Virtualised Trusted Computing Platform for Adaptive Security Enforcement of Web Services Interactions

Ivan Djordjevic[1*], Srijith K. Nair[2*†], Theo Dimitrakos[1]
[1] *SOA Security, Security Research Centre, British Telecommunications, UK*
[2] *Department of Computer Science, Vrije Universiteit, Amsterdam, The Netherlands*
{ivan.djordjevic, theo.dimitrakos}@bt.com, srijith@few.vu.nl

Abstract

Security enforcement framework is an important aspect of any distributed system. With new requirements imposed by SOA-based business models, adaptive security enforcement on the application level becomes even more important.
Our work on the enforcement framework to date has resulted in a comprehensive middleware-based solution leveraging on web services technologies. However, potential merits of hardware-based solutions to further secure application exposure have not been considered so far.
This paper describes a method for combining software resource level security features offered by Web Services technologies, with the hardware-based security mechanisms offered by Trusted Computing Platform and system virtualisation approaches. In particular, we propose trust-based architecture for protecting the enforcement middleware deployed at the policy enforcement endpoints of web and grid services. The main motivation is to additionally secure execution environment of the applications, by providing virtual machine level separation that maps from logical domains imposed by web services level enforcement policies.

1. Introduction

1.1 The Context

With the adoption of Service Oriented Architectures (SOA), a new use of a term "virtualisation" is appearing. It is being used in the context of service virtualisation, to describe an advanced way of cross-enterprise integration of application services and virtualisation of the (cross-organisational) computational environment where these services are hosted and executed. We call this a Virtual Hosting Environment (VHE) middleware capability.
Virtualisation of Hosting Environments refers to the federation of a set of distributed hosting environments for execution of an application and the possibility to provide a single (logical) access point to this set of federated hosting environments. In addition to the application services, this virtualised service bundle needs to include a number of middleware services (potentially provided by a third party) for managing non-functional aspects of the application. From the perspective of a VHE consumer, the latter are transparent. VHE as such requires two main things − trust federation and security enforcement.
The former deals with the establishment of the trust relationships among multiple parties and its management over the lifecycle of the collaboration, including mechanisms for specifying and adapting the agreements between the parties that define contracts of service interactions. The basis of the preferred federation model used in our work is described in [1].
The latter (security enforcement) deals with monitoring and enforcement of these contracts during the service interactions and execution. For the preferred security enforcement model used in our work see [2].

1.2 The Motivation

In this paper we extend the security enforcement design proposed in [2] in order to extend the integrity, reliability and availability of the web services enforcement system by leveraging on trusted computing technology as described in [3] and [5].
The architectural extension builds on the concepts of *virtualization* and *process isolation / compartmentalization* in the presence of a trusted computing hardware. While these concepts are not new, commercial-grade hardware that provides support

[*] Editing authors
[†] Work carried out during the author's scientific visit at BT Security Research Centre.

2007 IEEE International Conference on Web Services (ICWS 2007)
0-7695-2924-0/07 $25.00 © 2007 IEEE

for the technology has only recently been introduced into the market making their use economically viable.
The main aim is to design a web service architecture that can provide secure isolation between several distinct exposures of a (web) service of the same type - which share the same physical hardware resources, but are being used in different interaction contexts. The three important security problems addressed in this paper are:

1. Securing the configuration file of each "instance" of the web service;
2. Securing the data of each service instance;
3. Securing the memory used by the service instance.

The approach taken in addressing the above concerns is to use the principle of secure containment, with the appropriate support of a Trusted Platform Module (TPM) [3] -like secure hardware. The underlying idea is that each instance of a web service is run inside an isolated partition that has no access to any resources (including memory and data handles) used by another similarly isolated compartment. The trusted hardware base such as the TPM is logically emulated within each compartment and can hence be assumed to be dedicated to each partition for all practical purposes.

2. Web Services Security Enforcement Framework

The enforcement middleware is a system for adaptive, extendable policy-based message level enforcement for heterogeneous computational resources exposed as network services and distributed over a network.
The full set of requirements that address management, adaptation and functional aspects of the enforcement middleware, as well as the detailed description of the system is presented in [2].
The enforcement middleware provides a composable system that intercepts each message that is targeted at or is originating from a network service endpoint. It dynamically deploys a collection of message interceptors in a chain (interceptor chain) through which the message is processed prior to transmission. The interceptor chain is formed per intercepted message based on the content and protocol context of the intercepted message, and constraints derived from the configuration policies of the enforcement middleware.
The same enforcement middleware can be used to protect a number of resources offered by different providers. Further, some aspects of the enforcement middleware configuration are specific to each resource being protected, whereas others can be common the enforcement middleware instance. The management of the enforcement middleware is decoupled from the Enforcement Point itself, allowing several disparate administrators or management services to have ownership for managing only certain specific aspects of the enforcement middleware.

2.1 Policy Framework Structure

The behaviour and configuration of the enforcement middleware is based on the data contained within a number of supporting policies, which can be of four types:

- Interceptor Reference Policy (IRP) contains mapping between each available enforcement action and the computational entity (i.e. the interceptor) that executes this action.
- Enforcement Configuration Policy (ECP) specifies the enforcement state, the actions that are to be enacted, the conditions under which each action is executed, the parameters for each action and the sequencing of the actions. The action types in the ECP are a subset of the action types included in the IRP policy.
- Utility Service Policy (USP). While processing the message some of the interceptors may require the use of capabilities of external services (e.g. access to a key-store or a policy authorisation service). All information regarding the alternative services available and the locations of these services are contained in the Utility Service Policy.
- Capability Exposure Policy (CEP) is used for publishing additional conditions for interacting with a protected resource. The network entities interacting with a protected resource are obliged to provide the required data in addition to any other application-specific data requested by the provider of the application service.

ECP policies are deliberately separated from IRPs in order to abstract away implementation specific detail. For example, an ECP policy can state that encryption of the SOAP message body is executed but there are different algorithms that can be applied to do so. In order to dynamically deploy the appropriate interceptor in the chain that will correctly execute enforcement actions the content of the message has to be analysed and the actual encryption scheme will need to be identified from the intercepted message. Once the relevant enforcement actions and their configuration parameters and order in which they will form the chain have been identified, the IRP is loaded and inspected in order to determine the references to the interceptor implementing each enforcement action.

2.2 Enforcement Process

The enforcement middleware defines different groups of interceptors that fall into three generic types: verification, transformation, and for invoking common infrastructure capabilities.
Once a message is intercepted, a dispatcher analyses the contents of the message, as this may convey information required to determine which ECP policy to use. For SOAP messages, ECP is typically determined based on the reference endpoint of the targeted service and optionally any other related information from the message header.
Based on this contextual information and the state of the enforcement middleware, the enforcement middleware determines suitable ECP policy for processing the message. Once the appropriate interceptors have been identified and deployed in the interceptor chain, the message is pushed through the chain and each interceptor executes the corresponding processing / enforcement action in a sequence (see Figure 1).
After the message is processed, the interceptor chain is dissolved in anticipation of another message being intercepted. After the processing has completed, the message is dispatched to the recipient.

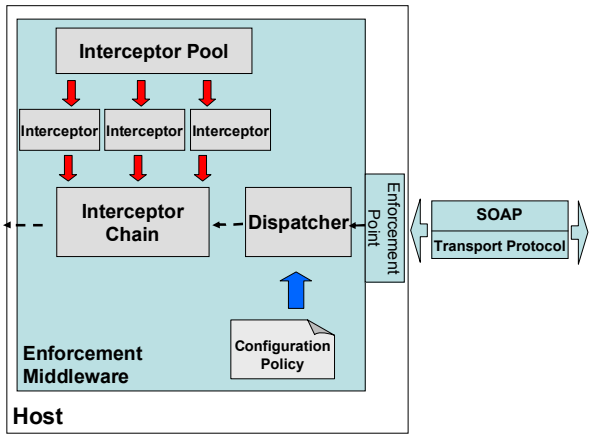

Figure 1: Enforcement process overview

2.3 Deployment architectures

There are several alternatives in the deployment of the enforcement middleware: at the same host (machine) as the service it is protecting, as a network intermediary, or in a distributed fashion (by aggregating functionality of a set of enforcement components distributed over a network). In the context of this paper, we focus on the first alternative.

3. Virtualisation and Trusted Computing

Virtualisation and trusted computing technologies provide system architectures and tools that allow building of inherent security and trust mechanisms in grid and web services hosting environment.
Virtualisation provides mechanisms to create partitions that share the hardware resources but are isolated in its working. This helps in improving the security of the processes, as well as making the process distribution efficient and optimises the use of limited resources. Trusted computing allows cryptographic keys and application data to be stored securely within the machine hardware and provide mechanisms to check the integrity of a remote machine.
In this section we give a brief summary of these technologies.

3.1 Virtualisation

The concept of a virtual machine was first developed by IBM in order to provide concurrent access to the mainframe resources [6]. Each virtual machine (VM) provided a completely protected and isolated abstraction of the underlying hardware architecture to the applications running inside it. In the recent years, hardware virtualization has become a very popular technology due to the fact that sharing of hardware among multiple workloads reduces operating costs as well as makes the system utilization more efficient [7].
In general, the process of virtualization allows the creation and maintenance of multiple virtualization layers providing each of them with an isolated and partitioned access to the underlying hardware resources like CPU, memory, input and output channels, and so on. Virtual Machine Monitor (VMM) is the software layer that provides this virtualization layer and supports its creation, maintenance and teardown. Commonly, the VMM layer is called the host and the individual VMs are termed the guests. Figure 2 demonstrates the relationships between the VMM and the VMs.
Different levels of virtualisation are possible, like the instruction set architecture level virtualisation implemented by QEMU [8] and others that provide full instruction set level virtualisation,, to hardware abstraction level virtualization as implemented by Xen [9], which provides virtualisation at the hardware abstraction layer. Detailed explanation of these implementations are beyond the scope of this paper and interested readers are referred to [10],[11].
The increased interest in the field of application and web service technologies has made the hosting providers of such services look for system architectures

to increase their system efficiency and consolidate their hardware resources. Virtualisation has proved to be a perfect fit for this requirement. Several recent research works have been investigating on how virtualization can be extended to support the *on demand* nature of web service hosting requirements. SODA [12] is one such architecture that virtualizes each service nodes by running it within individual VMs on the hosted machine. By designing their architecture in a 'Master-Agent' setup they create the needed services on demand, across several machines in the hosting farm.

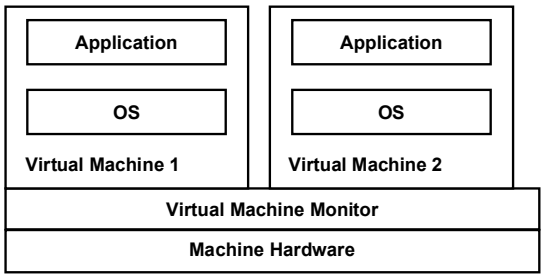

Figure 2: Virtual Machine Architecture

Up to date, works in the area of virtualisation provide only basic process isolation and system security that is inherent in the virtualisation paradigm. The additional security requirements for hosting web and application services have not been addressed so far. In this paper we use the security features of Trusted Computing to address these requirements.

3.2 Trusted Computing

Trusted computing aims to provide open commodity systems with certain desirable properties, usually associated with high-assurance closed systems. For example, such trusted platforms, while allowing applications from different sources to run on the same physical platform, allow third parties to determine which software are running on the system. This provides a degree of trust about the expected behaviour of these systems.

The Trusted Platform Module (TPM) specifications [3], defined by the Trusted Computing Group [13], provide a mechanism to implement such a trusted computing architecture by using (among other things), a hardware root of trust. The TPM is implemented as a chip that is attached to the motherboard of the machine. It provides several cryptographic operations, such as random number generation, asymmetric and symmetric key encryption and decryption, signing, secure hashing, etc. Each TPM has several cryptographic keys built in.

Storage Root Key (SRK) forms the Root of Trust for Storage and always resides in the non-volatile memory of the TPM. When a TPM generates a key, it is generated by its parent key and SRK forms the root of this tree. Endorsement Key (EK) is used to uniquely identify the TPM. Each TPM manufacturer provides a certificate to the EK attesting the compliance of the TPM to the specifications. The TPM produces Attestation Identification keys (AIKs) that are linked to the platform using certificates from the EK. Certification Authorities (CAs) uses the certificate issued by the EK and the manufacturer's certificate of EK to attest the AIKs.

These TPM chips are being increasingly deployed in consumer level laptops / desktops, as well as industrial level machines like the IBM eServer x366 [14].

Each TPM has at least 16 Platform Configuration Registers (PCRs) that store measurement values (usually hash values) of platform configurations. The PCR values along with the AIKs can be used to attest the state of a machine using the process of remote attestation [15].

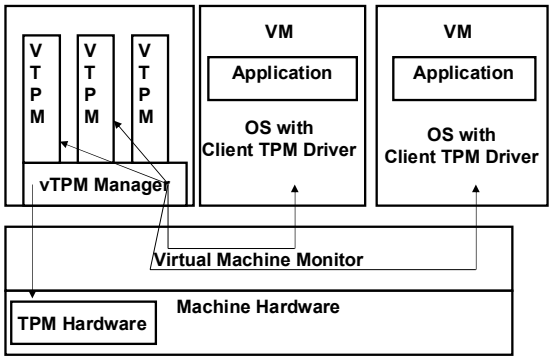

Figure 3: vTPM Architecture

Just like any other hardware, the TPM needs to be virtualized in order to be used within a VM setup. IBM's work on vTPMs [5] is an excellent starting point in this front (see Figure 3). The hardware TPM is controlled by a vTPM Manager that resides in one of the VMs. It also creates other vTPM *instances* that are then associated with individual VMs. Each vTPM instance performs the full set of TCG TPM specifications, thus allowing each VM to us the vTPM instances as if the VM had a direct control over the physical TPM chip.

By generating an EK per vTPM, this architecture allows each vTPM, and hence each VM that uses the instance, to decrypt information using the private key associated with the EK. It also enables the creation of independent key hierarchy per vTPM.

By using the trusted computing architecture for deploying web services, one can not only increase the security of the system (keeping the encryption key secure in the TPM), but can also support the possible requirement of consumers to verify the integrity of the deployed system (using remote attestation).

4. Candidate Architectures for Virtualisation of Enforcement Middleware

As described earlier in section 2.2, before a SOAP message is delivered to a web service instance, it is passed through a number of Policy Enforcement Points (PEPs), which adapt their configuration at the runtime based on the context surrounding the interaction and the ECP policy.
When virtualising the web services framework, careful consideration has to be given to the process of virtualising the enforcement framework, in order to preserve the semantics of the ECP policy at the VM level.

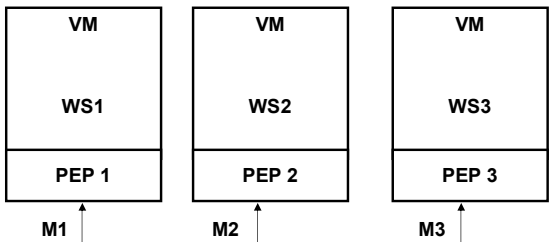

Figure 4: Shared PEP/service partition model

One straightforward way of doing this is shown in Figure 4. In this architecture each PEP is virtualised along with the service it protects into separate VMs. It may seem that such a process isolation would increase the system security, However, since the web service (along with all the components that constitute it) share the same VM partition as the enforcement framework, they will also share the same memory pages as the enforcement framework. Hence a malicious component could potentially attack the enforcement framework through malicious buffer overflows or similar mechanisms. In addition, this model would require replication of the interceptor repository across each partition in order to provide at runtime a secure non-shared instance of every legal interceptor combination (that corresponds to any applicable action) within that VM. In turn, this increases complexity of the management and configuration of the security enforcement system.
An alternative approach that is a better match for the PEP middleware is the architecture shown in Figure 5.

It contains a dedicated VM partition that holds all the interceptors in the repository, as well as the individual ECPs for each web service instance hosted on that machine. Effectively, there would be a single PEP deployment with different PEP configurations depending on the targeted web service (instance).
It is clear that the repository containing the different PEP configurations need not be replicated across the individual partitions. However, this setup would require that resources like physical memory be shared between action chains that operate on different SOAP messages directed for different web service instance. We do not consider this being an issue, since it is possible to use a shared resource environment at the PEP level without affecting the security and integrity of the application part of the SOAP messages, which contains the payload for the WS instance.

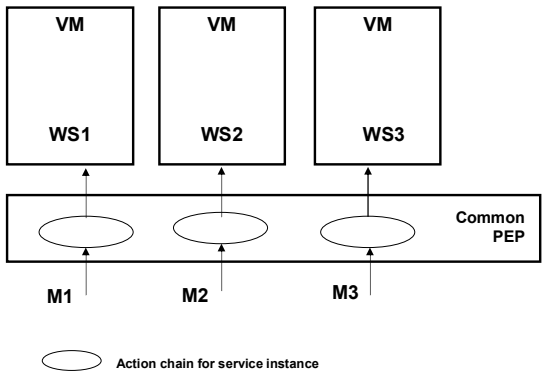

Figure 5: Dedicated PEP partition model

Since the SOAP message consist of a number of XML elements, the transformation/ manipulation of the SOAP messages allows for different processing of different parts of the message, as prescribed by the SOAP specification [17]. Furthermore, WS-Security [18] prescribes that arbitrary parts of the message can be selectively protected (i.e. encrypted/ signed).
Therefore, the portion containing the application data (typically the body of the message) can be encrypted with a key that is accessible only by the virtualized TPM of the service instance partition. If required, the headers that need to be accessed by the common PEP partition can then be encrypted, for a key that is available within the TPM of this common PEP partition.
Next section describes the partitioning architecture that builds on the preferred model of the dedicated PEP as proposed in Figure 5. Even though this model could have potential scalability issues when the number of WS VMs becomes very large, we expect it to be advantageous compared to the model proposed in the

Figure 4 that would require interceptor repository to be replicated across every VM. In addition, the model presented in Figure 5 provides better separation of the management of application and infrastructure functionalities. Another reason for choosing the dedicated PEP partition model is that it closely resembles the vTPM architecture model, making it easier to add trusted computing support into the system, as explained in the next section.

5. Proposed Partitioning Architecture

Keeping in mind the design criteria discussed in the previous section, we propose the partitioning architecture shown in Figure 6 to support virtualised web and grid services deployment. The architecture preserves all the properties of the security enforcement middleware framework (i.e. PEP), while enabling independent TPM support for each of the web service instances and the dedicated PEP.

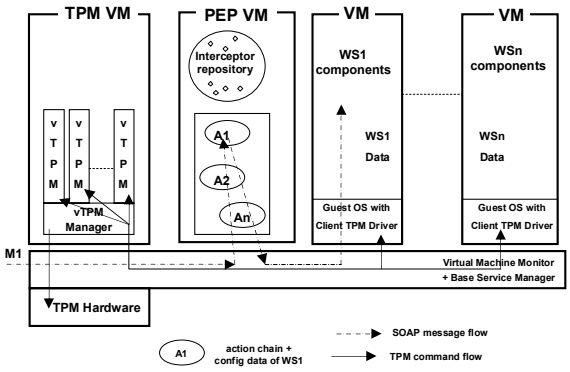

Figure 6: Proposed "TCP-PEP" partition architecture for WS instances

The Base Service Manager (BSM) is a trusted piece of software that manages the creation, maintenance and teardown of WS instances and is logically located at the same level as the type2 VMM[3]. When the hosting environment is initialised for the first time, the BSM instructs the VMM to create a dedicated VM partition for the PEP, and populates it with the interceptor repository.

The PEP partition is used in order to store any general (i.e. service-instance-independent) configuration data and components. In particular, the interceptor pool of the enforcement middleware, the interceptor reference policy, and the utility service policy are stored there. The run-time engine of the enforcement middleware can access the data and components in this partition. Also authorized security infrastructure administrators may obtain access to information in this partition by using the programmatic interfaces as explained in [2].

At the time of the web service deployment, an administrator provides the application files and the Enforcement Configuration Policy for the web service. When requested by the administrator to instantiate a web service, the BSM first instructs the VMM to create a new secure VM partition with a dedicated vTPM. This provides a secure execution environment for the web service instance. The BSM then instructs the PEP to create a 'base' configuration file for the service, which combines external address of the service instance (i.e. EPR – end point reference), VM identifier of the partition where the instance is hosted, and the reference to the appropriate Enforcement Configuration Policy. This allows the PEP to apply the correct policy to the incoming messages at runtime whilst channelling them to/from the appropriate VM. Then the WS instance is created in the newly virtualised partition.

The ECP data as well as all general purpose configuration policies, i.e. interceptor reference policy (IRP) and the utility service policy (USP) of the enforcement middleware are saved within the PEP partition while all the data (including service instance state) that is received or created by the service instance is isolated within the WS's VM partition.

Once the WS instance partition has been created with its own virtual TPM, the vTPM creates relevant key pairs (EK, SRK, AIKs etc.) for the WS instance. The public key of the EK pair is then advertised as the public key of the WS instance. Any data sent for the WS instance can then be encrypted with this public key; an additional, "signature" key-pair can be created[4]. In general, an arbitrary number of the "crypto" material, secret(s) etc. can be created for various purposes and protected by the vTPM keys.

The AIKs, together with the PCR values of the vTPM, can be used by the web service customers to perform remote attestation of the VM's state. This is particularly useful for security auditing and traceability of the creation and management of system partitions and TPM virtualisations.

When a SOAP message is intercepted, it is sent to the PEP partition where the endpoint address (EPR) and other declared identification meta-data is used for deciding which enforcement configuration policy (ECP) to apply as described in [2]. The PEP uses the

[3] VMM runs within level 2 protection ring (OS) with guest OS running at level 3 (applications)

[4] In reality, this will yield more complex scheme due to the fact that web services would use XML – style assertions to request or provide authentication information. However, the corresponding encryption / signature EK keypairs can be used to secure this information at the vTPM.

information specified in the ECP to collect the appropriate handlers and to create the handler chain through which the message is then passed.

Once the PEP has performed the necessary enforcement and checks at its level, the SOAP message is passed into the BSM, which uses the EPR data in the message to route the message to the appropriate WS instance partition. Since the data for the WS instance can be decrypted only with a key encrypted with the partition's vTPM specific key, it can be guaranteed that the data is not accessible outside the service partition.

The web service instance however needs to have access to read, as well as other (management) services to edit its configuration file contained within the PEP partition. Since an uncontrolled cross-partition data access process should not be allowed, a standard interface is defined that allows the WS partition to read its configuration file. Since the WS partition is not allowed to read the file directly, the configuration changes have to be first conveyed to the PEP partition through the defined interface via the BSM. Using the AIK of the WS partition's vTPM to sign the request is enough to prove the authenticity of the request and access the configuration file via an authorised process located in the PEP partition. Note that editing and modification of the configuration file should not be allowed by the WS itself. This is the responsibility of the management services that are normally remotely hosted, so this would typically be a web services call to the management interfaces of the PEP (see section 2 and reference [2] for more information on the PEP manageability).

The hosting platform provider considers the VMM and the TPM VM trusted, and the WS components (including the interceptor chains) as non-trusted. An attack targeted against a specific WS component is confined within the WS VM and does not affect any other WS instance on the same physical machine nor the core trusted part of the hosting platform. Similarly, since the interceptor chains reside within a separate PEP VM, a malicious or buggy interceptor provider would not endanger the WS operation and data.

The vTPM's secret key used to encrypt data in the WS VM provides confidentiality. The remote attestation functionality of the TPM (and vTPM), which allows a remote user to verify the status of the hosting environment, provides integrity. The use of VM architecture allows for the migration of the WS instance during its execution (should it be required), therefore contributing to the availability of the architecture.

6. Related Work

Although there is significant existing work on the subject of virtualisation, its application in the web and grid service framework is less studied. SODA [12] is one of the few systems that use virtualisation technology to create on-demand VM partitions to improve resource utilisation. However, they do not consider the policy enforcement framework and security in general in their design. Daonity [16] uses Trusted Computing technologies to bolster the trust model of grid service framework by enhancing the security of credential and membership management. However they too do not consider the implementation of the policy enforcement framework nor about the security and integrity of web service components.

IBM's vTPM [l] project proposes and implements architecture to virtualise the TPM hardware across VMs. Their work also provides mechanism to migrate the vTPM along with the rest of the VM in an efficient manner. However they do not consider the use of vTPM within a web or grid service infrastructure. The architecture proposed in this paper uses and also draws inspiration from the vTPM design.

7. Conclusion

In this paper we propose an architecture for virtualising the policy enforcement framework of web services. By incorporating the functionalities provided by the trusted computing technologies with the isolation properties of virtual machines partitions, our architecture enhances the security and integrity of web service infrastructure, in particular the crucial functionality of the dynamic policy enforcement framework.

We consider two alternative architecture designs and choose the one that can more efficiently support our requirements for secure provisioning of multiple web services in a single hosting environment. The key idea is to separate policy enforcement from application at the virtual machine level, so that applications cannot compromise security mechanisms.

In addition, the architecture provides separation of the management of application and infrastructure functionalities. This provides a baseline for the secure service hosting and operation in federated environments, where providers of the applications (ASPs), hosting environments, and security services may be different entities, hence requiring separation of control. For example, host provider may wish to limit maximum number of the partitions in order to deliver committed quality of service, ASPs to be able to upgrade their application functionality without having

to deal with managing its protection, and the security service providers to enforce the application security based on the higher level agreements established with the ASPs[5].

The system environment assumed in the architecture builds on top of the "TCG Generic Server Specification" proposal [4] of the Trusted Computing Group, which increases the confidence that it can be assumed a realistic production environment.

Next stage in this work is experimental implementation of the proposed architecture for securing interactions on the WS-based B2B Gateway described in [21].

Acknowledgments

Some aspects of the work reported in this paper are funded by EU IST integrated projects TrustCoM [22] and BEinGRID [23].

References

[1] Djordjevic I., Dimitrakos T., Romano N., Mac Randall D., Ritrovato P.: Dynamic Security Perimeters for Inter-Enterprise Service Integration. Future Generation Computer Systems, the International Journal of Grid Computing: Theory, Methods and Applications, Elsevier B.V. 2007, Volume 23, Issue 4, Pages 633-657 (May 2007). URL: http://www.sciencedirect.com/science/journal/0167739X

[2] Maierhofer A., Dimitrakos T., Titkov L., Brossard, D.: Extendable and Adaptive Message-Level Security Enforcement Framework, In proceedings of the International conference on Networking and Services, (ICNS '06), Paolo Alto, July 2006, ISBN: 0-7695-2622-5

[3] "TCG Specification Architecture Overview", Trusted Computing Group, Revision 1.2, April 2004, https://www.trustedcomputinggroup.org/

[4] "TCG Generic Server Specification", v 1.0 revision 0.8, Trusted Computing Group, May 2005, https://www.trustedcomputinggroup.org/

[5] Berger S., Caceres R., Goldman K.A., Perez R., Sailer R., van Doorn L.: "vTPM: Virtualizing the Trusted Platform Module", IBM Research Report RC23879, February 2006.

[6] Creasy R.J.: The Origin of the VM/370 Time-Sharing System, IBM Journal of Research and Development, 25(5):483, 1981.

[7] Figueiredo R., Dinda P.A., Fortes J.: Resource Virtualization Renaissance, IEEE Computer Magazine, 38(5):28-31, 2005.

[8] QEMU Macine Emulator and Virtualizer, http://fabrice.bellard.free.fr/

[9] Xen Virtual Machine Monitor, http://www.cl.cam.ac.uk/research/srg/netos/xen/

[10] Goldberg R.P.: Survey of Virtual Machine Research, IEEE Computer Magazine, 7(6):34-45, 1974.

[11] Nanda S., Chiueh T.: A Survey of Virtualization Technologies, Research Proficiency Report, Stony Brook, ECSL-TR-179, February 2005.

[12] Jiang X., Xu D.: SODA: A Service-On-Demand Architecture for Application Service Hosting Utility Platforms, In proceedings of 12th IEEE International Symposium on High Performance Distributed Computing, pp. 174, 2003.

[13] Trusted Computing Group, https://www.trustedcomputinggroup.org/

[14] IBM eServer x366, http://www-03.ibm.com/servers/eserver/xseries/x366.html

[15] Sailer R., Zhang X., Jaeger T., van Doorn L.: Design and Implementation of a TCG-based Integrity Measurement Architecture, In proceedings of the USENIX Seceurity Symposium, 2004.

[16] Mao W., Yan F., Chen C.: Daonity: Grid Security With Behaviour Conformity from Trusted Computing, In proceedings of the First ACM Workshop on Scalable Trusted Computing, pp. 43-46, ISBN:1-59593-548-7, 2006.

[17] SOAP Version 1.2 Part 1: Messaging Framework, W3C Recommendation, M. Gudgin, M. Hadley, N. Mendelsohn, J-J. Moreau, H. Nielsen, 24 June 2003; see http://www.w3.org/TR/2003/REC-soap12-part1-20030624/

[18] OASIS Web services security TC. WS-Security 1.0 http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf

[19] Dimitrakos T., Titkov L., Maierhofer A.: Message Processing Methods and Systems: a method and system for adaptive SOAP-level security enforcement. PCT application No. GB2006/004088; to be published September 2007

[20] Leiva R.G., Rodríguez I.H., Álvaro D.M., Warren P., Djordjevic I., Dimitrakos T., Romano N., Biette M.: A Grid Computing for Online Games. In proceedings of Game Design and Technology Workshop 2006 (GDTW'06), Liverpool, UK, November 2006.

[21] Dimitrakos T.: Securing application service exposure & integration in B2B collaborations. In business track of ECOWS 2006, the 4th IEEE European Conference on Web Services, Zurich, December 2006.

[22] TrustCoM project website: www.eu-trustcom.com

[23] BEinGRID project website: www.beingrid.eu

[5] Middleware architecture that can benefit out of this concept is being developed in the context of the BEinGRID project [20].