In Proc. IEEE Military Communications Conference (MILCOM), October 2015

A Multi-Level Secure File Sharing Server and its Application to a Multi-Level Secure Cloud

Mark R. Heckman, *Member, IEEE*, Roger R. Schell, *Member, IEEE*, and Edwards E. Reed, *Member, IEEE*

***Abstract*—Contemporary cloud environments are built on low-assurance components, so they cannot provide a high level of assurance about the isolation and protection of information. A "multi-level" secure cloud environment thus typically consists of multiple, isolated clouds, each of which handles data of only one security level. Not only are such environments duplicative and costly, data "sharing" must be implemented by massive, wasteful copying of data from low-level domains to high-level domains. The requirements for certifiable, scalable, multi-level cloud security are threefold: 1) To have trusted, high-assurance components available for use in creating a multi-level secure cloud environment; 2) To design a cloud architecture that efficiently uses the high-assurance components in a scalable way, and 3) To compose the secure components within the scalable architecture while still verifiably maintaining the system security properties. This paper introduces a trusted, high-assurance file server and architecture that satisfies all three requirements. The file server is built on mature technology that was previously certified and deployed across domains from TS/SCI to Unclassified and that supports high-performance, low-to-high and high-to-low file sharing with verifiable security.**

***Index Terms*—Cloud computing, Computer security, Multi-level security, Security kernel, GEMSOS, Network file service**

I. Introduction

The U.S. National Institute of Standards and Technology (NIST) defines cloud computing as "a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction" [2]. These benefits of scale promised by cloud computing are naturally attractive to institutions that manage large amounts of data. The United States Government's 2011 Federal Cloud Computing Strategy, for example, called for increased use of cloud computing to address low asset utilization, duplicative systems, and other expensive and wasteful IT inefficiencies. The strategy's "cloud first" policy required Government agencies to consider cloud computing options before making any new IT investments [1].

The security of a cloud service must be as good as or superior to that of the enterprise environments it replaces, because not only is a cloud service expected to keep multi-domain data separate, the concentration of information in a cloud system is a logical target for an attacker, simultaneously offering an economy of scale (because of the concentration of data from potentially hundreds or thousands of domains) and a reduction in exposure risk (because the attacker only need attack one system, not hundreds or thousands). For example, unknown attackers repeatedly penetrated for over a year a NASDAQ cloud application that permitted corporate directors to share sensitive documents before board meetings, potentially revealing the confidential information of hundreds of corporations [3].

Cloud services, however, have many of the same potential security vulnerabilities as self-managed systems, plus additional vulnerabilities specific to the cloud. The result is a situation where virtual servers hosted on the cloud may be less secure than the servers they replaced [4]. A study by Context Information Security, Ltd., for example, found major weaknesses in the security of several cloud services that allowed data from other users to be accessed, and determined that "the current technology used by these providers is lacking from a security perspective" [5][6]. A U.S. Department of Defense task force study concluded that, due to the current state of security in today's cloud technology, "*sensitive, classified, and time-critical DoD applications should be deployed only in private clouds or conventional non-cloud approaches*" [7] (emphasis added).

Suggestions for "fixing" the security of cloud infrastructure include implementing various controls, such as encrypting stored data, requiring the cloud service provider to adopt secure policies and procedures [8][9], and even attempting to validate the security of all applications that are permitted to run on the cloud [10]. Those solutions, however, are completely ineffective at solving a fundamental security problem in contemporary cloud computing systems: The shared infrastructure is built using low-assurance, commodity technology vulnerable to software subversion. Without dealing with the threat of subversion, cloud security is broken, no matter how clever the architecture or how many extra layers of security controls are added to the low-assurance components.

One common solution to this problem is to have multiple, isolated clouds, each of which handles data of only one security domain. That way, vulnerabilities within each isolated cloud cannot lead to cross-domain information compromise. An example of this approach is that proposed by Raytheon, where a "Trusted Thin Client" is used to access multiple clouds at different levels [11]. But this design is expensive and not easily scalable when there are a large number of distinct domains.

An additional level of complexity is added in environments where controlled sharing (as opposed to complete isolation) of

M.R. Heckman is with the Center for Cyber Security Engineering and Technology at the University of San Diego, CA 92110 USA (e-mail: mheckman@sandiego.edu).

R.R. Schell is President of Aesec Corporation, Palo Alto, CA 94301 USA, and a member of the University of San Diego advisory group for their Center for Cyber Security Engineering and Technology, San Diego, CA 92110 USA (e-mail: schellr@ieee.org).

E.E. Reed is with Aesec Corporation, Palo Alto, CA 94301 USA (e-mail: ed.reed@aesec.com)

data between different domains must be supported. Cross-domain services (CDS) of this type typically must massively and wastefully copy low domain data to higher domains, using techniques like one-way data diodes.

Not only does this CDS approach not scale, it may not even be secure. The U.S. Government's Unified Cross Domain Management Office (UCDMO) maintains a "baseline list" of commercially available CDSs that are available for deployment by U.S. Government agencies. A cursory examination of the (27 January 2012) list reveals that many of the products are based on low-assurance, commodity technology, which indicates that they share the same weaknesses as the cloud infrastructure components [33].

Raytheon, for example, advertises that the Trusted Thin Client runs on a "trusted operating system with security enhanced features (SELinux)", but the original sponsor of SELinux, the NSA, says that "Security-enhanced Linux is only intended to demonstrate mandatory controls in a modern operating system like Linux and thus is very unlikely by itself to meet any interesting definition of secure system" [34].

Even if the components in a cloud environment were individually evaluable to a high standard of security, another problem remains: how to compose the security of the individual components into an overall system security policy. Systems composed of many different components, even "secure" components, may display emergent behaviors that violate the overall system security policy. Any attempt at formulating a cloud system-wide security policy with high-assurance will likely fail if the overall system is constructed without a sound composition method. Even the international standard *Common Criteria for Information Technology Security Evaluation* (CC) (ISO/IEC 15408), however, does not have a formal approach for handling the composition of individually evaluated components [12].

Arguably, the only proven security composition technique that would scale to systems of the size of a cloud environment is that in the Trusted Network Interpretation (TNI) [14] of the Trusted Computer Security Evaluation Criteria (TCSEC) [13]. That is consistent with the conclusion of David Bell that, "the general composition problem is unavoidable and one faces enormous difficulties in the absence of a network-view like that of the TNI [16]."

The requirements for verifiable, multi-level cloud security are thus threefold: 1) To have high-assurance components available for use in creating a multi-level secure cloud environment; 2) To design a cloud architecture that efficiently uses the high-assurance components in a scalable way, and 3) To compose the secure components within the scalable architecture while still verifiably maintaining the system security properties.

This paper introduces a high-assurance, multi-level secure (MLS) file server – the Aesec Assured Sharing Platform Service (ASPS) – that was built to satisfy the rigorous requirements of the highest assurance level of the TCSEC, Class A1. Evaluation under Class A1 substantially addresses the problem of software subversion and permits the evaluation of a network of ASPS systems according to the mature and proven composition method of the TNI. After describing the ASPS, we sketch a proposed MLS architecture for a cloud storage system that can take advantage of these features of the ASPS.

Section II of this paper describes the ASPS system and its low-to-high sharing architecture (the high-to-low sharing architecture of the ASPS system leverages the virtual guard architecture previously described by Heckman, Schell, and Reed [31] and, due to space constraints, will not be described in this paper). Section III briefly describes cloud storage concepts and the assurance limitations of current cloud infrastructure technology. Section IV explains how the ASPS can be used to implement the proposed multi-level secure cloud storage architecture. Section V concludes the paper and describes future work.

II. MLS File Server

The Aesec Assured Sharing Platform Service (ASPS) was designed to serve as a networked Class A1 file server system that permits controlled data sharing across security domains, both "low-to-high" (a.k.a. "read-down") and "high-to-low" (a CDS sometimes referred to as a "transfer guard"). Aesec has demonstrated a high-assurance prototype of the file server that provides sharing across both one and two domains – low-to-high sharing (U-S, U-TS, S-TS) and high-to-low sharing (S-U, TS-S, TS-U).

A. MLS Sharing Hypervisor

The ASPS is built on the previously-evaluated, high-assurance, and commercial off-the-shelf (COTS) Gemini Trusted Network Processor (GTNP) [27] (currently available as an original equipment manufacturer product from Aesec Corporation). The GTNP incorporates the Gemini Multiprocessing Secure Operating System (GEMSOS) Trusted Computing Base (TCB), which contains the high-assurance, mandatory access controls used to enforce the MLS sharing policy for networked applications [28]. The NSA has previously evaluated the GEMSOS security kernel and product Ratings Maintenance Phase (RAMP) plan at Class A1 as part of the evaluation of the GTNP, confirming that it meets the highest standards for security, protection against subversion, and certifiability [27].

The GTNP implements the most critical feature identified by Paul Karger as a requirement in an MLS sharing hypervisor: a secure shared file store [25]. A secure shared file store is secure storage at the level of the hypervisor. The secure shared file store in GEMSOS is provided by the hardware-enforced storage objects – segments – managed by the kernel [28]. Untrusted code outside the TCB creates the file abstraction using segments and cannot bypass the kernel's enforcement of the MLS policy.

The kernel provides complete protection required for the MLS "low-to-high" sharing capability, implementing "secure read-down". No trusted code outside the kernel is required to permit a high-level subject to have read access to segments in a lower-level domain while simultaneously and securely allowing a low-level subject to have read and write access to the same data, so there is no need to copy the data into the high domain. Similarly, only untrusted code is used to implement the network stack and network file service in the ASPS, significantly simplifying certification and accreditation efforts.

B. ASPS Low-to-High Sharing Architecture

Heckman, et al., previously described the high-to-low sharing architecture of a MLS virtual guard system [31]. This

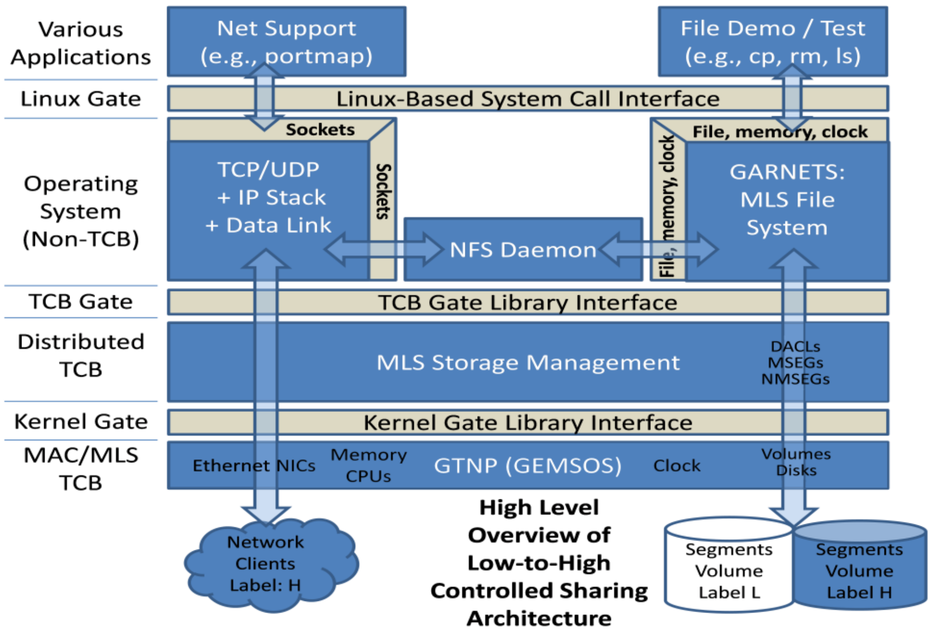

Fig. 1. MAC File Server Architecture for Low-to-High Sharing.

method of high-to-low sharing has been adopted by the ASPS and will not be described here. The architecture of the ASPS for low-to-high sharing is depicted in Fig. 1.

The Aesec COTS Gemini Application Resource and Network Support (GARNETS) file system, which runs as an untrusted application in a layer above the TCB, creates the file management capability using TCB objects that are themselves built using kernel-managed segments [35]. The multi-level secure file storage system on the ASPS operates like standard file storage, but the data is securely labeled with mandatory access classes and the TCB reliably isolates security domains, permitting only sharing allowed by the MLS policy.

The GARNETS file system creates a logical name space that spans domains, leveraging the MLS property that higher domains can read lower domain data (i.e., the directory and file structure of lower domains), but not modify it in any way. Thus, secure "low-to-high" sharing does not require trusted code outside of the TCB or copying of unused file data to higher domains, because higher domains have direct access to lower-domain data.

The interface to the system is the widely-used Network File System (NFS) [37]. The NFS Daemon and network stack run as untrusted code in a layer above the TCB. There is a different NFS Daemon and network stack for each domain. In the ASPS prototype, there is a separate network interface card (NIC) for each domain, although crypto-seal guard appliances, such as those described in [31], can be used to create multi-level interfaces through a NIC shared by multiple domains.

Clients access files on the server through NFS calls. The NFS Daemon retrieves or modifies files through GARNETS. The kernel alone provides complete protection for low-to-high sharing, so high domain requests cannot modify low domain data and high domain accesses to low domain data are completely invisible to the low domain. The untrusted NFS Daemon and GARNETS cannot bypass the kernel's enforcement of the MLS policy.

As shown in the figure, various other services run as applications above the TCB, calling the network stack and file system through a POSIX-style system call interface that is intended to support Linux application source-code compatibility. This is where the code to implement a cloud system also resides.

The ASPS can be configured for many different policies. As shown in the figure, it can be configured for controlled data sharing between domains, but it can also be configured, for example, to have completely separate and isolated domains where no sharing is possible – creating completely separate file storage systems, which is a common scenario in multi-tenant cloud environments – using mandatory access class "categories".

GEMSOS assigns mandatory security labels to every subject and object managed by the TCB. The security labels contain both secrecy and integrity components. Each component consists of a hierarchical level and set of categories. The set of possible security labels (combinations of secrecy levels, secrecy categories, integrity levels, and integrity categories) form a partially-ordered lattice of access classes of the type used in the Bell and La Padula security model [36].

It is a property of a partially-ordered lattice of access classes that some classes are "non-comparable" so that, for any two domains whose access classes are non-comparable, the security label for neither domain dominates the other and so each domain is completely isolated from the other domain. Non-comparable access classes can be implemented in the ASPS simply by assigning unique sets of integrity or secrecy categories where the set of categories of any domain are neither a superset nor subset of any other domain's category set. The Class A1 ASPS can thus verifiably emulate complete physical isolation of security domains through high-assurance MAC domain separation by using non-comparable access classes.

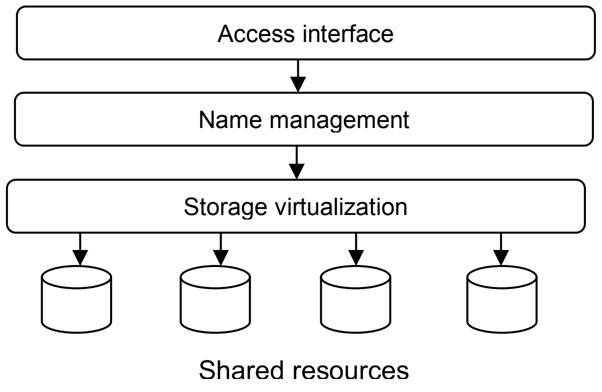

Fig. 2. Abstract model of a cloud storage service. The service maps object names to actual locations in the pool of shared storage resources.

III. Cloud Storage Concepts and Assurance Limitations of Current Cloud Technology

Cloud services present a customer with an interface to a large, logical resource space made up of shared, distributed physical resources. The cloud system components are responsible for isolating and protecting the logical resources.

A. Cloud Storage Services

An abstract model of a cloud storage service is shown in Fig. 2. Users access objects in the cloud service through a network interface, usually web-based and using the SOAP [39] or REST [38] design models. The service implements a global object naming system to differentiate between the different data objects it stores. For example, in the Amazon Simple Storage Service (Amazon S3), an object consists of file data and metadata. The metadata is a set of name-value pairs that describe the data, such as "date last modified", plus customer-defined data. Each object is contained in a "bucket". Buckets can contain one or more objects, and provide the high-level name for the data objects. The high-level name is used for account identification and usage measurements, and must be unique in the entire S3 system [17].

A cloud storage service maps object names to the actual storage locations in the shared pool of storage systems, presenting a unified virtual storage space to the customer. Most cloud systems transparently handle distribution, replication, consistency, and other data integrity and reliability requirements, as well as the rapid provisioning of additional resources when needed and the release of resources that are no longer needed that are the hallmarks of cloud storage.

B. Assurance Limitations of Current Technology

Cloud storage management software typically runs on commodity, low-assurance hardware, hypervisors, and operating systems. The widely-used and open source Xen hypervisor, for example, underlies such popular services as Amazon's Elastic Compute Cloud [19]. Xen is a hypervisor that runs directly on the host hardware. The main focus of the Xen hypervisor is to isolate guest virtual machines, called "domains", from each other and from the underlying hardware. Device drivers and other critical operating system services, however, are not part of Xen, but are in one or more domains [20].

Although some have called the Xen hypervisor a "trusted computing base" [18], Xen is not, as the TCSEC defines a TCB, "all of the "hardware, firmware, and software critical to protection" [13]. The Xen architecture as a whole contains many components outside of the hypervisor – hardware and software –that would need to be evaluated together with the hypervisor in order to create a high-assurance system that could be trusted with sensitive data. This includes the device drivers and resource management components typically installed in Xen Domain 0 (and commonly implemented in a low-assurance Linux variant [30]).

Attempts are being made to create a "secure" Xen through various means, such as by shrinking the hypervisor and putting more functions in separate domains, adding special security modules, etc. (e.g., [21]). These efforts are doomed to fail, however, until they address a fundamental assurance problem: Any trusted component not evaluated and maintained as part of the TCB can be subverted, rendering the assurance of the rest of the system moot. An exemplary published vulnerability demonstrates how a Xen system can easily contain many unknown flaws when it is not evaluated together with the hardware on which it runs [32].

Versions of VMWare ESXi, another widely-used, but proprietary, virtual machine manager, have undergone a security evaluation under Common Criteria for Information Security Evaluation [15], achieving an Evaluation Assurance Level (EAL) of 4 in 2010 [22]. But EAL 4 is a relatively low standard for assurance, achieved by many other systems of questionable security, such as Windows XP [23]. More recent versions of ESXi, as part of VMWare's vSphere product, were evaluated at only the much lower EAL 2 [22]. Moreover, as mentioned earlier, the Common Criteria does not have a formal approach for handling the composition of individually evaluated components, precluding any attempt to create a high-assurance composed system using ESXi.

This same assurance problem persists in more sophisticated, "high-assurance" systems proposed to maintain isolation and security (in cloud environments and otherwise). In general, the systems may suffer from three different kinds of problems: 1) They include low-assurance components in the TCB, which negates any assurance arguments made about the rest of the TCB; 2) They add large amounts of trusted, but not necessarily trustworthy, code with simultaneous interfaces to domains of different levels; or 3) they fail to address fundamental and inherent scientific limitations on making requests securely from high to low through low-assurance interface components, thereby opening a covert (or possibly overt) channel through the pattern of requests from the high level.

For example, NSA examined a project built using the Green Hills Software INTEGRITY-178B separation kernel, which had been certified CC at the EAL6+ level (EAL 7 is the highest) using the Separation Kernel Protection Profile (SKPP). The separation kernel was used to provide multi-level separation between virtual machines on the same workstations. But the separation kernel was originally certified with a relatively simple hardware platform, not the commodity workstations used in the project. NSA concluded that the commodity workstations hardware and firmware were too complex to be evaluated and that "the use of an SKPP certified kernel as one part of a system does not immediately make a system in totality highly robust [24]." Shortly afterward, NSA "sunsetted" the protection profile under which the separation kernel had been evaluated, saying that "conformance to this protection profile, by itself, does not offer sufficient confidence that national security information is appropriately protected in the context of a larger system in which the conformant product is integrated [26]".

IV. Proposed MLS Cloud Storage Architecture

A typical cloud environment consists of many servers. The cloud management software runs on top of the hypervisor and handles linking the various servers into a single virtual storage space. A MLS cloud storage system (MLSCSS) that permits data sharing between domains can be built using multiple, networked, ASPS hosts. The cloud storage management software in the MLSCSS runs as untrusted code outside the TCB on each host.

A. Proof-of-concept MLSCSS Design

In our simple, proof-of-concept, MLSCSS design, the cloud system consists of two kinds of servers: 1) a single metadata server that is the interface to the storage cloud and that maintains the metadata directory that maps object names to their actual storage locations in the cloud; and 2) multiple, storage servers that store the data. The metadata server and storage servers in the MLCSS are all ASPS systems.

Clients access objects (files) through the metadata server using Representational State Transfer (REST) over HTTP [38], connecting to the metadata server through the network interface assigned to the client's domain.

Every object stored in the MLCSS has a unique identifier. Because the system is MLS, the unique identifier consists of a name and the security domain of the object. The security domain of a data object is determined by the network interface of the metadata server over which the request to create the object arrived. In the initial prototype design, clients can only create objects in the client's domain, although it would be technically possible to create objects in a higher domain. Clients in higher domains can retrieve objects from lower domains by specifying the (name, domain) identifier of the object.

There is a metadata directory for each domain that maps object identifiers to the object storage location. The metadata server has an untrusted mapping process for each security domain that maintains the metadata directory for its domain. Due to the MLS "read down" property, the mapping processes for higher domains can also read the metadata directories of lower domains, so higher domains have the ability to locate and fetch objects contained in their own and lower domains.

Once the mapping process has located the object, it spawns an object service process that sends the object access request via NFS to the storage server that holds the object. The object service process acts on behalf of the client that made the request and returns the result of the access request to the client.

Each storage server is an ASPS that requires no additional code. The metadata server accesses the objects on a storage server through NFS operations supported by the ASPS. The metadata server is aware of the storage capacity of each of the available storage servers and, as one storage server's capacity is reached, can put new objects on servers with available storage. In the prototype design, storage is pre-allocated to each domain on a server to avoid illegal information flow channels due to resource exhaustion.

New storage servers to expand storage capacity can easily be incorporated by updating the metadata directories to reflect the newly added storage. Scalability of the metadata directories can be maintained by partitioning the directories onto different metadata servers, each managing a non-overlapping subset of the total space.

B. MLS Cloud Network

The ASPS prototype has a separate network interface for each domain. In the MLSCSS proof-of-concept design, therefore, there must be a separate network that connects the metadata server to the storage servers for each domain. Obviously, this does not scale. The challenge is to maintain the MLS feature across multiple servers in a multi-domain environment using a shared network.

Heckman, Schell, and Reed described how this can be accomplished using a device to encrypt, seal, and forward packets from different domains to create virtual networks, securely multiplexing multiple access domains on the same network. These devices, called GemSeal guards, which are themselves built on GEMSOS, seal packets with their source label and forward them over the shared network. The seals protect the integrity of the labels and data. Unlabeled or altered packets cannot enter a guarded destination because they will not have a crypto seal that binds a label to a matching destination label, preventing high-domain data from being released to low-domain clients [31].

C. Composition into a Class A1 Cloud

Each of the servers that make up the MLSCSS (and the crypto-seal GemSeal devices, if used) is a Class A1 network component. The servers can be composed into a high-assurance, Class A1 cloud system following the Trusted Network Interpretation (TNI) of the TCSEC [14].

The TNI defines a method for composing a Network TCB (NTCB) that consists of NTCB partitions. Each of the servers in the MLSCSS meets the requirements for an NTCB partition in that there is a clearly defined TCB with a definitive protection domain boundary. Using either a separate network for each domain or using a crypto-seal guard to create virtual networks, the TNI requirement for a secure networking path, i.e., "channel", is also met. Thus, the composition technique in the TNI can be used to compose a Class A1 cloud infrastructure.

V. Conclusion

In this paper we have presented a multi-level secure file sharing server and described how the secure file server can be used to create a high-assurance, MLS storage cloud. The server and proposed architecture satisfy the three requirements for creating certifiable, scalable, multi-level cloud systems 1) Availability of trusted, high-assurance components; 2) An architecture that efficiently uses the high-assurance components in a scalable way, and 3) A composition method to compose the secure components within the scalable architecture while still verifiably maintaining the system security properties. Our proposed system permits controlled, scalable, and verifiably secure sharing between domains. Notably, the software that virtualizes the shared storage resource pool into a single virtual storage space is untrusted code, simplifying certification and accreditation efforts, and the system eliminates the need for massive copying of data from low to high domains.

Our prototype design, however, does not include all of the functionality of current cloud storage systems. Cloud software for administration, backup, migration, etc. that was originally developed to be used in a single-domain environment will need to be adapted to work in an MLS environment. This will be a focus of future work, along with development of a working prototype. Additional research will focus on techniques for secure connectivity between components (through further development of the GemSeal guards) and high-assurance identity management.

REFERENCES

[1] V. Kundra, Federal Cloud Computing Strategy, 8 February 2011. (http://www.cio.gov/documents/Federal-Cloud-Computing-Strategy.pdf).

[2] P. Mell and T. Grance, The NIST Definition of Cloud Computing: Recommendations of the National Institute of Standards and Technology. Special Publication 800-145. National Institute of Standards and Technology, September 2011 (http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf).

[3] "Hackers penetrated Nasdaq computer network" Feb. 5, 2011 (http://www.nbcnews.com/id/41435523/ns/us_news-crime_and_courts/t/hackers-penetrated-nasdaq-computer-network/).

[4] Dark cloud: Study finds security risks in virtualization, Government Computer News, 18 March 2010 (http://gcn.com/articles/2010/03/18/dark-cloud-security.aspx).

[5] Assessing Cloud Node Security. White paper. Context Information Security, Ltd., March 2011 (http://www.contextis.co.uk/research/white-papers/assessing-cloud-node-security/).

[6] M. Jordon and J. Forshaw, Dirty Disks Raise New Questions About Cloud Security. Blog. Context Information Security Ltd., 24 April 2012, http://www.contextis.co.uk/research/blog/dirtydisks/

[7] Task Force Report: Cyber Security and Reliability in a Digital Cloud, Department of Defense, Defense Science Board, January 2013. http://www.acq.osd.mil/dsb/reports/CyberCloud.pdf

[8] Security Guidance for Critical Areas of Focus in Cloud Computing V3.0, Cloud Security Alliance, 14 November 2011 (https://cloudsecurityalliance.org/research/security-guidance/).

[9] Creating Effective Cloud Computing Contracts for the Federal Government: Best Practices for Acquiring IT as a Service, CIO Council, CAO Council, and Federal Cloud Compliance Committee, 24 February 2012 (http://www.cio.gov/cloudbestpractices.pdf).

[10] G.I. Seffers, "U.S. Army innovates on cloud computing front", March 1, 2013, Signal Online, http://www.afcea.org/content/?q=node/10743.

[11] Secure Cloud Computing Across Multiple Sensitivity Levels. White paper. Raytheon Company, 2014 (https://www.trustedcs.com/resources/whitepapers/RTN_WP_CloudsFINAL.pdf)

[12] M.S. Merkow and J. Breithaupt. Computer Security Assurance Using the Common Criteria. Delmar Publishers Inc., Albany, NY, USA, 2004. p. 194.

[13] Department of Defense Trusted Computer System Evaluation Criteria 5200.28-STD, United States National Computer Security Center, December 1985 (http://csrc.nist.gov/publications/history/dod85.pdf)

[14] Trusted Network Interpretation of the Trusted Computer System Evaluation Criteria, DoD 5200.28–STD, 31 July 1987, NCSC–TG–005. Available: http://csrc.nist.gov/publications/secpubs/rainbow/tg005.txt

[15] Common Criteria for Information Technology Security Evaluation, Version 3.1, CCMB-2009-07-001, July 2009 (http://www.commoncriteriaportal.org/cc/).

[16] Bell, D.E., Looking back at the Bell-La Padula model, *Computer Security Applications Conference, 21st Annual*, pp.15 pp.,351, 5-9 Dec. 2005.

[17] Amazon Simple Storage Service, Developer Guide, API Version 2006-03-01, http://awsdocs.s3.amazonaws.com/S3/latest/s3-dg.pdf

[18] D. Murray, G. Milos, and S. Hand, 2008. Improving Xen security through disaggregation, in Proc. of the 4th ACM International Conference on Virtual Execution Environments (VEE 2008). Seattle, WA, USA, 2008 (http://www.cl.cam.ac.uk/research/srg/netos/papers/2008-murray2008improving.pdf)

[19] J. Brodkin, Zynga's cloud runs just like Amazon's, but it's all in-house, ArsTechnica, 9 May 2012, http://arstechnica.com/business/2012/05/how-amazon-saved-zyngas-buttand-why-zynga-built-a-cloud-of-its-own/

[20] Why Xen? White paper. Xen.org, 12 August 2010 (http://xen.org/files/Marketing/WhyXen.pdf).

[21] G. Coker, Xen Security Modules (XSM) (http://mail.xen.org/files/summit_3/coker-xsm-summit-090706.pdf).

[22] Common Criteria Evaluation & Validation (CCEVS), (https://www.vmware.com/security/certifications/common-criteria)

[23] Microsoft Windows Platform Products Awarded Common Criteria EAL 4 Certification, December 14, 2005, (https://news.microsoft.com/2005/12/14/microsoft-windows-platform-products-awarded-common-criteria-eal-4-certification/)

[24] Separation Kernels on Commodity Workstations, Systems and Network Analysis Center, Information Assurance Directorate, NSA, 11 March 2010. (http://www.niap-ccevs.org/announcements/Separation%20Kernels%20on%20Commodity%20Workstations.pdf)

[25] P.A. Karger, "Multi-level security requirements for hypervisors", in Proc. of the 21st Annual Computer Security Applications Conference (December 5-9, 2005). ACSAC. IEEE Computer Society. (Available: http://www.acsac.org/2005/papers/154.pdf)

[26] http://www.niap-ccevs.org/cc-scheme/pp/pp.cfm/id/pp_skpp_hr_v1.03/

[27] Final Evaluation Report, Gemini Computers, Incorporated, Gemini Trusted Network Processor, Version 1.01, National Computer Security Center, 1995. Available: http://www.aesec.com/eval/NCSC-FER-94-008.pdf

[28] R.R. Schell, T.F. Tao, and M. Heckman, Designing the GEMSOS security kernel for security and performance. In Proc. of the 8th DoD/NBS Computer Security Conference, 1985, pp. 108-119.

[29] R.R. Schell and D.L. Brinkley, "Evaluation criteria for trusted systems", in Information Security: An Integrated Collection of Essays, Abrams, Jajodia, and Podell, eds., IEEE Computer Society Press, Los Alamitos, CA, pp. 137-159, 1995.

[30] "Dom0", http://wiki.xen.org/wiki/Dom0

[31] M.R. Heckman, R.R. Schell, and E.E. Reed, A high-assurance, virtual guard architecture. In Proc. MILCOM 2012, October 29 - November 1, 2012, Orlando, FL.

[32] Vulnerability Summary for CVE-2012-0217, https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2012-0217

[33] "UCDMO Cross Domain Baseline List: As of 27 January 2012", http://www.owlcti.com/pdfs/certifications/UCDMO_v.3.5.0_Baseline_Inventory.pdf

[34] "SELinux Frequently Asked Questions (FAQ)," http://www.nsa.gov/research/selinux/faqs.shtml#I13

[35] C.E. Irvine, "A multilevel file system for high assurance." In Proc. of the 1995 IEEE Symposium on Security and Privacy, 1995. (http://www.cisr.us/downloads/papers/95paper_mls.pdf)

[36] D.E. Bell and L.J. LaPadu1a, Computer Security Model: Unified Exposition and Multics Interpretation, Tech. report ESD-TR-75-306, AD A023588, The Mitre Corporation, Bedford, Mass., June 1975.

[37] B. Callaghan, B. Pawlowski, and P. Staubach, NFS Version 3 Protocol Specification, Sun Microsystems, Inc., June 1995. Available: https://tools.ietf.org/html/rfc1813

[38] R.T. Fielding, Architectural Styles and the Design of Network-Based Software Architectures. Ph.D. Dissertation. U.C. Irvine. 2000. AAI9980887. (Available: http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm)

[39] SOAP Version 1.2 Part 1: Messaging Framework (Second Edition), W3C, 27 April 2007, http://www.w3.org/TR/soap12-part1/