage of today. A high level architecture is depicted in Figure 3.

No	Name of requirement	met
1	Management	-
2	Independent Monitoring	-
3	Privacy	✓
4	A&A Integration	-
5	Interoperability	✓

Table I
Requirements and Cloud usage today

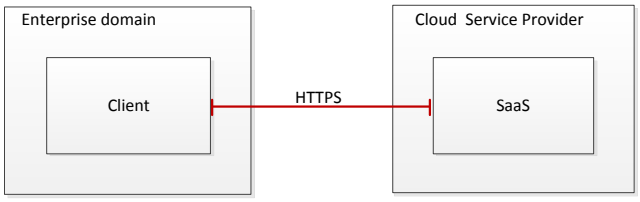

Figure 3. Abstract cloud service consumption today

B. 1st Step: The Local Proxy

In order to address shortcomings on customer's premises, a local proxy is introduced. This HTTP proxy can meet *management* requirements, since it is located in the network - the domain of the customer. Furthermore, existing authentication and authorization systems can be integrated.

SLA monitoring can be realized, but *independent monitoring* is impossible, as the proxy is controlled by the customer and located on the customer's premises.

Privacy and *interoperability* requirements are supported, since a secure channel can be setup from the proxy on the customer's premises to the cloud service provider's domain (e.g. SaaS). A browser can still be used as an application client, given that the proxy integration and the authentication and authorization can be realized by the customer itself and, by design, in a transparent way. An overview of supported requirements is displayed in Table II, while a high level architecture is depicted in Figure 4.

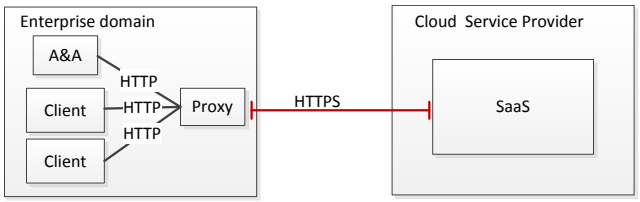

Figure 4. Local Proxy (on customer's premise)

No	Name of requirement	met
1	Management	✓
2	Independent Monitoring	-
3	Privacy	✓
4	A&A Integration	✓
5	Interoperability	✓

Table II
Overview: local proxy

C. 2nd Step: 3rd Party Proxy and Monitoring

Since *independent monitoring* is not supported by the previously introduced local proxy (chapter III-B), a proxy is now shifted to an independent 3rd party. This will enable *independent monitoring*, but will in addition lead to several drawbacks.

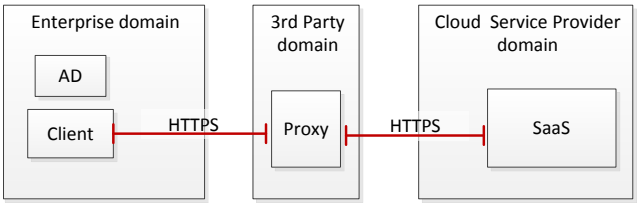

Figure 5. trusted 3rd Party Proxy

As depicted in Figure 5, a secure end-to-end HTTP channel based on TLS, from the customer's domain to the SaaS offering, provided by a cloud service provider, cannot be supported anymore. All encrypted HTTPS connections must terminate at the 3rd party proxy in order to support *independent monitoring*. Otherwise *monitoring* becomes impossible, since the proxy must access management information located in each HTTP header. In conclusion, the *privacy* requirement is violated, since a 3rd party has full access to each HTTP messages passing by. Next, meeting the requirement *A&A Integration* is difficult. Credentials cannot be reused in a simple manner when authenticating and authorizing employees towards an 3rd party proxy, located outside of a company's premises. However, credential delegation is not impossible. SPNEGO [10] allows Kerberos tickets to be transported by HTTP, firewalls can open ports to allow communication with kerberized services (e.g., the 3rd party proxy) out of premises. Precondition for combining multiple domains is supported by Kerberos [11]. However, this leads to unwanted Kerberos related lock-in effect for both: customers and 3rd party proxy. At last, the shift does not affect *interoperability* requirements, since plain web browsers can still be used when accessing cloud services.

An overview of the current supported requirements is presented in Table III.

No	Name of requirement	met
1	Management	✓
2	Independent Monitoring	✓
3	Privacy	-
4	A&A Integration	-
5	Interoperability	✓

Table III
OVERVIEW: 3RD PARTY PROXY, MONITORING ENABLED

D. 3rd Step: 3rd Party Proxy and Privacy

To come around the lack of *privacy*, as described in section III-C, a secure end-to-end connection between customers and cloud service providers, must exist. As shown in Figure 6, all HTTPS requests, initiated by the customer, terminate at the cloud service provider's premises. This will regain compliance to *privacy* requirements, since a secure end-to-end connection can be established again, but will remove support for *management* and *independent monitoring*. *A&A Intergation* support and *Interoperability* capabilities remain unchanged (compare chapter III-C). An overview of now supported requirements is presented in Table IV.

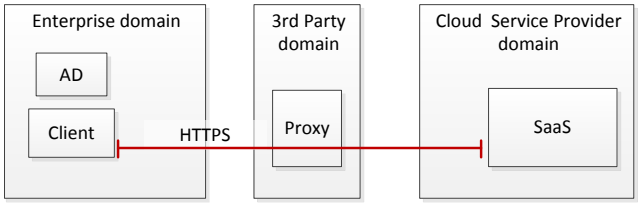

Figure 6. trusted 3rd Party Proxy and TLS passthrough

No	Name of requirement	met
1	Management	-
2	Independent Monitoring	-
3	Privacy	✓
4	A&A Integration	-
5	Interoperability	✓

Table IV
OVERVIEW REQUIREMENTS: 3RD PARTY PROXY + 2ND. TLS OPTION

E. 4th step: 3rd Party Proxy and TCTP

Concluding the results achieved in Chapter III-C and III-D, it stands to reason that a combination of both derivation steps will lead to full support of *Management*, *Independent Monitoring* and *Privacy* aspects, as stipulated in Section II-A.

As introduced by Thatmann et al. [1], we propose TCTP as described in Chapter II-B. The use of TCTP in combination with the aforementioned derivation steps is depicted in Figure 7. As a result, the current architecture fulfills *management*, *independent monitoring* and *privacy* requirements but still lacks *A&A integration*.

The requirement *transparency* is not met, since TCTP is based on a double TLS encryption, which must be initiated in a different way compared to a common TLS connection. All applications (e.g. browsers, SaaS) have to be adopted to TCTP, which must be avoided. Table V presents the overview of currently supported requirements.

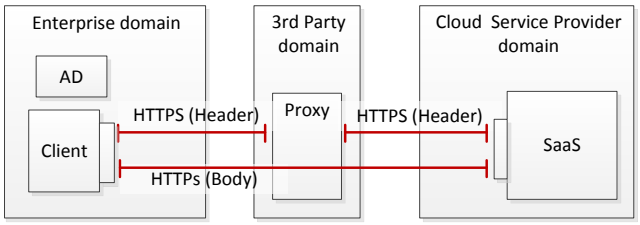

Figure 7. TCTP: separate TLS encryption for HTTP header and body

F. 5th step: Distributed Proxy and TCTP

In addition to the 3rd party proxy and its interaction with TCTP, as describe in the previous Chapter VI, limitations

No	Name of requirement	met
1	Management	✓
2	Independent Monitoring	✓
3	Privacy	✓
4	A&A Integration	-
5	Interoperability	-

Table V
OVERVIEW REQUIREMENTS: 3RD PARTY PROXY + TCTP

related to *A&A Integration* can be eliminated by reintroducing the local proxy, as described in Chapter III-B. Defining the local proxy as a starting point for TCTP communication will relieve all client applications (e.g., browsers) from the need to adopt TCTP, thus meeting the *interoperability* requirement. Transferring this approach to the premises of the cloud service provider, will relieve services (e.g. SaaS) from adapting TCTP, too.

Since all requirements are met, as summed up in Table VI, the final architecture, the distributed cloud proxy for the *TRESOR* cloud ecosystem is created and displayed in Figure 8.

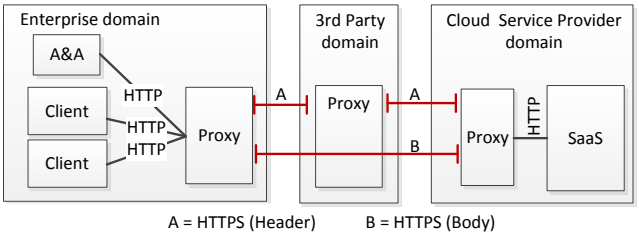

Figure 8. distributed trusted 3rd Party Proxy using TCTP

No	Name of requirement	met
1	Management	✓
2	Independent Monitoring	✓
3	Privacy	✓
4	A&A Integration	✓
5	Interoperability	✓

Table VI
OVERVIEW REQUIREMENTS: DISTRIBUTED 3RD PARTY PROXY + TCTP

IV. PRELIMINARY EVALUATION

The *TRESOR* distributed cloud proxy is the first real-world implementation of the derived cloud proxy architecture. In this section the impact of the implementation and its interconnection to other components of the *TRESOR* trusted cloud ecosystem is evaluated. As the *TRESOR* cloud ecosystem is not yet fully enacted, part of this evaluation is based on the *TRESOR* proxy proof-of-concept and the current specification status of the *TRESOR* components.

A. Meeting Architecture Requirements

The *TRESOR* distributed cloud proxy is a trusted mediator between cloud consumers and cloud providers. It is a vital infrastructure element of the proposed *TRESOR* cloud ecosystem, which is designed to regain capabilities for managing the consumption of cloud services. The *TRESOR* proxy is distributed between client proxies, the one central *TRESOR* proxy and proxies operated by the service providers. All proxies share one code base, which currently consists of a proof-of-concept implementation using the Java Grizzly NIO framework [12] to implement a flexible HTTP reverse proxy. The next subsections detail the impact of the implementation based on the five main requirements set for the cloud proxy architecture.

B. Management

The distributed *TRESOR* cloud proxy meets the architecture requirement *management* as it contains manifold mechanisms for controlling the consumption of cloud services: First, the *TRESOR* cloud proxy will include a flexible policy decision point, which enables enterprises to define fine grained access rules for their booked *TRESOR* cloud services. These policies will be evaluated by the client proxy, so that an illegal request will never be sent to the central and service proxy.

Second, the *TRESOR* ecosystem will include location aware features, which makes it possible for enterprises to reincorporate the access restrictions of their enterprise premises in the cloud and also allows the policy decision point to enable location-aware computing functionality within cloud services.

C. Independent Monitoring

Within the *TRESOR* ecosystem, the central part of the distributed cloud proxy will be operated by a trusted 3rd party. This party is independent from the cloud service consumers and providers. As one of the responsibilities of this trusted 3rd party is independently monitoring the service level agreements between consumers and providers, the architecture requirement *independent monitoring* is met within the *TRESOR* ecosystem. These agreements will not only include common terms, such as service availability, but also incorporate enhanced cloud computing SLAs, such as mean time to repair (MTTR) or mean time between failure (MTBF), as proposed by Emeakaroha et al. [13], [14] and partially incorporated by Dobson et al. [15].

D. Privacy

Privacy is one of the main concerns of the health care institutions involved with the *TRESOR* project. Using the Trusted Cloud Transfer Protocol for securing the communication between its distributed components, the *TRESOR* proxy is able to manage the consumption of sensitive cloud solutions without compromizing the privacy and security of the processed patient data.

E. Authentication & Authorization

A proposed *TRESOR* proxy plugin mechanism lets enterprises reuse their existing authentication and authorization systems. The *TRESOR* proxy proof-of-concept implementation allows different authentication and authorization schemes to be realized. Current work focuses on using existing Java frameworks, such as Apache Shiro [16], to permit the *TRESOR* proxy to use functions provided by Kerberos [11] based solutions, such as Microsoft Active Directory [17].

F. Interoperability

As the *TRESOR* distributed cloud proxy acts as a reverse HTTP proxy, user agents (e.g. browsers or client software) and servers do not have to be modified in order to participate in *TRESOR*. Modifications on the server side are only then necessary, when services use extended functionality of the ecosystem, such as pricing cloud resource usage based on metering information conveyed in API calls to a *TRESOR* billing component. Implementation effort is also reduced, as applications do not have to implement specific proprietary APIs, but can instead rely on HTTP.

V. Conclusion

This paper motivates and presents the derivation of the distributed *TRESOR* cloud proxy incrementally, based on basic design principles and previously identified requirements [1], for regaining management capabilities within the *TRESOR* Cloud Ecosystem.

The proxy enables an end-to-end encrypted connection through a trusted 3rd party proxy from a cloud service customer to a specific cloud service. This connection ensures privacy and security compliance, since sensitive utility data is not accessible in the 3rd party proxy, while the proxy can fulfill all management related tasks, such as monitor a specific communication and even enforce corresponding SLAs.

From a cloud service customer's perspective, the distributed proxy requires installation on customer's premises. In addition, customers will be able to manage access to cloud services for their employees easily, while reusing existing authentication and authorization systems (e.g., Kerberos, LDAP and Microsoft Active Directory).

From a cloud service provider's perspective, existing RESTful services can be adapted in a simple manner, since the service side proxy, which has to be installed on the provider's premises, takes care of the TCTP connection.

Acknowledgment

The work presented in this paper was performed in the context of the *TRESOR* project. *TRESOR* is funded by the German Federal Ministry of Economics and Technology "Bundesministerium für Wirtschaft und Technologie (BMWi)".

References

[1] D. Thatmann, M. Slawik, S. Zickau, and A. Küpper, "Towards a Federated Cloud Ecosystem: Enabling Managed Cloud Service Consumption." in *Proceedings of the 9th Intl. Conference on Economics of Grids, Clouds, Systems, and Services (GECON 2012).* Springer, 2012.

[2] R. T. Fielding and R. N. Taylor, "Principled design of the modern Web architecture," *ACM Trans. Internet Technol.*, vol. 2, no. 2, pp. 115–150, May 2002. [Online]. Available: http://doi.acm.org/10.1145/514183.514185

[3] C. Pautasso, O. Zimmermann, and F. Leymann, "Restful web services vs. "big"' web services: making the right architectural decision," in *Proceedings of the 17th international conference on World Wide Web*, ser. WWW '08. New York, NY, USA: ACM, 2008, pp. 805–814. [Online]. Available: http://doi.acm.org/10.1145/1367497.1367606

[4] C. Pautasso and E. Wilde, "RESTful web services: principles, patterns, emerging technologies," in *Proceedings of the 19th international conference on World wide web*, ser. WWW '10. New York, NY, USA: ACM, 2010, pp. 1359–1360. [Online]. Available: http://doi.acm.org/10.1145/1772690.1772929

[5] Potti, Ahuja, Umapathy, and Prodanoff, "Comparing Performance of Web Serice Interaction Styles: SOAP vs. REST," in *Proceedings of the Conference on Information Systems Applied Research*, 2012.

[6] S. Meng and L. Liu, "Enhanced Monitoring-as-a-Service for Effective Cloud Management," *Computers, IEEE Transactions on*, vol. PP, no. 99, p. 14, 2012.

[7] P. Leitner, C. Inzinger, W. Hummer, B. Satzger, and S. Dustdar, "Application-level performance monitoring of cloud services based on the complex event processing paradigm," in *Service-Oriented Computing and Applications (SOCA), 2012 5th IEEE International Conference on*, dec. 2012, pp. 1 –8.

[8] Amazon, "CloudWatch," http://aws.amazon.com/cloudwatch/, 2012.

[9] Compuware, "Compuware Gomez - SaaS," http://www.compuware.com/application-performance-management/gomez-apm-products.html/.

[10] K. Jaganathan, L. Zhu, and J. Brezak, "SPNEGO-based Kerberos and NTLM HTTP Authentication in Microsoft Windows," RFC 4559 (Informational), Internet Engineering Task Force, Jun. 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4559.txt

[11] C. Neuman, T. Yu, S. Hartman, and K. Raeburn, "The Kerberos Network Authentication Service (V5)," RFC 4120 (Proposed Standard), Internet Engineering Task Force, Jul. 2005, updated by RFCs 4537, 5021, 5896, 6111, 6112, 6113, 6649. [Online]. Available: http://www.ietf.org/rfc/rfc4120.txt

[12] java.net, "index.html - Java.net," http://grizzly.java.net/, 2012.

[13] V. C. Emeakaroha, I. Brandic, M. Maurer, and S. Dustdar, "Low level Metrics to High level SLAs - LoM2HiS framework: Bridging the gap between monitored metrics and SLA parameters in cloud environments," in *High Performance Computing and Simulation (HPCS), 2010 International Conference on*, 28 2010-july 2 2010, pp. 48 –54.

[14] V. Emeakaroha, T. Ferreto, M. Netto, I. Brandic, and C. De Rose, "CASViD: Application Level Monitoring for SLA Violation Detection in Clouds," in *Computer Software and Applications Conference (COMPSAC), 2012 IEEE 36th Annual*, july 2012, pp. 499 –508.

[15] G. Dobson and A. Sanchez-Macian, "Towards unified QoS/SLA ontologies," in *Proceedings of Third International Workshop on Semantic and Dynamic Web Processes (SDWP 2006)*, 2006.

[16] Apache Software Foundation, "Apache Shiro," http://shiro.apache.org/, 2013.

[17] Microsoft Corporation, "About Active Directory Domain Services (Windows)," http://msdn.microsoft.com/en-us/library/windows/desktop/aa772142.aspx, 2012.