Trusted Virtual Domains on OpenSolaris: Usable Secure Desktop Environments

Hans Löhr, Thomas Pöppelmann, Johannes Rave, Martin Steegmanns, Marcel Winandy
Horst Görtz Institute for IT Security
Ruhr-University Bochum, Germany
{hans.loehr, marcel.winandy}@trust.rub.de
{thomas.poeppelmann, johannes.rave, martin.steegmanns}@rub.de

ABSTRACT

Trusted Virtual Domains (TVDs) are a security concept to create separated domains over virtual and physical platforms. Since most existing TVD implementations focus on servers and data centers, there are only few efforts on secure desktop environments. To fill this gap, we present in this paper an implementation of TVDs based on the OpenSolaris operating system. We leverage several of its existing features (e.g., lightweight virtualization, security labels and a secure graphical user interface) and extend OpenSolaris with components for automated management and policy enforcement to create a usable desktop implementation of TVDs. This includes the transparent encryption of external storage and home directories of users, restriction of copy-and-paste according to the TVD policy, efficient deployment of images for user environments, and a central management interface for the administration. Our system enables organizations to securely separate different work flows with sensitive data from each other and from untrusted environments.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*Security and Protection*; D.4.6 [**Operating Systems**]: Security and Protection

General Terms

Security

Keywords

TVD, secure desktop, management, OpenSolaris

1. INTRODUCTION

The concept of Trusted Virtual Domains (TVD) [12, 4, 5] allows the enforcement of a common security policy on a coalition of machines that trust each other based on that policy. This concept originates from the problem of separating security-sensitive workloads in data centers using shared hardware resources. TVDs provide a framework that enforces the needed separation transparent to the instances of different domains on this shared hardware. To support a highly dynamic virtualized infrastructure, the TVD policy is defined and managed in an abstract way centrally, whereas each platform enforces the central policy locally.

Besides data centers, the TVD approach can be extended to the desktop environment in an enterprise scenario. This enables to control and restrict the information flow for end-user systems, e.g., to enforce complex enterprise rights management policies [11] or to simply separate data of different workflows with varying security requirements. For example, in a hospital one domain may be privacy-sensitive patient records and another one for accounting and billing. In order to separate these different data domains, i.e., to avoid unauthorized accesses from one to the other domain, each individual logical task can be migrated into a single TVD which enforces protection of the data, including encryption of data stored externally or sent over the network.

While the incorporation of the TVD concept in data centers [3, 5] can be expected to be available in the near future, only few research prototypes address the realization on desktop systems [7]. However, a practical and usable implementation of TVDs for end-users must focus on a secure desktop environment and the end-user experience. Hence, in this paper we aim to answer the question whether it is possible to transform an off-the-shelve (secure) operating system, where users may have already certain experience with, into a TVD-enforcing platform that is usable for end-users. For this purpose, we have chosen the OpenSolaris[1] operating system because it is freely available, it uses a desktop system that is known from the popular Linux operating system, and offers in addition advanced security features of a multi-level security (MLS) system based on the Trusted Extensions [10], derived from Trusted Solaris[2].

Contribution.

In particular we make the following contributions:

- We present a secure desktop environment for TVDs based on OpenSolaris (Section 3). We show how to realize separation of data and applications for each TVD based on the OpenSolaris zone mechanism, and how to integrate transparent encryption of external storage devices according to a TVD policy.

http://doi.acm.org/10.1145/1867635.1867650

[1]See: http://www.opensolaris.org
[2]See: http://www.sun.com/trustedsolaris/

- We describe the details of our implementation (Section 4) and show how to map the non-hierarchical TVD security model to the MLS system of OpenSolaris. Our implementation consists of a TVD layer that is installed as an additional software package on top of OpenSolaris, and it does not modify the OpenSolaris kernel nor any of its core security features.

- In contrast to most existing TVD implementations, our system has a more user-centric focus and offers the following main advantages: (i) it provides a central management of zone images and TVD policies, which can be easily maintained by an administrator via a web-based graphical frontend; (ii) it has an automated mechanism for efficient deployment of zone images within TVDs; and (iii) it integrates a transparent encryption of remote and external storage (including USB memory sticks) and provides an automatic key management and revocation functionality.

As our system is based on widely deployed and tested code of OpenSolaris, our solution should be stable and efficient enough to be accepted by end-users as well as security administrators. Our implementation is using standard OpenSolaris components and will be made available for download as open source software[3].

2. BACKGROUND

2.1 Trusted Virtual Domains

Trusted Virtual Domains (TVDs) [12, 4, 5] have been developed as a security framework for distributed multi-domain environments leveraging virtualization and trusted computing technologies.

In a virtualized environment, virtual machines (VMs) that share the same physical infrastructure execute operating systems with different applications and services. Each virtual machine runs in a logically isolated execution environment, controlled by an underlying security kernel that acts as virtual machine monitor (VMM). A TVD is a coalition of virtual machines that trust each other, share a common security policy and enforce it independently of the particular platform they are running on. In particular, the TVD infrastructure provides isolated computing environments, secure communication and storage, explicit trust relationships, and transparent policy enforcement within TVDs.

Typically, the central management component is the *TVD Master*, which handles most of the configuration. Moreover, the TVD policies and related cryptographic keys are distributed to the individual physical platforms. On these platforms, a *TVD Proxy* is created for each TVD, which is the local security service that is responsible for the correct policy deployment and enforcement. For end-user desktop systems, the protection of data on external storage devices is particularly important. To ensure that data is kept securely isolated within a TVD, all data leaving the TVD infrastructure has to be encrypted with keys that are controlled by the TVD Master [8].

To implement a TVD, a secure hypervisor with some form of trusted computing support is needed. TVDs have been implemented as research prototypes based on different virtualization technologies [7, 5]. For instance, the research project OpenTC[4] has experimented with Xen [1] and L4 [13]. They leverage Trusted Computing support based on the TPM – in particular, to realize an authenticated boot process and to integrate attestation functionality of the TPM to verify the client platform integrity, and for the protection of cryptographic keys. Commercial products that support TVDs are becoming available like, e.g., Turaya made by Sirrix security technologies[5].

The problem with existing TVD implementations is that they are not directly suitable for a desktop environment, e.g., they lack a TVD-specific user interface or they require large virtual machine images to be downloaded. Moreover, they are not easy to implement [7] and they are not widely deployed or tested by a large user-base. Ideally, we want to have an off-the-shelf operating system that can be easily transformed to a TVD-enforcing platform. However, commonly used mainstream operating systems lack support for essential security functionality, such as isolated execution environments or secure user interface systems.

2.2 Security Features of OpenSolaris

OpenSolaris is well-suited to realize TVDs because it provides sophisticated security mechanisms and operating system features. In this section, we briefly revisit the most relevant security features of OpenSolaris.

Zones. The lightweight virtualization technology integrated in OpenSolaris, called *zones* [14], provides the illusion of exclusive access to the shared resources of a physical machine. Zones allow users to run different instances of OpenSolaris sharing the same kernel. This approach lowers the resource requirements and enables the execution of more virtualized environments than with full virtualization solutions. As the virtualization is implemented on the operating system level, only OpenSolaris can be used as host and guest operating system (however, OpenSolaris also offers a Linux API). The kernel ensures that from within a zone, it is not possible to affect processes, filesystems, and users outside of the zone and that there are no name and resource conflicts with other zones.

Multi-Level Security. OpenSolaris supports mandatory access control (MAC) according to a multi-level security (MLS) policy [2]. OpenSolaris uses labels [14] to associate a protection level with data, files and users, and enforces access decisions based on these labels. Labels consist of two components: classifications (levels) and compartments (categories). Access control decisions are taken based on the dominance order: Label L_1 dominates label L_2 if and only if the classification of L_1 is higher or equal to the classification of L_2, and the set of compartments of L_1 contains all compartments of L_2.

To enforce a MAC policy, the OpenSolaris kernel compares the security labels of resources with the label of subjects that request to access these resources. The MAC system heavily relies on zones to enforce separation and isolation of data. A global zone is used to manage the system's trusted computing base (TCB) which is a specially protected environment equipped with more privileges and access rights than other zones.

[3]See: http://www.trust.rub.de/projects/tvd-solaris

[4]See: http://www.opentc.net

[5]See: http://www.sirrix.com

Secure GUI and Trusted Extensions. OpenSolaris features a multi-level secure GUI desktop which follows the general principles of [9]. In contrast to other MLS systems, the GUI is based on the OS-independent Gnome desktop environment, which is extended with several security-relevant features. It allows running applications with different security levels in parallel and makes them easily identifiable by displaying a colored stripe on top of each application window. This information is also displayed in the trusted path located on the top of the screen, which is important because users can now easily determine which level of trust a specific application or part of the system has by moving the mouse over a specific item. Moreover, it prohibits fake applications that trick the user into entering sensitive data such as credentials into untrusted applications. The operating system ensures that no application is able to overwrite the trusted stripe to trick the user.

The MLS extension and the secure GUI for OpenSolaris are provided by the (optional) *Trusted Extensions* package. This package is based on Trusted Solaris, a security-enhanced version of Solaris 8 with MLS support. The Trusted Extensions assign a specific label to each zone, which permits to connect a labeled workspace to this particular zone. Only one zone per label per host is allowed. The communication between different hosts with Trusted Extensions installed can be secured and isolated by using Commercial IP Security Option (CIPSO) [14] which labels IP packets.

3. REALIZING TVDS ON OPENSOLARIS

In order to realize TVDs on OpenSolaris, the TVD security model has to be mapped onto the OpenSolaris MLS system. Although Trusted Extensions provide network separation, process isolation, and a secure desktop, the TVD concept demands more. The problem is to integrate various TVD policies and a common management into the system to allow features like transparent encryption of external storage. Moreover, as supposed to be used as a desktop system for users, a fast deployment and configuration of working environments is necessary.

The general idea of our architecture is to use the built-in lightweight virtualization features of OpenSolaris, i.e., the zones, to separate the different TVDs from each other. The global zone executes the necessary management code, and deploys and starts the virtualized environments (zones) representing a TVD. Our system relies on the OpenSolaris kernel which enforces and provides security features such as mandatory and discretionary access control. For intra-TVD communication, our TVD layer establishes logical links between the virtualized environments on different platforms that belong to the same TVD. This logical network is completely isolated from any network traffic from outside that TVD, thus establishing secure channels between the TVD members. The transmission of policies and keys, as well as management messages, is separated in another logical network which cannot be accessed by any TVD. This management network is also used for accessing the network storage that is provided to every user as persistent storage mechanism. Our architecture is illustrated in Figure 1.

3.1 Separation of Data and Applications

One of the main aspects of TVDs is the separation of data and program execution and therefore the isolation and mediation of access to shared resources. As we rely on OpenSolaris, we can use the security functionality of zones and the existing MLS-aware desktop, as illustrated in Figure 2. The image shows the trusted path and two applications running in two TVDs on the same desktop. To enforce the separation of the TVDs, we configure the system to forbid copy & paste data transfers between different TVDs and to notify the user with a pop-up. In order to support a centrally controlled management of TVDs, we added a management layer on each platform that dynamically configures the operating system services as required by the TVD policy.

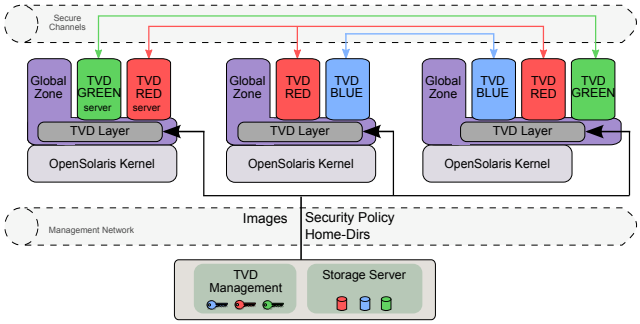

Figure 1: Architecture Overview

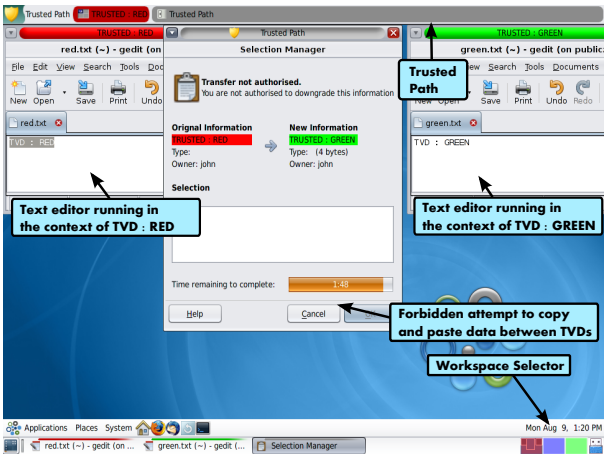

Figure 2: The OpenSolaris Trusted Desktop

To separate the network traffic of different TVDs, we use labeled IP (CIPSO). This implies that — in contrast to TVD implementations that virtualize network traffic on ISO/OSI layer 2 (c.f. [6]) — our TVD framework currently only supports IP-based network traffic. However, in particular for desktop environments, support for IP-based network traffic should be sufficient for most application scenarios. The necessary configuration of IP address ranges, as well as the restriction of members of a TVD to the corresponding address range, is centrally managed by the TVD Master. To enforce this network policy automatically, we implement a security service in our TVD layer, which configures virtual network connections and parameters dynamically whenever a new TVD is added to the platform.

3.2 Transparent Storage Encryption

Our TVD layer also implements a security service that transparently encrypts the data stored on external storage.

Moreover, to offer the user a management option for mobile storage devices (MSD), we add GUI elements to the system. As we can see in Figure 3, the user is able to label new USB sticks, i.e. assign them to a TVD, and to remove the device from the system again. For the users, those are the only actions they are concerned with, as encryption and management of cryptographic keys are handled automatically and transparently by the TVD layer.

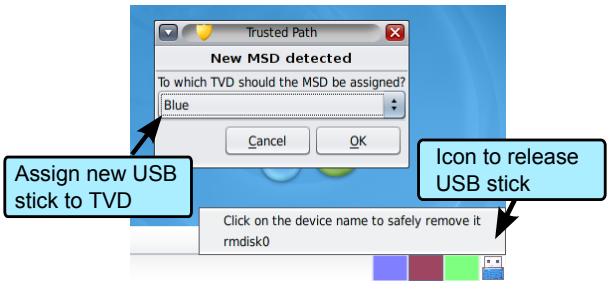

Figure 3: Managing mobile storage devices

4. IMPLEMENTATION

In this section, we describe our implementation in detail, and particularly how we handled the technical challenges that arise when using a multi-level security operating system to realize TVDs by adding and modifying operating system components. As our framework is a complex system we present only the most important key components that were not straight forward to implement and provide an important feature of our implementation.

4.1 Labeling

In our TVD implementation, strict isolation of data and processes is the main goal. We therefore take advantage of the flexibility of the MLS labeling system by creating a labeling scheme that only uses and allows disjoint labels to forbid the transfer and access to data across different zones. This means that we use one common classification for all TVDs, as the classifications are strictly ordered and no way exists to circumvent the ordering.

The separation of TVDs is achieved by introducing a non-hierarchical relationship in the compartment component of the labels. Technically, this is done by assigning only one distinct compartment bit to every TVD which prevents any ordering and dominance relationship. The limitation of this approach is that only 240 disjoint compartments, and therefore 240 TVDs, can be created due to the internal, limited set of compartment bits in OpenSolaris.

4.2 User and TVD Management

In contrast to previous TVD implementations, our management is user-centric and designed for deployment in centrally managed corporate environments on the desktop. User management is therefore an integral part and we enable TVD users to log in to any platform assigned to the TVD infrastructure without prior registration on that platform. After the login the TVD framework creates the defined software environment and allows to access the data permanently stored by the user.

To achieve this behavior, our current prototype distributes the `passwd` and `shadow` files, which are maintained by the master server, to the platform on each boot[6].

During the development of the management GUI, we focused on hiding the technical complexity. The administrator does not have to care about choosing the right label when creating a TVD or remember to create a home directory on the storage server. All these steps are automatically performed, which leaves less room for critical mistakes and lets the administrator focus on the organizational aspects of TVDs. For example, when administrators want to create a new TVD, they have only to choose a name, description and network segment associated with that TVD. When a user is assigned to a TVD, the user will see the corresponding TVD as an option to work in during the next login on a computer that is a member of the TVD framework. The GUI offering the choice can be seen in Figure 4.

4.3 Policy Distribution and Enforcement

The behavior of our TVD framework (e.g. isolation properties) is defined in a security policy. When this policy is applied, our components translate the abstract parts of the policy into OpenSolaris commands and configurations, which are enforced by the underlying security subsystems. We divide the security policy into three parts for efficient distribution. The *Global Policy* defines in a client-ready format the MLS labels we adopted to create the TVDs. Moreover, it contains the information that allows users to log in on every machine that is part of the TVD framework. The *Local Policy* is an XML file that is generated after a user has successfully logged in. It contains the TVDs available to this user, the allowed computing environments, network configuration and detailed MLS label information. To establish a secure channel for policy deployment and authentication, each platform has a *Platform Policy*, which contains a symmetric key that is shared with the master.

As every policy is automatically generated by the master there is no need for an administrator to edit XML configuration files. Manual work is only needed when a new physical machine should be added to the TVD system. The administrator has to install the TVD layer software on the platform and needs to securely transfer the platform policy, containing the key used to authenticate the platform, to that host. Subsequently, all management tasks can be performed with the web management system on the master. This also prevents the definition of invalid or conflicting policies by an administrator, as the consistency of the policy is ensured by the master.

4.4 Protected Computing Environments

Users should be able to work with different software applications according to the needs of their TVDs. These software configurations must then be executed in protected computing environments. In our system, the protected environments are based on the concept of zones, and the corresponding software configurations are basically virtual OpenSolaris user-space images, which can be transferred over the network. Therefore, these images contain all the software available to a user inside of a TVD. It is even possible to allow multiple images for a user which makes it possible to adapt the images to the needs of the user. For example,

[6]Note that this is just a prototype implementation. In a real productive system one would use, e.g., an LDAP server or similar for user authentication.

when a TVD is used for development, the system can offer an environment containing a compiler and Integrated Development Environment (IDE) while the same user has only access to office and spreadsheet applications in another TVD used for document management. This approach is also necessary as a zone could not be created on-the-fly as the native OpenSolaris zone installation process is very slow.

4.4.1 *Efficient Zone Image Deployment*

To support a wide range of zone environments, we have designed a system that is able to download zone images dynamically from a server, adding a lot of flexibility to that approach. For performance reasons, the zone images are cached on the local hard drive and only loaded when necessary. To prevent tampering of zones by an offline adversary and during transmission, every zone has a unique cryptographically secure hash value associated with it, which ensures the integrity of a zone before every boot. The hash values of the zones that are permitted to run in a TVD are specified in the local policy. Due to the OpenSolaris whole-root zone model, which provides the independent execution environments, the disk space needed for a zone is up to 1.4 GB for a standard zone residing in a ZFS[7] dataset (and 350 MB as a compressed backup of the dataset). If every deployed zone had this size, the zone deployment process would be too slow and the amount of data to store and transmit too high to be usable in practice. However, it is clear that almost every zone will need a base installation (GUI extension, text editor, common libraries) plus some smaller modification to adapt them to the TVD use case.

We achieved a considerable reduction of the disk space occupied by our zone instances by using the ZFS snapshot and clone functionality. In our implementation, every image is a snapshot of a zone representing a software configuration that is defined and authorized by the TVD policy. These snapshots can have dependencies to allow delta image files containing only the information that differs from the base image. This leads to a tree-like organization, supporting multiple base images which have leaves as well. These leaves can be further configured into different dependable states. This relationship is specified by the image ID which consists of a number specifying the base, a character determining the leaf that is derived from this specific base, and a number specifying the snapshot in this leaf.

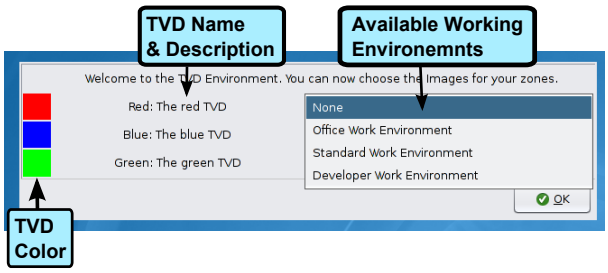

Figure 4: New GUI for choosing TVD zones

4.4.2 *Benefits of Protected Computing Environments*

A benefit of this approach is that the administrator has the ability to upgrade software by specifying a derived image that, e.g., contains fixes for a bug found in an application, and by disallowing the old images. On reboot, the delta image is downloaded by the platform, and the new software can start without a disruption or long period of waiting for the user, and without the need for an administrator to get local access to the platform. This realizes an efficient mechanism for software management on TVD platforms.

[7]ZFS is a special filesystem of OpenSolaris.

4.5 Protected Remote Home Directories

Our TVD implementation provides automatically encrypted remote home directories. They are of special importance as there is no possibility to store user data persistently in our zones. This is due to the fact that we create fresh zones on each reboot for the purpose of integrity measurement. Therefore, we combine the Network File System (NFS), enabling us to access the home directories over the network, with loopback devices mounted from a container file. This mechanism is called *lofi* and has built-in encryption support (AES-256), enabling us to store the containers with the user data on the network and making them available as normal filesystem to a user on every platform in the TVD system.

When a user logs in to a system, the user's NFS share with the encrypted containers is mounted transparently. However, the OpenSolaris Trusted Extensions only allow read or write access on NFS servers with the same label as the zone they are mounted into. As a result, our TVD framework creates and labels a special zone on each platform. This zone hosts the user's remote share container. Subsequently, lofi is used to create the decrypted loopback devices, which are mounted as home directories into the corresponding zones.

To separate the data associated to TVDs, every user has a different container file for every TVD he or she is a member of. From outside these containers look like a huge binary file making it impossible to derive even meta-information or filenames. Unfortunately, this also introduces the problem of a fixed storage size, as the containers cannot grow dynamically in size. However, this could be resolved by an automatic mechanism to transfer the data of a container file into a bigger one when the storage space is not sufficient.

4.6 Mobile Storage Devices

We implement the transparent encryption of mobile storage devices (MSDs) in a similar way as the protected remote home directories. Again lofi loopback devices provide the required encryption capabilities, but now the containers reside on a mobile device. Moreover, key retrieval is more difficult in the mobile scenario as the system does not know in advance which devices will be plugged into which platform. Therefore, following the approach of [8], we have implemented a key download mechanism, which requests the device key from the master server when needed and transmits it to the platform. In our implementation, all necessary information needed to retrieve the correct key is embedded into the device's XML identification record which is protected by a cryptographic signature created by the TVD master. The TVD master also maintains the public and private key and handles the distribution of the identification record onto the platforms. As a consequence, the record's signature of every MSD is validated before the key is requested from the master.

4.6.1 *Creation and Usage*

When new storage devices are introduced into the TVD

system, the user is able to create, depending on the policy, new encrypted devices. Therefore, the platform generates a random device ID and requests a signed identification record and an encryption key from the server. After erasing all data on the device, a new lofi container is created and saved together with the signed identification record. The only necessary interaction by the user is to choose the TVD to which the MSD should be attached to. Subsequently, the device can be used on all platforms that are connected to the same master under the condition that a zone of the corresponding TVD is running. No interaction with the user is needed: When the user plugs in a device, it automatically appears as an icon on the desktop and can be used instantly.

4.6.2 Revocation

An inherent issue with mobile storage devices is revocation, for instance, because they might get lost or the original creator might have to leave the company. With normal USB drives, the company has no chance to prevent the exposure of private or sensitive data to unauthorized entities. To solve this problem, our master web GUI includes a list of all mobile storage devices in the system. To support auditing of the use of MSDs, the user who created the device and the time he created it are stored on the master. This allows an administrator to identify rogue devices and to revoke them by deleting their records on the master. As a result, the platforms can no longer request the decryption key and any new access to the contents of the MSD becomes impossible.

All these management options are a great benefit since no user or administrator has to handle keys manually when working with the TVD framework. Creation of container files, key management, and validation is automatically handled by the master.

5. CONCLUSION AND FUTURE WORK

In this work, we have shown that it is possible to implement TVDs for end-user desktop systems based on OpenSolaris. Our TVD framework features integrated management and transparent data encryption, an efficient deployment of zone images, and puts a particular focus on the ease of administration. Our implementation adds a TVD layer to the OpenSolaris system without any modification of the existing kernel or core security features. All features described in this paper can be managed via a central administration GUI which creates and maintains the TVD policy.

In the future, we plan to make our TVD framework interoperable with existing implementations for data centers, as they complement each other and could cover many current enterprise scenarios. We also plan to integrate Trusted Platform Module (TPM) support, which our prototype currently lacks, in order to measure and verify the integrity of the whole platform from the startup of the kernel to the deployment of zones. Furthermore, we will investigate the realization of secure inter-domain information transfer.

6. REFERENCES

[1] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. L. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In *19th ACM Symposium on Operating Systems Principles (SOSP'03)*, pages 164–177. ACM Press, 2003.

[2] D. Bell and L. LaPadula. Secure computer systems: Mathematical foundations. Technical Report MTR-2547, Vol 1, MITRE Corp., Bedford, MA, Nov. 1973.

[3] S. Berger, R. Cáceres, D. E. Pendarakis, R. Sailer, E. Valdez, R. Perez, W. Schildhauer, and D. Srinivasan. TVDc: Managing security in the trusted virtual datacenter. *Operating Systems Review*, 42(1):40–47, 2008.

[4] A. Bussani, J. L. Griffin, B. Jansen, K. Julisch, G. Karjoth, H. Maruyama, M. Nakamura, R. Perez, M. Schunter, A. Tanner, L. V. Doorn, E. A. V. Herreweghen, M. Waidner, and S. Yoshihama. Trusted Virtual Domains: Secure foundations for business and IT services. Technical Report RC23792, IBM Research, 2005.

[5] S. Cabuk, C. I. Dalton, K. Eriksson, D. Kuhlmann, H. V. Ramasamy, G. Ramunno, A.-R. Sadeghi, M. Schunter, and C. Stüble. Towards automated security policy enforcement in multi-tenant virtual data centers. *Journal of Computer Security*, 18(1):89–121, 2010.

[6] S. Cabuk, C. I. Dalton, H. V. Ramasamy, and M. Schunter. Towards automated provisioning of secure virtualized networks. In *ACM Conference on Computer and Communications Security (CCS'07)*, pages 235–245. ACM, 2007.

[7] L. Catuogno, A. Dmitrienko, K. Eriksson, D. Kuhlmann, G. Ramunno, A.-R. Sadeghi, S. Schulz, M. Schunter, M. Winandy, and J. Zhan. Trusted Virtual Domains – design, implementation and lessons learned. In *International Conference on Trusted Systems 2009 (INTRUST 2009)*. Springer Verlag, Dec. 2009.

[8] L. Catuogno, H. Löhr, M. Manulis, A.-R. Sadeghi, and M. Winandy. Transparent mobile storage protection in trusted virtual domains. In *23rd Large Installation System Administration Conference (LISA'09)*, pages 159–172. USENIX Association, 2009.

[9] J. Epstein, J. McHugh, H. Orman, R. Pascale, A. Marmor-Squires, B. Danner, C. R. Martin, M. Branstad, G. Benson, and D. Rothnie. A high assurance window system prototype. *Journal of Computer Security*, 2(2):159–190, 1993.

[10] G. Faden. Solaris trusted extensions: Architectural overview, Apr. 2006.

[11] Y. Gasmi, A.-R. Sadeghi, P. Stewin, M. Unger, M. Winandy, R. Husseiki, and C. Stüble. Flexible and secure enterprise rights management based on trusted virtual domains. In *3rd ACM Workshop on Scalable Trusted Computing (STC'08)*, pages 71–80. ACM, 2008.

[12] J. L. Griffin, T. Jaeger, R. Perez, R. Sailer, L. van Doorn, and R. Cáceres. Trusted Virtual Domains: Toward secure distributed services. In *Proceedings of the 1st IEEE Workshop on Hot Topics in System Dependability (HotDep'05)*, June 2005.

[13] J. Liedtke. On micro-kernel construction. In *Fifteenth ACM Symposium on Operating System Principles (SOSP'95)*, pages 237–250. ACM Press, 1995.

[14] Sun Microsystems, Inc. *Solaris 10 System Administrator Collection*. http://docs.sun.com/app/docs/coll/47.16?l=all.