Cloud Computing Adoption Framework – a security framework for business clouds

Victor Chang[1], Yen-Hung Kuo[2], Muthu Ramachandran[1]

1. School of Computing, Creative Technologies and Engineering,
Leeds Beckett University, Leeds, UK.
2. Data Analytics Technology & Applications, Institute for Information Industry, Taiwan, R.O.C.

Corresponding authors: V.I.Chang@leedsbeckett.ac.uk ; keh@iii.org.tw

Abstract

This paper presents a Cloud Computing Adoption Framework (CCAF) security suitable for business clouds. CCAF multi-layered security is based on the development and integration of three major security technologies: firewall, identity management and encryption based on the development of Enterprise File Sync and Share technologies. This paper presents our motivation, related work and our views on security framework. Core technologies have been explained in details and experiments were designed to demonstrate the robustness of the CCAF multi-layered security. In penetration testing, CCAF multi-layered security could detect and block 99.95% viruses and trojans and could maintain 85% and above of blocking for 100 hours of continuous attacks. Detection and blocking took less than 0.012 second per trojan and viruses. A full CCAF multi-layered security protection could block all SQL injection providing real protection to data. CCAF multi-layered security had 100% rate of not reporting false alarm. All F-measures for CCAF test results were 99.75% and above. How CCAF multi-layered security can blend with policy, real services and blend with business activities have been illustrated. Research contributions have been justified and CCAF multi-layered security can offer added value for volume, velocity and veracity for Big Data services operated in the Cloud.

Keywords: Cloud Computing Adoption Framework (CCAF); OpenStack; CCAF multi-layered security; security for business clouds

1. **Introduction**

Security, trust and privacy always remain challenges for organizations that adopt Cloud Computing and Big Data. While there are demands for businesses to move their data in the Cloud and centralize management for data centers, services and applications are designed to achieve cost savings and operational efficiencies. At the same time, system design and deployment based on current security practices should be enforced to ensure all data and services are security compliant with up-to-date patches and policies. A risk-based approach to the development of a security program that recognizes (and funds) appropriate controls will ensure that all users can be protected, and that data can be confidential, have integrity and be available.

Some researchers have adopted a framework approach to allow organizations to follow guidelines, policies and standards. For example, Zhang et al. [1] propose their Usage-Based Security Framework (UBSF) which can consolidate guidelines and policies with their framework, architecture and digital certificates. Takabi et al. [2] describe their comprehensive security framework by presenting a model to explain how to work with different service integrators and service providers. Zia and Zomaya [3] present their wireless sensor networks model with their algorithms and a software engineering approach. All these frameworks have provided recommendations on how they should be used. However, there are no details on how to actually use these proposals, and there are no clear evidence that these proposals can be adopted for Business Clouds, whereby the requirements include the ease of use, adaptability, best practice compliant and supported by large scale experiments such as penetration testing to validate robustness of such proposals [4-5]. Indeed, without such a clear "line on sight" between conception and implementation, such frameworks are unlikely to achieve operational status.

In order to meet requirements for Business Clouds, the Cloud Computing Adoption Framework (CCAF) has been developed to ensure that all implementations and service deliveries can meet all the technical challenges. There are real-life case studies to show how different Cloud Computing design, development and service delivery can meet both technical and organizational challenges. In the first example, CCAF was the framework used to develop Cloud storage and bioinformatics solutions for biomedical scientists based at Guy's Hospital and King's College London, UK [6]. The use of the framework ensured that the deliveries of storage services to back up thousands of medical data with terabytes of size. Bioinformatics services can simulate DNAs, proteins, genes, tumor and organs related to human bodies. The use of their security is limited to authentication, encryption and users with authorized access. In the second example, CCAF is used to provide guidelines for financial modeling, so that the best put and call prices can be computed in regard to the change of risks. Advanced computational techniques have been used to calculate risks and market volatility [7]. Security is limited to password authentication and users with authorized access and biometrics checks to the financial simulations. In the third example, investigations of hacking methods have been studied and made as part of prototype requirements. As a result of a user requirement and literature review, factors for successful implementations have been identified. All the collected and synthesized information have been instrumental in the development of CCAF Version 1.1, which emphasizes on the security policies, recommendations, techniques and technologies to be updated in our framework [8]. In these three examples, a more comprehensive Cloud security solution is required to ensure that services are robust and can be resistant to attack, hacking and unauthorized attempts to gain access. More experiments and simulations are required to validate the robustness and effectiveness of our security framework. This motivates us to consolidate our CCAF framework by providing a holistic approach involved with service integration, OpenStack security and multi-layered security to enhance security for business clouds. We propose an integrated security framework for business clouds to have the multi-layered security in place and have the large scale penetration testing and experiments to validate the robustness and effectiveness of our approach. All these proofs-of-concepts and lessons learned are important to Big Data in the Cloud as follows. First, it ensures that all the cloud services are safe and secure, including data coming in and out of the organizational data centers hosted on hundreds and thousands of virtual machines (VMs). Second, it ensures that large amount of data and large datasets can be processed and analyzed safely in the Cloud, which also explains why large scale penetration testing is necessary to validate the framework.

The breakdown of this paper is as follows. Section 2 presents the literature for security. Section 3 describes our core security technology for Enterprise File Sync and Share, including the architecture and layered components. Section 4 explains our multi-layered approach with core technologies and results from large scale experiments for penetration testing, SQL injection and data scanning. Section 5 illustrates topics of discussion and Section 6 sums up Conclusion and Future Work.

2. **Literature**

Different types of security frameworks are introduced as follows. First, Zhang et al. [1] propose their Usage-Based Security Framework (UBSF) for Collaborative Computing Systems. They explain their motivation, techniques behind, architecture and conditions for experiments. Usage decision of the USFB is based on subjects, objects, authorization, obligations and conditions. They explain how their model can work in collaborative ways supported by their literature and hypotheses. The usage-based authorization architecture uses sensors, directory service, policy decision point (PDP) and usage monitor (UM) to functions. Steps have been described to justify how the USFB can function effectively. To assist USFB, Zhang et al. [1]

have a prototype system architecture. They use OpenLDAP and OpenSSL to enforce security. They have three types of digital certificates: User, Attribute Repository (AR) and Resource Provider (RP) certificate. They explain how these certificates work in their workflow-type of security processes. They also adopt extensible access control markup language (XACML) to enforce policy specification which aligns with the UBSF approach for security. Ko et al [9] investigate trust for Cloud computing and propose a TrustCloud Framework focused on accountability. They have three layers: (1) System layer, which covers all the underlying hardware and platform; (2) Data layer, which contains the data for the work and (3) Workflow layer, which uses workflow to execute all the services and requests. There are two non-functional layers attached throughout the three layers. The first layer is Laws and Regulations, which can ensure all services follow the legal requirements in the country that the service is delivered. The second layer is Policies, which are the consolidated service level agreements and the best practice approach. This framework is considered a conceptual framework focused on the recommendations and best practice, since they do not have quantitative analyses, computational demonstrations and case studies. Pal et al. [10] present their proposed Cloud security that has emphasized on the architecture and steps of interactions between different services. They explain the role in each major user, their agents and all the fifteen steps involved. They use UML diagram to justify their approach and use architecture to explain relationship between the user, provider, proxy server, user agent and provider agent. They present their two algorithms and their experimental results. Using "Trust Value Updation", they validate their approach. However, their assumption is based on the probability of having a trusted user is 0.8 and a non-trusted user is 0.2. There are no any evidences to support this and they do not use any references or large number of surveys to justify their research. This also depends on their sample size, demographics and the country that the research is conducted. The NIST [11] framework provides a common language for establishing cybersecurity. The core NIST framework provides a set of activities on Identify, Protect, Detect, Respond, and Recover without more specific examples and case studies implementing a full security solution. However, our work on CCAF extends detailed activities and implementation on security for Cloud Computing and Big Data.

All these examples have security framework. However, those proposals do not demonstrate their contributions for business clouds. In other words, when businesses adopt Cloud Computing solution, they should be able to provide architecture, approaches for their framework, steps and experiments to support the robustness and validity of the framework. Our proposal on Cloud Computing Adoption Framework (CCAF) will provide details in core technologies in Section 3, and then the theoretical framework mapping to core technologies with experimental results to validate our framework will be shown in Section 4. Following that, key topics including security policy, business and security alignment, framework and core technology integration, relation of the big data in cloud, overall contributions with limitation are discussed in Section 5. Finally, research conclusion and future work will be in Section 6.

3. **Core technologies**

Prior to introduce the Cloud Computing Adoption Framework (CCAF), this section uses a concrete instance of the CCAF to be an example to explain CCAF core technologies and implementations. To hit the needs of moving big data in a semi-public business cloud, an enterprise cloud storage application - the semi-public Enterprise File Sync and Share (EFSS) service is chosen to be the CCAF instance to explain how CCAF protect enormous enterprise files (a kind of unstructured big data) in a business cloud environment.

To provide enterprises with the convenient cloud file sync and share service while taking enterprise concerns such as security, compliance, and regulation concerns into consideration, the cloud file sync and share service was been deployed by either on-premise or hybrid cloud model to target high value Enterprise File Sync and Share market [12-14]. Existing EFSS systems focus on system security and manageability, which encrypt data on transfer and at rest and also support system audit trail. As part of the core technologies of the CCAF framework, important EFSS security issues should be well addressed, particularly for businesses with critical data services. Following items describe the opened EFSS security issues.

(1) **Employee Privacy:** To prevent data leak, enterprise data are usually encrypted in existing EFSS systems. However, most existing EFSS systems only uses single master key to encrypt entire data space, which can prevent enterprise data leaks from outside but not from inside. For example, an EFSS system administrator (IT or MIS in enterprise) can spy enterprise sensitive data by self-granted authority.

(2) **Share Link:** The share link is widely adopted to share data to business partners who do not have an EFSS system account. From enterprise's perspective, the share link is convenient but not secure since it introduced new security loophole and might be used to leak data to unauthorized domain without leaving enough audit trail [15].

(3) **Cloud File Synchronization:** The natural of the cloud file synchronization is a security loophole to enterprises. It synchronizes shared and collaborative enterprise data from a managed EFSS service to employees' endpoint devices, and enterprises then have less/no control to the synchronized enterprise data. The synchronized enterprise data can then be distributed from the endpoint devices to other unauthorized domains by email, USB disk, and other communication interfaces available on the endpoint devices.

(4) **Enterprise Directory Integration:** To enable SSO (Single Sign On) in enterprise, most existing EFSS systems via direct network connection to integrate its authentication with existing enterprise directory, e.g., AD (Active Directory) and LDAP (Lightweight Directory Authentication Protocol). It also introduces two new security issues. Firstly, the EFSS system is able to access employees' profiles available in enterprise directories. Secondly, the EFSS system can log employees' credential information when authentication since every employee's username and password are pass through EFSS system to enterprise directory to conduct the actual authentication. Both cases provide EFSS system with chances to get authorized information.

An integrated security approach is introduced in follow sections, which implements several key components to form a scalable Secure EFSS system to address the enterprises' concerns by leveraging the on-premise OpenStack infrastructure [16]. EFSS has been integrated with our CCAF framework as an overarching model for cloud security for businesses.

Architecture and Design of the Integrated Security Approach

The key components of the Secure EFSS system are virtual appliances which can be provisioned and ran on the OpenStack compute service – Nova, and the data (e.g., metadata in database and uploaded enterprise files in user storage space) generated by the key components are stored in the OpenStack storage services – Cinder for block storage and Swift for object storage in which the OpenStack storage services are managed storage system controlled by the enterprise. The separation of the compute and storage makes the Secure EFSS system is scalable and more secure than existing EFSS systems.

All key components/virtual appliances including Load Balancer, Firewall, VFS (Virtual File System) Service, Directory Service, Log Service, MQ (Message Queue) Service, and the Database Service are provisioned from an all-in-one image stored in the OpenStack VM image management service – Glance to form the Secure EFSS system as shown in Figure 1. The VFS service of the proposed Secure EFSS system provides clients with a REST API set to manipulate a directory structure and files of a virtual file system. The backend of the virtual file system is the Database Service, which stores metadata to represent the directory structure and the nodes' attributes of the directory structure. Enterprise files uploaded by clients are leaf nodes of the directory structure, and they have a node attribute to point to encrypted objects in the Swift. Since there is no knowledge kept in the VFS service, the VFS service can dynamically provision more VFS VM to satisfy growing EFSS system usage in a scalable and distributed manner.

In Figure 1, rest key components' interactions and how they benefit to security and scalability are shown below. The Load Balancer is deployed in the DMZ (Demilitarized Zone), which dynamically distributes every request made by the Sandbox-based Cloud File Sync App to one of VFS VM for handling the requests by according to loads of the VFS VM for maximizing overall VFS Service performance. Moreover, a Firewall is deployed in between the Load Balancer and the VFS Service to form a secure deployment scheme which defends external direct network attacks for internal services. Once the VFS Service receives a request, it firstly sends the authentication information that came with the request to the Directory Service to check the request identity and authority. Subsequently, the VFS Service handles the request and logs related audit trails to the Log Service. When VFS Service is serving, it also reports overall service status including concurrent serving requests, CPU and memory usages, etc. to MQ service which actively calls OpenStack compute service API to provision and de-provision VFS VM to handle Secure EFSS system peak and idle situations.

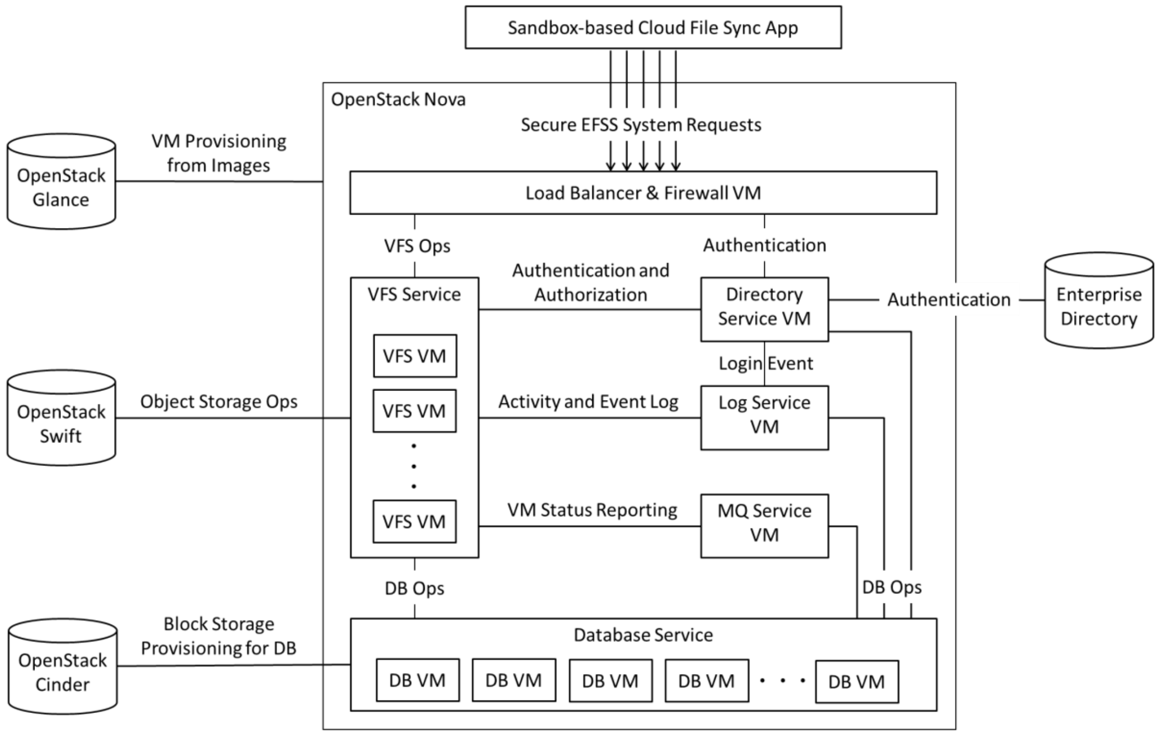


Figure 1. The Secure EFSS system architecture in the OpenStack.

Designs of key components of the proposed integrated security approach are introduced in following subsections which include User Storage Space Modeling, Distinct Share Link, Zero-knowledge Cloud Scale File Sharing System, Sandbox-based Cloud File Synchronization, Out-of-band Authentication, and a NoSQL adoption consideration.

A. User Storage Space Modeling

To make sure the system is scalable, metadata generated by key components especially by the VFS Service are stored into a MongoDB cluster. The MongoDB is a document based NoSQL database which sacrifices the relationship between documents to increase its scalability [17-18]. Further, we use user accounts as database sharding key to shard entire user storage space into smaller parts to mitigate the MongoDB scalability limitation which impacts database performance when there are too many documents inserted into a data collection. Following that, every user has own directory structure formed by metadata stored in own data collection, which physically isolates users' metadata, and every user's metadata then can be protected by owner's encryption key without worrying sensitive metadata being leaked when Database Service been hacked.

B. Distinct Share Link

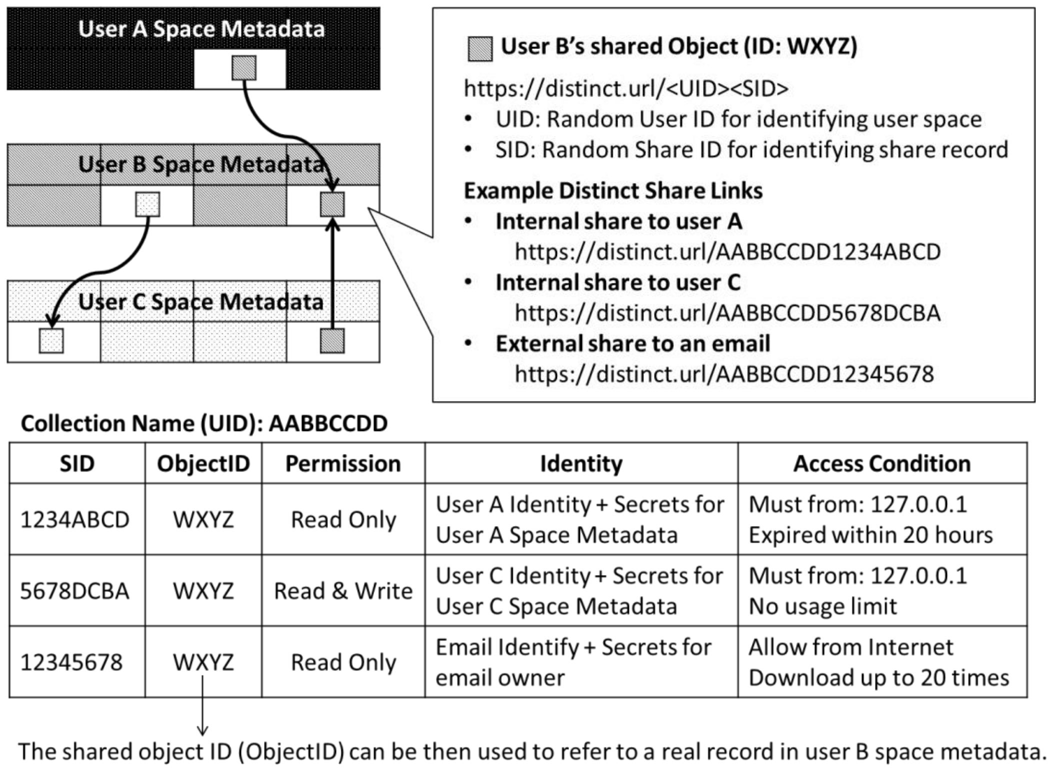
A share link provides conveniences for sharing cloud files in enterprise and external business partners, but it is insecure since web crawlers can find it by scanning emails and social network and then download it simply [19]. To address this issue, it introduces a utility named Distinct Share Link to secure sharing cloud files and to build a secure sharing relationship between two user storage spaces which are isolated data collections sharded from entire user storage space [20].

A Distinct Share Link is an additional layer attached with permissions, identities, and access conditions which encapsulates a share link, decreases diffusibility of the share link, adds traceability as well as controllability to the share link, and then be sent to recipients according to the attached identities. Whenever a recipient tries to access a Distinct Share Link, the recipient has to present his/her identities for access condition check. Once the recipient passes all checks, the Distinct Share Link layer then prepares an ephemeral representation to link to the share link for accessing. Since every Distinct Share Link access requires own identity (traceability) and controlled by permissions and access conditions (controllability), the diffusibility and convenience of the Distinct Share Link is decreased accordingly.

Compare to share link, always need to provide different identities when accessing a Distinct Share Link is inconvenient. To address the problem, it provides a recipient-defined identity function which allows a recipient to define own identity (e.g., password, secret, etc.) to each received Distinct Share Link. Hence, the recipient can define easy to remember identities to received Distinct Share Links. It also solves the share link identity management issue, that is, share links with different identities (e.g., passwords) are too much information to be remembered by a single recipient.

In addition to aforementioned benefits, the Distinct Share Link further support sandbox feature for creating a collaborative workspace with external business partners. In the collaborative workspace, every participant has own permission, and any inappropriate operation (e.g., delete all files) is only performed in a specific shared folder but not in entire personal user storage space. Note that with cloud storage capabilities, the performed inappropriate operations are able to be reverted to original, e.g., reverting previous cloud file version from file change history or reverting deleted shared files from the recycle bin.

The Distinct Share Link is also used to implement the internal cloud file sharing in the proposed Secure EFSS by building a secure sharing relationship between two isolated user storage spaces (data collections). Figure 2 depicts an example to explain how Distinct Share Link help build the internal cloud file sharing. In the top-left of the Figure 2, it represents two internal cloud file sharing relationships: (1) users A and C receive an object shared from user B and (2) user B receives an object shared from the user C. The callout box shown at the top-right of the Figure 2 further explains sharing relationships associated with the object (ID: WXYZ) in the former sharing relationship. It shows that three Distinct Share Links are used to share the object (ID: WXYZ) to recipients: user A, user B, and an email address. Moreover, all of the three Distinct Share Links are able to be parsed into three parts: (1) the service endpoint (e.g., https://distinct.url/), (2) the UID part for identifying user storage space as known as the location of the data collection, and (3) the SID part for identifying the metadata (as shown in the table of the Figure 2) which contains information about the sharing relationship in the located data collection. After retrieving the metadata, the "ObjectID" can be used to refer to either a shared object or a share link point to a shared object.



User B's shared Object (ID: WXYZ)

https://distinct.url/<UID><SID>

- UID: Random User ID for identifying user space
- SID: Random Share ID for identifying share record

Example Distinct Share Links

- **Internal share to user A**
  https://distinct.url/AABBCCDD1234ABCD
- **Internal share to user C**
  https://distinct.url/AABBCCDD5678DCBA
- **External share to an email**
  https://distinct.url/AABBCCDD12345678

Collection Name (UID): AABBCCDD

SID	ObjectID	Permission	Identity	Access Condition
1234ABCD	WXYZ	Read Only	User A Identity + Secrets for User A Space Metadata	Must from: 127.0.0.1 Expired within 20 hours
5678DCBA	WXYZ	Read & Write	User C Identity + Secrets for User C Space Metadata	Must from: 127.0.0.1 No usage limit
12345678	WXYZ	Read Only	Email Identify + Secrets for email owner	Allow from Internet Download up to 20 times

The shared object ID (ObjectID) can be then used to refer to a real record in user B space metadata.

Figure 2: Example for explaining distinct share link.

A forementioned external and internal sharing cases have similar Distinct Share Link creation and access processes. The key difference between the two cases is the identity delivery process. In the external sharing case, the identity is firstly defined by the Distinct Share Link creator, subsequently it is delivered by oral or identity itself (e.g., email), and finally it might be re-defined by the recipient. The internal sharing case automatically performs similar process by system with following steps: (1) generate random keys as identities in the sharing source, (2) encrypt generated identities by a recipient's public key, and (3) deliver the encrypted identities from the sharing source to the sharing destination for further usage.

C. Zero-knowledge Cloud Scale File Sharing System

All Cloud files are stream-encrypted by different random symmetric keys and stored in the Swift object storage when uploading to the proposed Secure EFSS system. This subsection will introduce a Zero-knowledge Cloud Scale File Sharing System which leverages introduced stacks including User Storage Space Modeling technique and Distinct Share Link. The Zero-knowledge Cloud Scale File Sharing System also introduces a new key cascading model [21-22] which uses in this work to manage the random symmetric keys crossing loose coupled domains and further to protect cloud files from internal/external unauthorized access attempts.

To develop a Zero-knowledge Cloud Scale File Sharing System, authors take following key enterprise security concerns and modern cloud system's scalability and manageability issues into consideration:

(1) **Zero-knowledge System:** Zero-knowledge means that a system knows nothing about things provided by users. The user provided things can be contents of cloud files, metadata of user storage spaces, and encryption keys including users' passwords used to protect cloud files and user storage spaces, and the last and the most important part is that it must apply to entire lifecycle of things starting from its first time appearing in the system to its end, that is, being permanently wiped from the system. To this end, the Zero-knowledge Cloud Scale File Sharing System is formed by stacking multiple distinct

domains with own services, and each of the distinct domain is serving as an additional complementary security layer to its one-level lower domain. As shown in Figure 3 and Figure 4, they represent stacking relationships of personal cloud file and shard cloud file situations respectively. In Figure 3, the lowest OpenStack Domain only provides cloud file storage capability, subsequently the one-level upper VFS Service Domain servers cloud file metadata management functions and adds additional stream cloud file and metadata encryption capabilities, and finally the top-level Directory Service Domain incorporates with User Domain to protect the encryption keys used in cloud file and metadata protections. In aforementioned stacking relationship, the worth to mention part is that the User Domain is not in cyberspace instead it is in users' brain, only users themselves know their own passwords, and it makes the zero-knowledge system work. Similarly, the Figure 4 further provides a sibling stacking relationship to the Figure 3 to isolate share participants' personal VFS Service Domains (see User1 and User2's distinct Personal Databases in Figure 4). It further makes the zero-knowledge system work in a multitenant cloud environment. Users do not need to worry about their cloud files when another user's storage space was conquered since that the sibling stacking relationship is a one-way relationship, and there is no way for a lower layer using existing knowledge to derive upper layer information. This zero-knowledge design also guarantees users have zero-knowledge with each other.

(2) **Security Compliance:** In addition to data encryption, enterprise security compliance also requires to support password history management capability. For example, an employee has to change its password every quarter, and the password cannot be equal to any password used in the last year. The Zero-knowledge Cloud Scale File Sharing System abstracts a Directory Service Domain to deal with the password history management requirement (as shown in the Figure 3 and Figure 4). Every time a user's password changed, the Directory Service Domain will automatically adjust its internal relationship to keep the stacking relationship be consist without bothering every lower domains. In other words, the Zero-knowledge Cloud Scale File Sharing System is also a security compliance friendly system.

(3) **Atomic Objects and Systems:** Considering the complex cloud environment, to make everything be atomic is a good idea. In the Zero-knowledge Cloud Scale File Sharing System, every service in the domains and every encrypted cloud file is an atomic unit which can be managed (e.g., create, update, and delete) without worried about dependencies with other atomic units. To achieve this, authors separate data from computation logic. As mentioned, metadata of a user are aggregated to form a personal database, and because of choosing MongoDB, the personal database is physically a set of manageable files stored in the block storage. In terms of cloud files, a cloud file itself is atomic and can be stored in object storage directly. However, once the cloud file be encrypted, to manage the encrypted cloud file as an atomic unit is difficult since the encryption key of the cloud file has to be managed as well. To keep it be atomic, it encrypts the encryption key of the cloud file, then combines the encrypted encryption key and the encrypted cloud file to form a manageable encrypted object to replace the original plain-text cloud file, and let the encryption key of the cloud file's encryption key be managed by the VFS Service Domain. As shown in the dot line block in the Figure 3 and Figure 4, the encrypted cloud file (*File*) and its encryption key (*File_Key*) is combined as an encrypted object stored in the OpenStack Swift object storage. The encrypted object is then an atomic unit which can then be migrated, backup, and managed by a cloud manner. Following that, rest parts are computation logics which can be categorized by domains (e.g., Directory Service and VFS Service Domains have their own computation logics), and different types of computation logics can be implemented in virtual machine or container images which can be managed by the OpenStack Glance image service as atomic units and be hosted on the OpenStack Nova cloud computing environment.

(4) **API Integration:** All domains in the Zero-knowledge Cloud Scale File Sharing System are integrated by REST API, which provides services in domains with flexibility by developing and growing themselves. With such a integration, every domain only need to focus on satisfying contracts and/or

SLA (Service Level Agreement) with other domains, e.g., keep 99.99% service availability or guarantee API calling 300 times per minutes. In addition to inter-domain integration, the VFS Service Domain also uses REST API to implement distinct user storage spaces' integration. Specifically, it uses Distinct Share Link to securely share one user storage space's cloud file to other user storage spaces (see the dot line arrow in the Figure 4). It forms a loose coupled sharing relationship which is especially suitable for cloud era.

Detailed designs and implementations of the Zero-knowledge Cloud Scale File Sharing System will be discussed next. The core technology integrates all above key considerations together is a key cascading model. Figure 3 and Figure 4 are used as examples to assist the follow-up explanations.

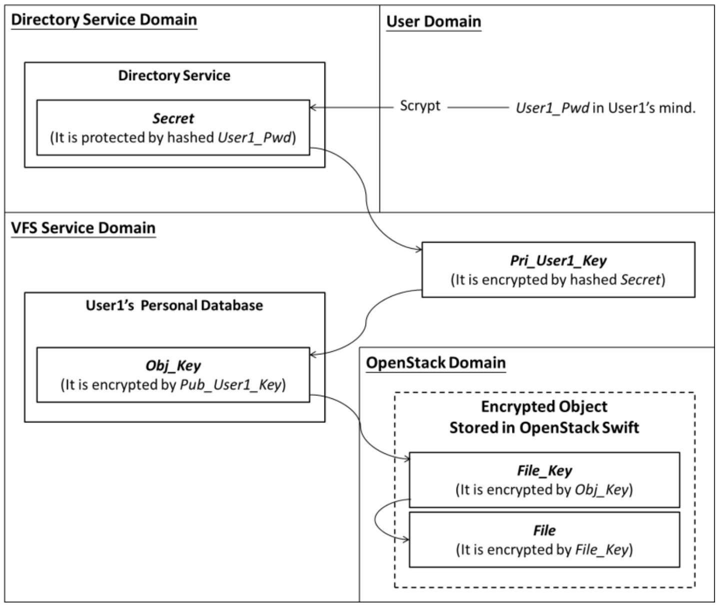

Directory Service Domain
User Domain
Directory Service
Secret
(It is protected by hashed User1_Pwd)
Scrypt
User1_Pwd in User1's mind.
VFS Service Domain
Pri_User1_Key
(It is encrypted by hashed Secret)
User1's Personal Database
Obj_Key
(It is encrypted by Pub_User1_Key)
OpenStack Domain
Encrypted Object
Stored in OpenStack Swift
File_Key
(It is encrypted by Obj_Key)
File
(It is encrypted by File_Key)

Figure 3: Personal file encryption key management concept

Figure 3 is the key management concept of the personal storage space, in which the solid line arrows are encryption/decryption directions. To explain the meaning of the encryption/decryption direction, two examples are shown in follow. In the first example (a symmetric key case), the "*Secret*" pointing to the "*Pri_User1_Key*" has two meanings: (1) the "*Pri_User1_Key*" is encrypted by the (hashed) "*Secret*" and (2) the "*Pri_User1_Key*" is able to be decrypted by the (hashed) "*Secret*". In this example, the "*Pri_User1_Key*" is the private key of the user1 and the "*Secret*" could be any information managed by the Directory Service. Note that keys in the key cascading model could like the "*Secret*" which is hashed before being used to encrypt other keys. In another example (an asymmetric key case), the "*Pri_User1_Key*" pointing to the "*Obj_Key*" has only one meaning which means the "*Obj_Key*" is able to be decrypted by the "*Pri_User1_Key*" since the "*Obj_Key*" is originally encrypted by the public key of the user1. Figure 4 is the key management concept of the shared storage space, in which the solid and dot line arrows have the same definition with the solid line arrows in the Figure 3.

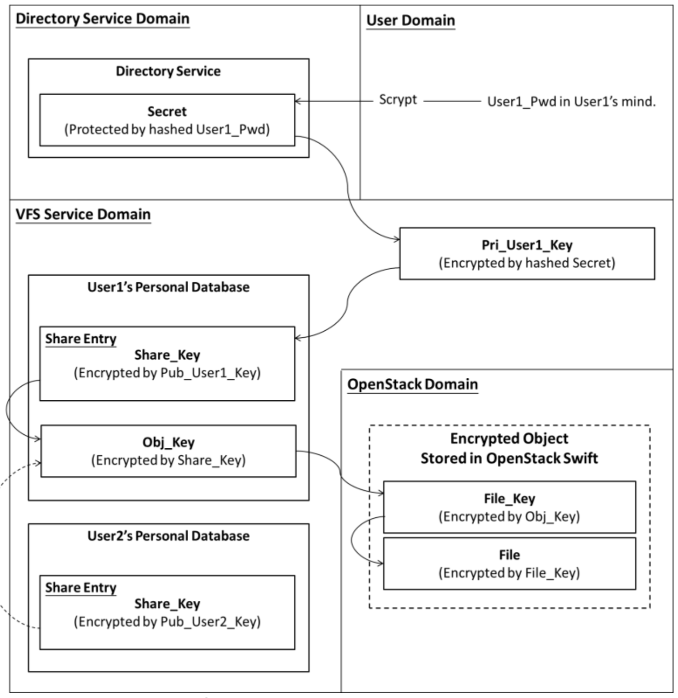


Figure 4: Shared file encryption key management concept

The key cascading relationship of the personal storage space as shown in Figure 3 crosses four distinct domains, which starts at the User Domain, passes through the Directory Service and VFS Service Domains, and finally ends in the OpenStack Domain. Starting at the tail of key cascading relationship, to obtain the plaintext "*File*", it requires the random symmetric key "*File_Key*" to decrypt the encrypted "*File*". The "*File_Key*" can be further be decrypted by another random symmetric key "*Obj_Key*" which is encrypted by user1's public key in the VFS Service Domain. To prevent the "*File_Key*" from exposing risk when sharing or changing ownership, the user1's public key does not directly encrypts the "*File_Key*", and instead it encrypts the "*Obj_Key*" which protects the "*File_Key*". Thus, the user1's public key can protect the "*File_Key*" indirectly, and the plaintext "*File_Key*" will be required only when trying to access the "*File*". With such a property, we can combine the encrypted "*File_Key*" and the encrypted "*File*" to form an encrypted object as known as an atomic unit to be stored in the Swift object storage, which is also beneficial to object migration, replication, and management.

Reconsidering the key cascading relationship, the user1's private key "*Pri_User1_Key*" is the entry of the VFS Service Domain which is formed by metadata stored in the user1's personal data collection. Since the entire VFS Service Domain may be conquered, it has to choose another key which is not available in the VFS Service Domain, to protect the user1's private key "*Pri_User1_Key*". The implementation chooses the "*Secret*" in the Directory Service Domain to protect the "*Pri_User1_Key*" because of the nature of authentication and authorization always happen prior to accessing the VFS Service Domain. The best practice of choosing suitable information form the Directory Service Domain to be the "*Secret*" is to choose the information which changing frequency is conform to enterprise compliances since the user1's private key "*Pri_User1_Key*" has to be re-encrypted along with the changing of the "*Secret*". Finally, the user1's password is used to protect the "*Secret*". In the key cascading model, the user1's password is the only information not available in the cyberspace, and the hash of the password has already qualified to be used to protect the "*Secret*". However, in our implementation, the password is hashed by the Scrypt [23] which generates different hashed values every time, and it results in that we need to choose another suitable "*Secret*" which is associated with password.

Contrast to the key management concept of the personal storage space, the key management concept of the shared storage space as shown in Figure 4 adds another key cascading relationship between sharing participants' private keys and object key of the shared object. To add another key cascading relationship among sharing participants' data collections, a symmetric key "*Share_Key*" is inserted in between the participants' private keys (e.g., the "*Pri_User1_Key*" and the user2's private key) and the "*Obj_Key*". Through the Distinct Share Link as the dot line arrow shown in Figure 4, the user2 is then able to use the "*Share_Key*" as part of the identity to access the user1's shared objects (e.g., the "*File*").

To formally represent aforementioned explanations, the key management algorithms for sharing a personal file and stopping a share are described in Figure 5 and Figure 6 respectively.

```
The key management algorithm for sharing a personal file:

STEP 1. Obtain Obj_Key:
  // The User_Pwd is the user's password.
  Secret ← User_Pwd (encrypted Secret);
  // The Pri_User_Key is the user's private key.
  Pri_User_Key ← Secret (encrypted Pri_User_Key);
  Obj_Key ← Pri_User_Key (encrypted Obj_Key);

STEP 2. Obtain Share_Key:
  IF Share_Key for the share does not exist THEN
    // Generate a new Share_Key.
    Share_Key ← Share_Key_Gen();

STEP 3. Encrypt Obj_Key by Share_Key:
  encrypted Obj_Key ← Share_Key (Obj_Key);
  store the encrypted Obj_Key into personal database to replace the
  encrypted Obj_Key in SETP 1;

STEP 4. Encrypt Share_Key by Public Key:
  // The Pub_User_Key is the user public key.
  encrypted Share_Key ← Pub_User_Key (Share_Key);
  SWITCH (identity)
    CASE 'owner':
      store the encrypted Share_Key into personal database with
      the encrypted Obj_Key as the share entry;
    CASE 'recipient':
      store the encrypted Share_Key into personal database as the
      share entry;
  END
```

Figure 5: Key management algorithm for sharing a personal file

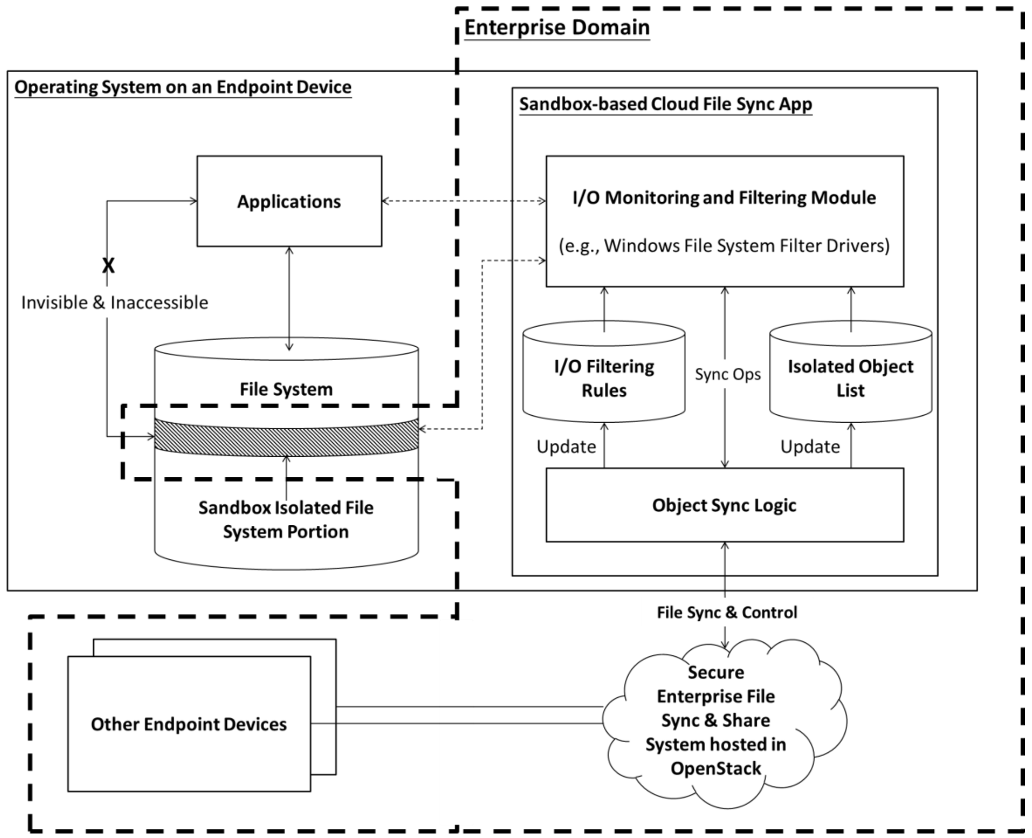
```
    The key management algorithm for stopping a share:

STEP 1. Obtain Obj_Key:
  Secret ← User_Pwd (encrypted Secret);
  Pri_User_Key ← Secret (encrypted Pri_User_Key);
  Share_Key ← Pri_User_Key (encrypted Share_Key);
  Obj_Key ← Share_Key (encrypted Obj_Key);

STEP 2. Encrypt Obj_Key by Pub_User_Key:
  encrypted Obj_Key ← Pub_User_Key (Obj_Key);
  store the encrypted Obj_Key into personal database to replace
  the encrypted Obj_Key in SETP 1;

STEP 3. Remove Share_Key:
  remove corresponding Share_Key from owner's personal
  database;
  /*
     Note that recipients' Share_Key are going to be removed
     asynchronously when trying to access the missing share entry.
  */
```

Figure 6: Key management algorithm for stopping a share

D. Sandbox-based Cloud File Synchronization

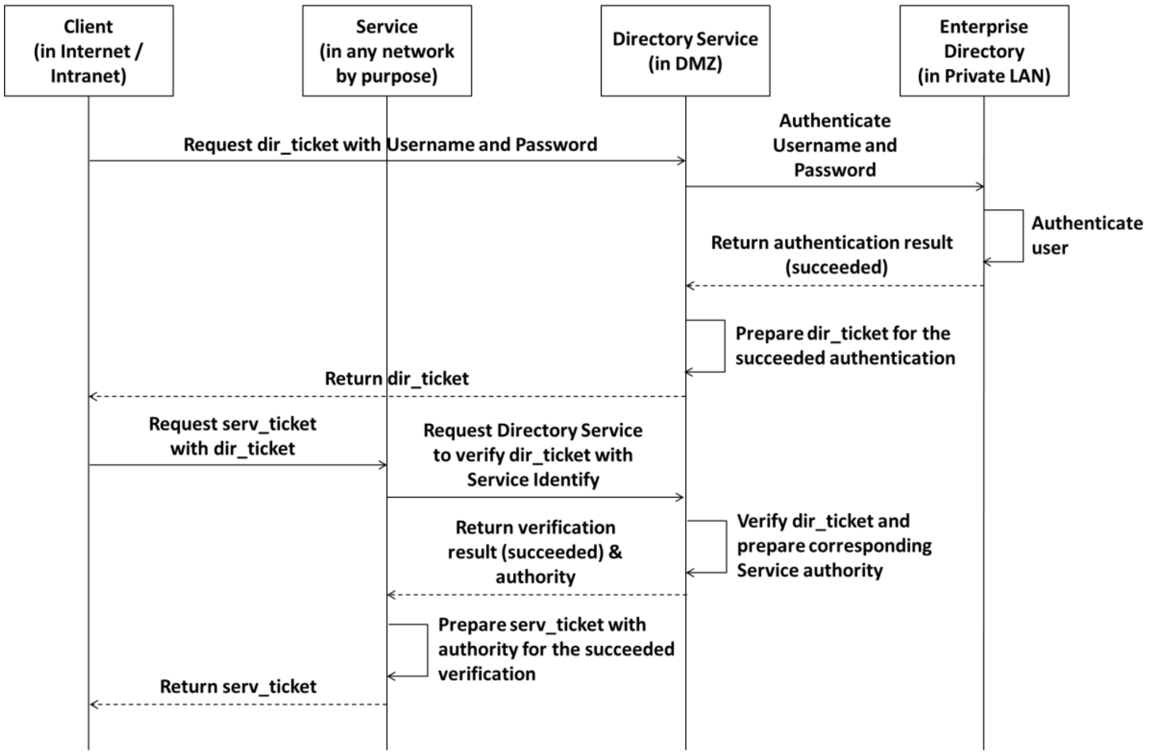
The EFSS service is convenient for collaboration in enterprise. However, most IT and MIS dislike to provide EFSS service in enterprise since that the once a cloud file being shared and synchronized with employee's client device, the file is no longer under enterprise's control. To provide an integrated Secure EFSS approach, it proposed a **Sandbox-based Cloud File Sync App** to solve the issue [24-25]. Differ from traditional cloud file synchronization tool, the Sandbox-based Cloud File Sync App creates a managed sandbox in the operating system's (O/S) file system and a local sync folder inside the sandbox, and then synchronizes cloud file between the local sync folder and user storage space in the Secure EFSS system. Since only authorized file system operations can be applied to the sandbox, the O/S and applications executed on top of the O/S are unable to read or update the sandbox. Such a property let the Sandbox-based Cloud File Sync App be able to actively prevent cloud files to be leaked from any employee's client device. The dot line area in the Figure 7 is the Enterprise Domain which is extended from the original Secure EFSS system to every enterprise employee's client device.

The core of the Sandbox-based Cloud File Sync App is the I/O Monitoring and Filtering Module which is a file system filter as shown in Figure 7. The I/O Monitoring and Filtering Module refer to Isolated Object List and I/O Filtering Rules to monitor and filter file system operations trying to apply to files and folders in the sandbox. There is a list of file and folder paths in the Isolated Object List to form the Sandbox Isolated File System Portion in the file system. Moreover, the list of file and folder paths is dynamically updated whenever the Object Sync Logic syncing files and folders between the local sync folder and the Secure EFSS system. To further make the Sandbox-based Cloud File Sync App an enterprise compliance facilitating tool, the I/O Filtering Rules provides enterprise IT and MIS with a capability to update its I/O filtering rules to control the local synchronized files through controls made by the Secure EFSS system.


Enterprise Domain
Operating System on an Endpoint Device
Sandbox-based Cloud File Sync App
Applications
I/O Monitoring and Filtering Module
(e.g., Windows File System Filter Drivers)
X
Invisible & Inaccessible
File System
I/O Filtering Rules
Sync Ops
Isolated Object List
Update
Update
Sandbox Isolated File System Portion
Object Sync Logic
File Sync & Control
Other Endpoint Devices
Secure Enterprise File Sync & Share System hosted in OpenStack
Figure 7: Sandbox-based cloud file synchronization

E. Out-of-band Authentication

As mentioned, directly integrating EFSS system with enterprise directory for enabling single sign on is insecure since sensitive information may be accessed or logged by the EFSS system. To address the issue, it introduces an out-of-band authentication method as parts of the integrated security approach, and the sequence diagram of the authentication method is shown in Figure 8. To take secure deployment requirement into account, the participants involved in the authentication process are deployed in different networks: (1) the Client can access the Service from either Internet or Intranet, (2) the Service is deployed in either DMZ or enterprise private LAN by according to purpose of the Service, (3) the Directory Service has to be deployed in DMZ for handling authentication requests from both Internet and Intranet, and (4) behind the firewall, the **Enterprise Directory** (e.g., AD or LDAP) is protected in the enterprise private LAN. The security deployment let the Directory Service act as a directory proxy to receive and then bypass the authentication requests to the Enterprise Directory for actual authentication. To let such a model work, the source of the Directory Service has to be examined and verified by the enterprise to get trust. Once the Directory Service passes the trustworthiness verification, the Service is the only untrustworthy participant in the authentication model, that is, the proposed VFS Service is untrustworthy.


Client (in Internet / Intranet)
Service (in any network by purpose)
Directory Service (in DMZ)
Enterprise Directory (in Private LAN)
Request dir_ticket with Username and Password
Authenticate Username and Password
Authenticate user
Return authentication result (succeeded)
Prepare dir_ticket for the succeeded authentication
Return dir_ticket
Request serv_ticket with dir_ticket
Request Directory Service to verify dir_ticket with Service Identify
Verify dir_ticket and prepare corresponding Service authority
Return verification result (succeeded) & authority
Prepare serv_ticket with authority for the succeeded verification
Return serv_ticket
Figure 8: Sequence diagram for illustrating out-of-band authentication.

To access the Service, the Client has to verify its credential (e.g., username and password) with the Enterprise Directory. However, with the secure deployment as shown in Figure 8, the Client has to authenticate with the Directory Service to verify is credential instead. After pass the check, the Directory Service will prepare a directory ticket (*dir_ticket*) to the Client for further exchanging a service ticket (*serv_ticket*) with the Service. Following that, the Client can access the Service by the service ticket, and the Service will then serve the Client based on the authority information provided by the Directory Service. To sum up, the proposed out-of-band authentication method has following advantages:

(1) **Security:** Login information (the username and password) only goes through managed trustworthy services, e.g., Directory Service and Enterprise Directory. Furthermore, the Enterprise Directory does not connect to any untrustworthy service (e.g., the VFS Service), which also prevents sensitive information stored in the Directory Service to be accessed by untrustworthy services.

(2) **Flexibility:** The proposed secure deployment is suitable for private and hybrid cloud services. Besides, services' proprietary requirements can be implemented in the Directory Service as an extension to the existing Enterprise Directory. For instance, to support additional authentication protocol like OpenID, the OpenID extension modules can be implemented in the Directory Service without bothering the Enterprise Directory.

(3) **Single-Sign-On (SSO):** The Directory Service can serve multiple services simultaneously. By the proposed out-of-band authentication method, the Client could have SSO feature among those services.

F. NoSQL Adoption

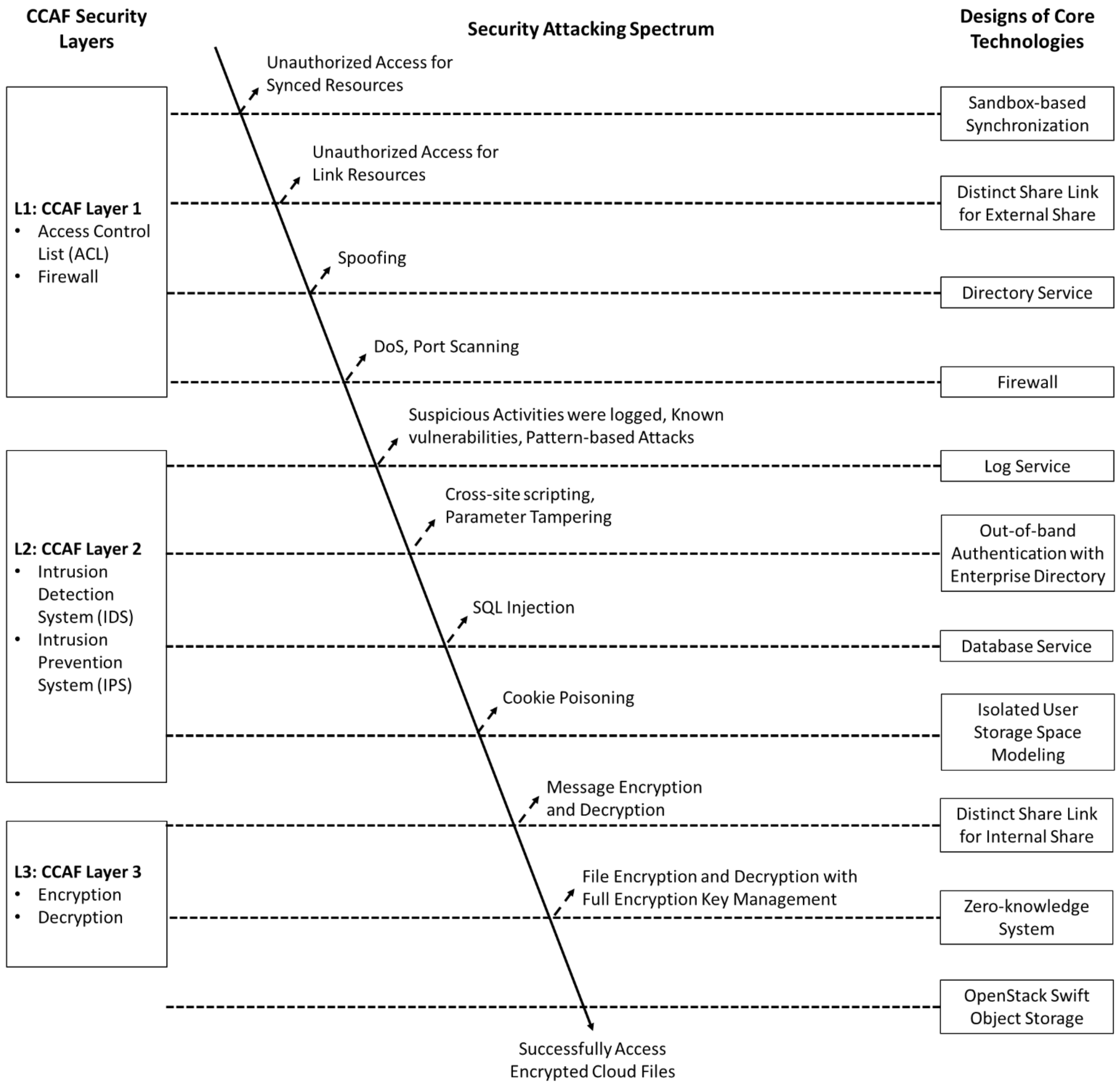
NoSQL plays important role in our security framework. Business clouds must be robust enough and not resistant to unauthorized attacks such as SQL injection, a commonly used technique to bring down database-driven websites. NoSQL database is less and not vulnerabilities. In order to test robustness, a NoSQL database, MongoDB and MySQL are used and then compared in performance under the penetration testing environments, which will be described in the next section. MongoDB is a document store database, in which data is stored in documents instead of rows and columns. It is normally stored in Javascript Object Notation (JSON) and Javascript is the language to query data. There are advantages of using MongoDB. First, developers can modify the schema dynamically without causing downtime. In other words, less time can be spent on preparing data for the database and more effort can be spent on making the data to work. Secondly, MongoDB can provide large scalability and allow thousands of transactions to scale up and down with ease.

4. **Multi-layered security**

This section describes our multi-layered security approach for the CCAF security framework. Our work on software security engineering [7] has motivated to develop a complete framework for cloud security which is known as Cloud Computing Adoption Framework [26]. The CCAF consists of a systematic process for building security, supporting cloud application development life cycle, right from requirements, design, implementation, and testing. The work described in Section 3 has been successfully integrated as the CCAF multi-layered security which has a three layer implementation model for cloud security as shown in Figure 9. This model can be applied to both private and public cloud delivery models. The cloud security layers are:

- Access Control and Firewall (L1) which provides support for password protection, access control mechanism and firewall protection
- Intrusion detection system (IDS) and intrusion prevention system (IPS) (L2) which aim is to detect attack, intrusion and penetration, and also provide up-to-date technologies to prevent attacks such as DoS, anti-spoofing, port scanning, known vulnerabilities, pattern-based attacks, parameter tampering, cross site scripting, SQL injection and cookie poisoning. Identity management is enforced to ensure the right users can access confidential data.
- Encryption/Decryption Control Layer (L3) which provides support for encryption and decryption of files, messages, and including security controls. This feature monitors and provides early warning as soon as the behavior of the fine-grained entity starts to behave abnormally; and end-to-end continuous assurance which includes the investigation and remediation after an abnormality is detected.

Each layer work in a recursive relationship to pass on security control to each other for extra validity and verification of security techniques that are implemented. This can ensure reduction in the infections by trojans, virus, worms, and unsolicited hacking and denial of service attacks. Each layer has its own protection and is in charge of one or multiple duties in the protection, preventive measurement and quarantine action presented in Figure 9. The left-hand side of Figure 9 shows the CCAF three layers of security and functionality. In layer 1 (L1), it can prevent unauthorized access for synced resources and link resources, spoofing, denial of services (DoS) and port scanning. In layer 2 (L2), it can prevent and counter actions for suspicious activities, known vulnerabilities, patterned-based attacks, cross-site scripting, SQL injection and cookie poisoning. In layer 3 (L3), it offers three types of encryption and decryption services: message, file and full. The right-hand side of Figure 9 explains how designs of core technologies in Section 3 are relevant to the CCAF security. In L1, it includes sandbox-based synchronization, distinct share link for external share, Directory Service and Firewall. In L2, it includes Log Service, Out-of-Band Authentication with Enterprise Security, Database service and isolated user storage space for modeling. In L3, it includes Distinct Share link for Internal Share and Zero-knowledge System. Once a cloud file access request passed all the three security layers, it can reach the managed Openstack Swift object storage, and then get the requested cloud file. How each preventive function of the CCAF security on the left-hand side can be matched to its respective system designs on the right-hand side is illustrated in of Figure 9.



CCAF Security Layers
Security Attacking Spectrum
Designs of Core Technologies
L1: CCAF Layer 1
• Access Control List (ACL)
• Firewall
L2: CCAF Layer 2
• Intrusion Detection System (IDS)
• Intrusion Prevention System (IPS)
L3: CCAF Layer 3
• Encryption
• Decryption
Unauthorized Access for Synced Resources
Unauthorized Access for Link Resources
Spoofing
DoS, Port Scanning
Suspicious Activities were logged, Known vulnerabilities, Pattern-based Attacks
Cross-site scripting, Parameter Tampering
SQL Injection
Cookie Poisoning
Message Encryption and Decryption
File Encryption and Decryption with Full Encryption Key Management
Successfully Access Encrypted Cloud Files
Sandbox-based Synchronization
Distinct Share Link for External Share
Directory Service
Firewall
Log Service
Out-of-band Authentication with Enterprise Directory
Database Service
Isolated User Storage Space Modeling
Distinct Share Link for Internal Share
Zero-knowledge System
OpenStack Swift Object Storage

Figure 9: The three-layered CCAF security

With regard to the technical descriptions in Section 3, the Out-of-Band Authentication belongs to the second layer since it has a single sign on and has the Directory Service and Enterprise Directory and then OpenID. The authentication on the OpenID can double check the right identity of the users. Upon successful verification, single sign on is activated and then all data and all services can undertake encryption as the third layer. The first layer of security is provided by a login feature to a network and IP-based firewall protocols to enforce security. Access control is achieved by providing a list of users and giving them different types of user privilege according to their roles. With regard to the system design, our multi-layered approach consists of listed three items, and the full connections mapping between each CCAF security layer and core technologies used by the system design can further refer to Table 1:

- Layer 1: Password protection, network and IP-based firewall and access control
- Layer 2: Out-of-Band Authentication and OpenID serving for identity management
- Layer 3: Encryption and decryption to ensure the true and authenticated files can be archived through encryption. Any files to be accessed will require the algorithms in Figures 5 and 6.

Table 1 shows the detailed maps between the CCAF security with its component described in Section 3. In L1, it includes access control, password protection and firewall to minimize data leakage, authorized access, secure deployment for internal users and careful shared resources with external users. In L2, it includes intrusion detection and prevention and then identify management to detect suspicious activities, report vulnerabilities, strengthen identity management, adopt NoSQL databases and isolate the user storage modeling. In L3, it offers file and message encryption and decryption to ensure all data and passwords are safe and well-protected.

Table 1: A mapping between CCAF security layers and corresponding core technologies

Multi-layered CCAF		Core Technology	
Layer	Capability	Component / Mechanism	Targeting Security Use Cases
L1	Access Control	Sandbox-based Synchronization	Prevent data to be leaked to unauthorized domain by limiting file system access at endpoint devices.
L1	Password	Distinct Share Link for External Share	Shared resources only present to external users who passed the password checking.
L1	Access Control		Only authorized authorities can be applied to shared resources.
L1	Password	Directory Service	Anti-spoofing by only allowing registered user to login into system.
L1	Access Control		By individual internal services, only grant own authorized authorities.
L1	Firewall	Firewall	Support secure deployment to defend external direct network attacks (such as DoS and Port Scanning) for internal services.
L2	Intrusion Detection	Log Service	As a part of Intrusion Detection System, Log Service provides audit trail capability to log every activities and events happened in the system for further investigation. It is useful for detecting suspicious activities which trying to abuse known vulnerabilities and/or conducting Pattern-based Attacks.
L2	Identity Management	Out-of-band Authentication	Incorporate with Directory Service and Enterprise Directory to support the Identity Management capability especially the single-sign-on service with all internal services. The Cross-site scripting and Parameter Tamping were ended here.
L2	Intrusion Prevention	Database Service	The adoption of the NoSQL database (MongoDB) provides sort of Intrusion Prevention capability. A SQL Injection testing results of the MongoDB is going to be provided and discussed in later sections.

L2	Intrusion Prevention	Isolated User Storage Space Modeling	Once one of personal user storage space was conquered (by such like the Cookie Poisoning), the Isolated User Storage Space Modeling still guarantees other users' data privacy since all user storage spaces were modeled in own physical separated MongoDB data collections.
L3	Message Encryption / Decryption	Distinct Share Link for Internal Share	Messages for accessing internal Distinct Share Link were encrypted by the recipient's public key before sending, which also prevents network packet inspecting from internal Malware on conquered services.
L3	File Encryption / Decryption	Zero-knowledge System	Cloud files were encrypted by different random encryption keys, and the encryption keys were managed by the Zero-knowledge System which knows nothing about decrypting the cloud files. Only users with correct Password (L1), right Identity (L2) in the Zero-knowledge System are able to decrypt the original cloud files (L3).

4A Penetration testing

Penetration testing is commonly used in ethical hacking to validate the robustness of the proposed security prototype and test any vulnerabilities. Weakest points and vulnerabilities can be detected by the end of the penetrating testing. Several tools and techniques such as Metasploits were used to enhance penetration testing, upon successful penetration, known 2012 viruses and trojans were used to test the robustness of the multi-layered security upon. All tests were conducted in the VMs. Comparisons with single-layered McAfee anti-viruses software were undertaken to identify any performance issue such as the number of viruses and trojans blocked and the percentage of blockage. The following information was recorded during the experiments:

- The number of viruses and trojans detected and blocked by each layer.
- The total numbers of viruses and trojans detected and blocked by the system.
- The number of viruses and trojans detected but unable to be blocked and sent to quarantine.
- In the quarantine, the number of viruses and trojans that can be destroyed.
- In the quarantine, the number of viruses and trojans that cannot be destroyed.

Penetration testing for multi-layered CCAF security were conducted. All the trojans and viruses will need to pass the three layers. Each layer can detect and filter out suspected malicious files and then move to the quarantine area waiting for the next action. With the consolidated defense, all the layers should detect and trap as many viruses and trojans as possible. The purpose of the penetration tests is to identify whether that is the case and the number of true vulnerabilities that the multi-layered CCAF security can detect, which are important for business clouds. Two types of experiments were conducted. The first one was focused on penetration tests involved with injecting 10,000 viruses and trojans in one single attempt. The second one was focused on continuous penetration test, such as injecting 10,000 same viruses and trojans every five hours to test that the robustness of the CCAF solution.

Figure 10 shows the number of viruses and trojans detected and blocked by the multi-layered CCAF security. 4,650 viruses and trojans have been detected and blocked by the firewall. Another 4,444 viruses and trojans have been detected and blocked by identity management and intrusion prevention systems.

Finally, 901 were detected and filtered out by encryption, the last step to identify what files in disguise can be detected. Amongst all these figures, there are remaining 5 viruses and trojans sent to quarantine that they could not be removed or destroyed.

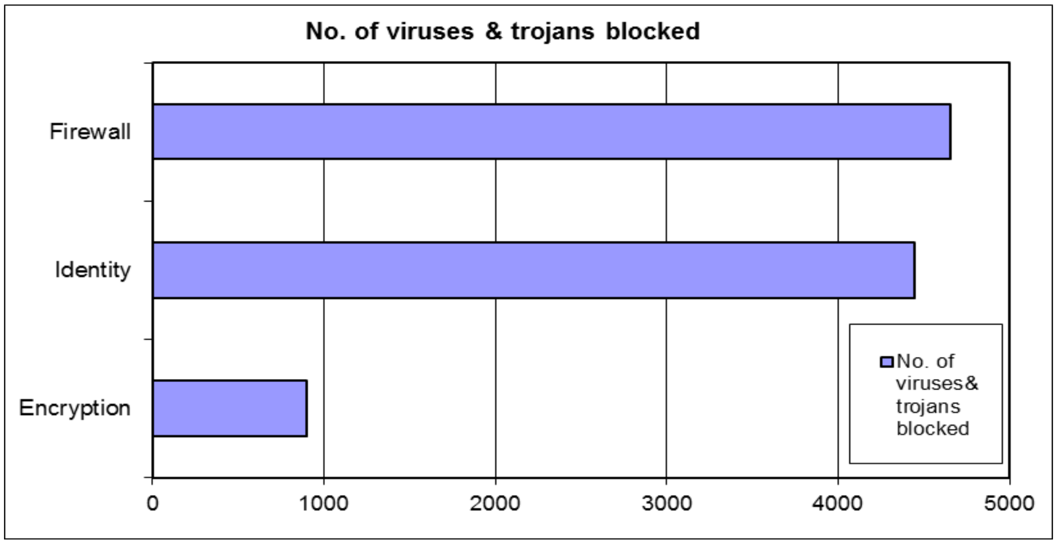


Figure 10: The number of viruses and trojans detected and blocked

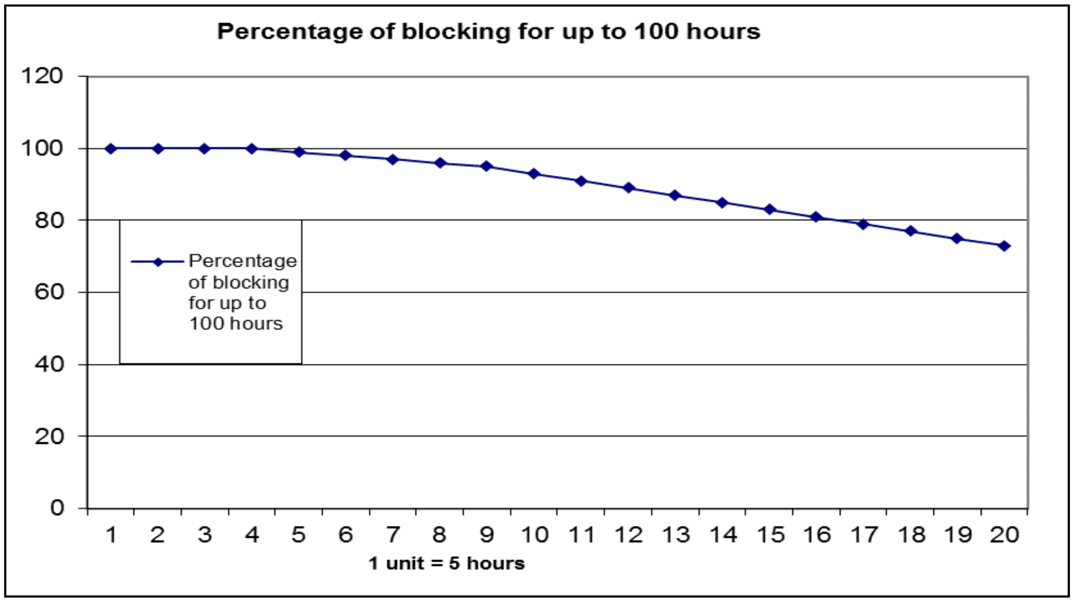


Figure 11: Percentage of blocking

Results are presented in percentages rather than the number of viruses and trojans blocked. The 10,000 same viruses and trojans are injected every five hours to test how the Data Center can cope with the vulnerabilities. Results in Figure 11 show the percentage that viruses and trojans that have been blocked, which dropped from 100% to 73% in 100 hours. It means that CCAF multi-layered security can withstand regular attacks for up to 5 for 99% blocking and above and up to 10 hours for 90% blocking and above. Performances become degraded after 10 hours of constant attack.

The execution time for detecting viruses and trojans and blocking viruses and trojans at each layer were recorded. This is important for the organizations that adopt Cloud computing since all vulnerabilities should be detected and blocked as soon as possible, or damages can be done to the organizations' systems and services. Figure 12 shows the execution time to detect and block in each layer. The firewall layer took the shortest time since it was designed to detect and block. Identity layer took the longest for above 200 seconds to detect and 451 seconds to block. The main reason was that a verification process was required in the identity layer to ensure that suspected files are viruses and trojans. As the name implied, identity layer required a verification process to ensure all files were safe and not in the suspected list. That was why it took more time for the verification to complete. Once the trojans and viruses were detected, they were blocked and sent to quarantine area for isolation. The reason why encryption layer had lower execution time was because by the time trojans and viruses came to the third layer, there were only 901 viruses and trojans left.

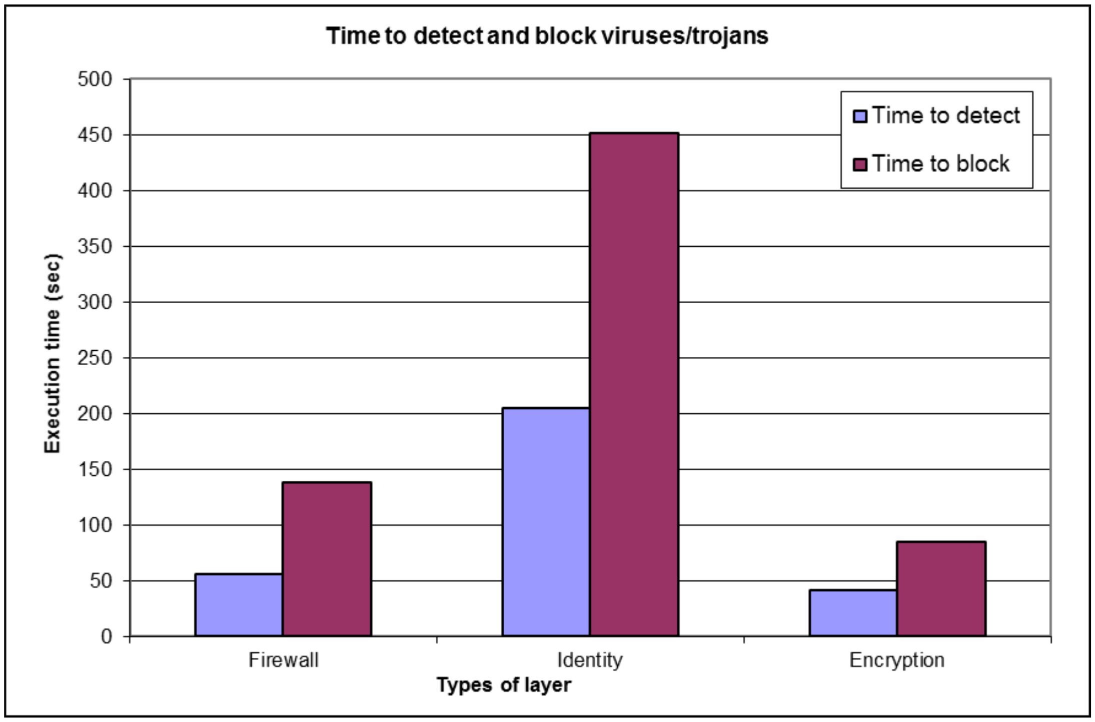


Figure 12: Execution time to detect and block viruses and trojans

As discussed earlier, 4,650, 4,444 and 901 viruses and trojans had been detected and blocked by the firewall, identity management and encryption respectively. Since the execution time detected and blocked was presented by Figure 12, the next action is to investigate the number of detects/blocks per second, which can be calculated by the total execution time in each layer divided by the number of trojans and viruses detected and blocked. Figure 13 shows the results. Number of blocks/detects are the lowest for the firewall due to its design and emphasis in security. Number of detects/blocks per second is similar between identity and encryption. However, reasons behind are different. The verification process required by the identity layer explains why some extra time is needed. When suspected files came to the third layer, some files had successfully disguised themselves. The encryption security used by CCAF and OpenStack could identity those malicious files in disguised. Key technologies are based on Encryption key management as described in Section 3C. The malicious files went into the encryption layer can be further shared to a VM/Sandbox environment to test. Once the test completed and reported, the VM/Sandbox can be directly destroyed.

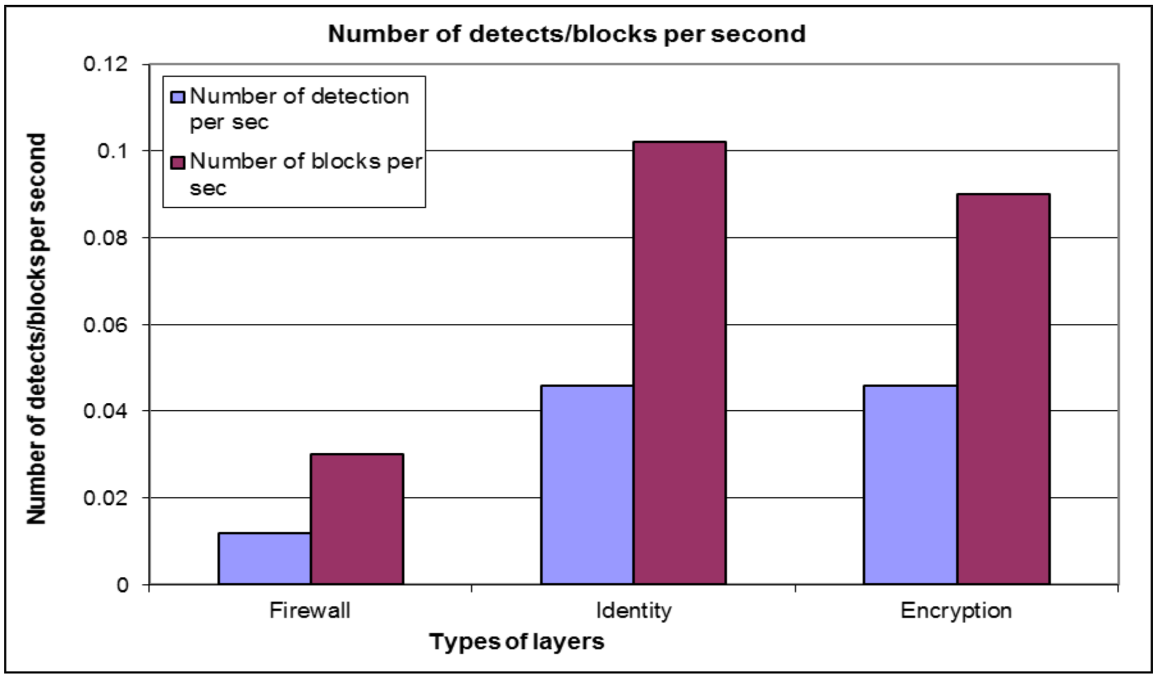


Figure 13: Number of detects/blocks per second

Antunes and Vieira [27] use four types of tools for penetration testing, explain the use of precision, recall and F-measure to justify the validity of their results.

Precision is the ratio of correctly detected vulnerabilities to the number of all detected vulnerabilities:

$$precision = \frac{t_p}{t_p + f_p} \qquad (1)$$

Recall is the ratio of true vulnerabilities detected to the number of known vulnerabilities:

$$recall = \frac{t_p}{t_v} \qquad (2)$$

where:

- True positive (t_p) refer to the number of true vulnerabilities detected;
- False positives (f_p) refer to the number of vulnerabilities detected but do not exist.
- True vulnerabilities (t_v) refer to the total number of vulnerabilities detected in penetration tests.

F-measure can be presented in terms of precision and recall as follows:

$$F\text{-}Measure = \frac{2 \times precision \times recall}{precision + recall} \qquad (3)$$

Services that can generate a high F-measure mean they are better services [27]. If a service obtains a precision of 0.6 means it can detect vulnerability with 60%. A recall of 0.7 means 70% of the known vulnerabilities is detected. While using formula (3), F-measure is equal to 0.646. The combination of precision, recall and F-measure can determine the quality of the security services. We reproduce the experiments conducted by [27] and then compare results of CCAF multi-layered security with VS1, VS2, VS3 and VS4 tools due to similarities with CCAF technologies except each is single layered security. Results in Table 2 show that the CCAF multi-layered security can provide a much better service since all the true vulnerabilities can be detected with precision as 1. Since only 5 out of 10,000 are missed, the recall is 0.995, resulting in F-measure as 0.9975, which are above all the test results. CCAF multi-layered security has better results than other four tools and has very high precision, recall and F-measure values, indicating there is a high extent of robustness and validity on security. See Table 2.

Table 2: Comparison between CCAF and other single-layered services

Services	Precision	Recall	F-Measure
CCAF	**1**	**0.995**	**0.9975**
VS1	0.455	0.323	0.378
VS2	0.388	0.241	0.297
VS3	1	0.019	0.037
VS4	0.567	0.241	0.338

4B SQL injections

Two types of databases were used for multi-layered security. The first type is a MySQL Server 5.5 that stored database information for users, customers, items and text-related information. A NoSQL database – the 64-bit MongoDB 3.0.4 storing the same information. The purpose of this ethical hacking is to test robustness of the two databases. This is useful for business clouds where enterprise security framework such as CCAF should be able to retain vital information and prevent unauthorized access or server outage due to attacks. SQL injections are common in unauthorized attacks and are used to perform ethical hacking. There are two ways to improve on the impacts on SQL injection. First, queries for SQL injection will also create "infinite" loops, in a way to make the database in an erroneous status. Second, 10,000 SQL statements were dumped every second to ensure that SQL injection prevention function cannot be rectified on time. Combining both methods, it makes the SQL injection being able to bring the system down or offline. To follow the SQL injection more effectively, steps were followed and codes were modified based on suggestions of [28], which has clearly indicated how to perform SQL injection successfully, which is still functional on MySQL 5.5. Three types of tests were conducted for MySQL.

- The first one was using SQL injection with a standard McAfee 2014 edition protection due to the availability of this product.
- The second test was using SQL injection with generic CCAF multi-layered security.
- The third test was using SQL injection with full CCAF multi-layered security protection that blocks all ports for MySQL, meaning that each SQL query will require authorization and verification.

Tests were undertaken for the first hour followed by the subsequent 24 hours. The first hour was the most important since the information can be leaked and used to hack other part of the systems [29]. The subsequent tests on the following 24 hours were taken. The reason is if a database has been compromised and no remedy actions have been taken within a reasonable time framewortk such as a maximum of 24 hours, it will pose threats to the organizations. Under the laboratory conditions, the first hour test and then the subsequent 24 hours tests have been performed for both MySQL and MongoDB. The 1,000 seconds per a simple response means that the database has been compromised.

4B.1 The first hour tests

Response time to MySQL server was recorded to show the impact of the SQL injection. If no SQL injection was done, the normal response time was **0.20 second** per simple query. If the response time was very high, SQL injection was then successful. To study the impact of SQL injection, one test was focused on how happened to the response time in the first hour. It is the most critical period to stop SQL injection hacking and to prevent any ways to make services and servers down. Our rationale is that if the proposed model could not withstand the most critical first hour, there would be little point to rescue data stored in the databases. Figure 14 shows MySQL (with McAfee) response time during SQL injection, whereby 1000 represents the infinity. The service could not withstand SQL injection after 16 minutes, resulting in the downtime and extremely slow response to the MySQL server.

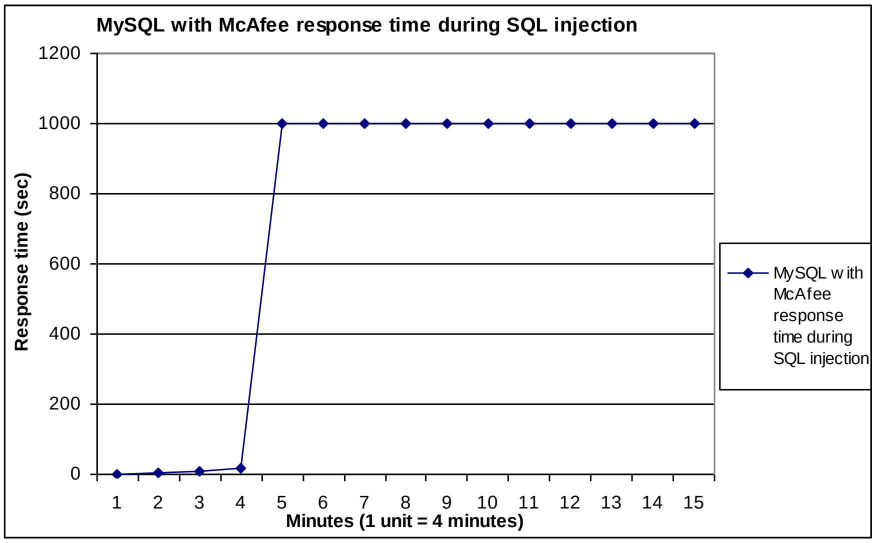


Figure 14: MySQL with McAfee response time during SQL injection

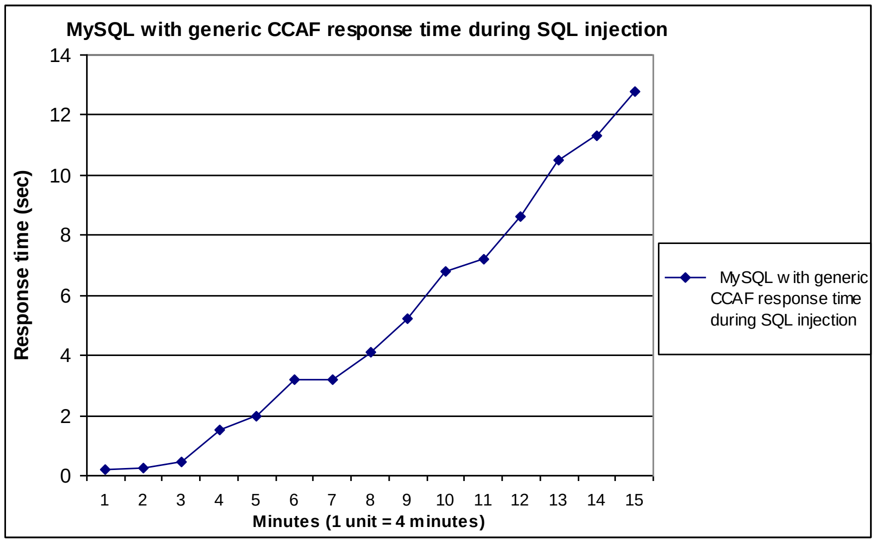


Figure 15: MySQL with generic CCAF response time during SQL injection

Figure 15 shows MySQL (with a generic CCAF) response time, which increased directly proportionally as the time went on to 1 hour. However, all response time was kept 12.8 second and below. Although CCAF multi-layered security can minimize the impacts of SQL injections, it does not provide a full protection to MySQL server. Even so, the response time was delayed to an acceptable extent of 12.8 second instead of 0.2 second per simple query. Figure 16 shows MySQL (with a full CCAF protection) response time, indicating that all the response time always stayed at 0.2 seconds regardless of the time. In other words, SQL injection could not hack its way into MySQL, since all the ports for MySQL had been enforced with security. For all unauthorized queries or queries not from the authorized users, SQL statements would not be accepted. Hence, the full protection could offer a better protection against any unauthorized access or malicious attack.

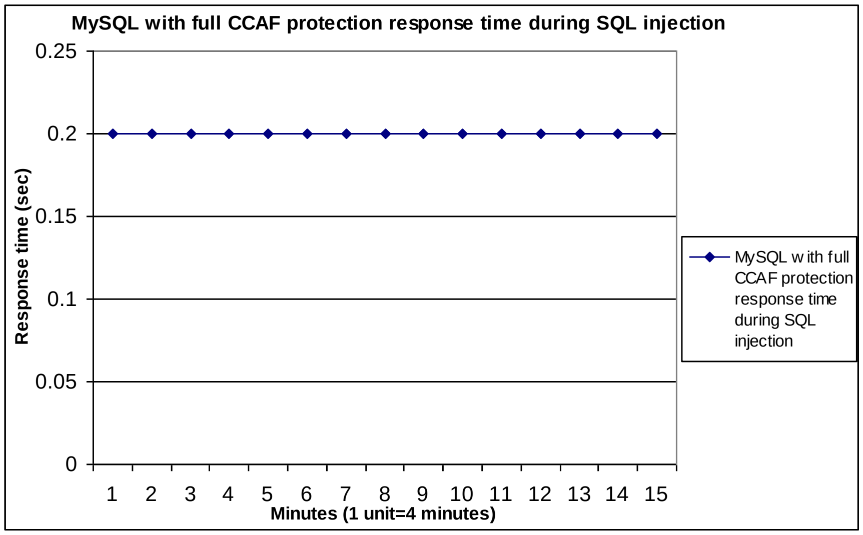


Figure 16: MySQL with a full CCAF protection response time during SQL injection

The next test was focused on SQL injection on a NoSQL database such as MongoDB, since it could work in the CCAF core technologies described in Section 3. The same set up and SQL injection approach for MySQL was adopted for the up-to-date version of MongoDB 3.0.4, with three scenarios to be tested. The first test was focused on a MongoDB with McAfee. The second test was focused on a MongoDB with a generic CCAF protection and the third test was focused on a MongoDB with a full CCAF protection. Before the start of all the tests, the same simple query for MySQL SQL injection was used for MongoDB, which could process SQL queries. However, it could take longer since it required more time to translate into a query that MongoDB could understand. This was similar to the development of the 64-bit Windows operating system in 2005, which used an internal emulator to translate 32-bit applications that can be portable and executable on 64-bit operating systems. Each simple query would take **0.40 seconds** with repeated experiments. All the experimental results for three tests showed that the response time was unchanged regardless of which security options, as indicated in Figure 17.

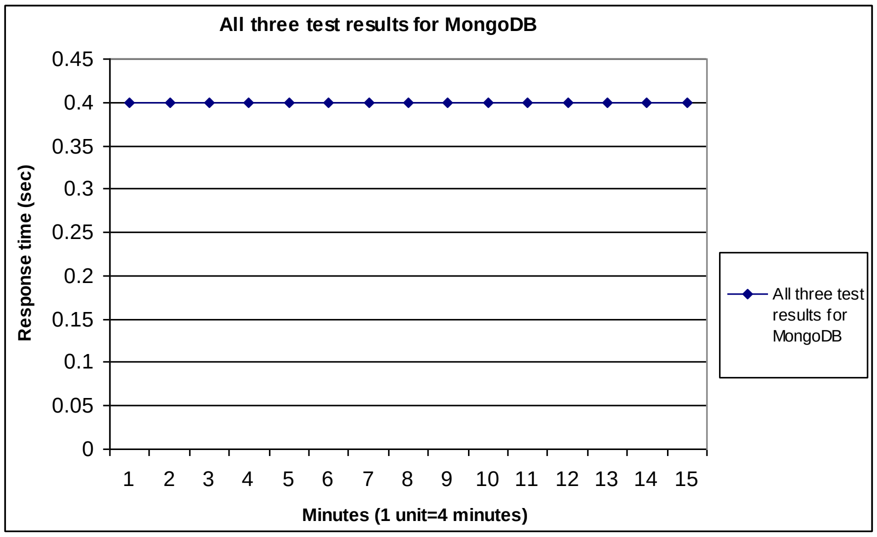


Figure 17: All three tests of conducting SQL injection for MongoDB

However, if hackers can gain partial access with MongoDB, then hacking can happen as illustrated by [30]. This means that hackers could use social engineering to trick the users believe the real website. Once the passwords were stolen, the hacker could then be broken into the MongoDB system console to change security settings. This does not mean that MongoDB is not vulnerable to security attacks.

4B.2 The subsequent 24 hours tests

This section reports the subsequent 24 hour SQL injection tests followed after the first hour test. The first pair of test was to conduct the SQL injections to MySQL without CCAF protection and MySQL with CCAF protection. Figure 18 shows MySQL SQL injection tests for both Macafee and CCAF generic protection, similar to the repeat of experiments for Figure 14 and Figure 15 for 24 hours. Figure 18 shows that the response time for CCAF generic protection grows to 1,000 seconds at the fourh hour, and remained all the ways to 1,000 seconds per simple response. The SQL injection with McAfee protection had 1,000 seconds per simple response throughout all hours.

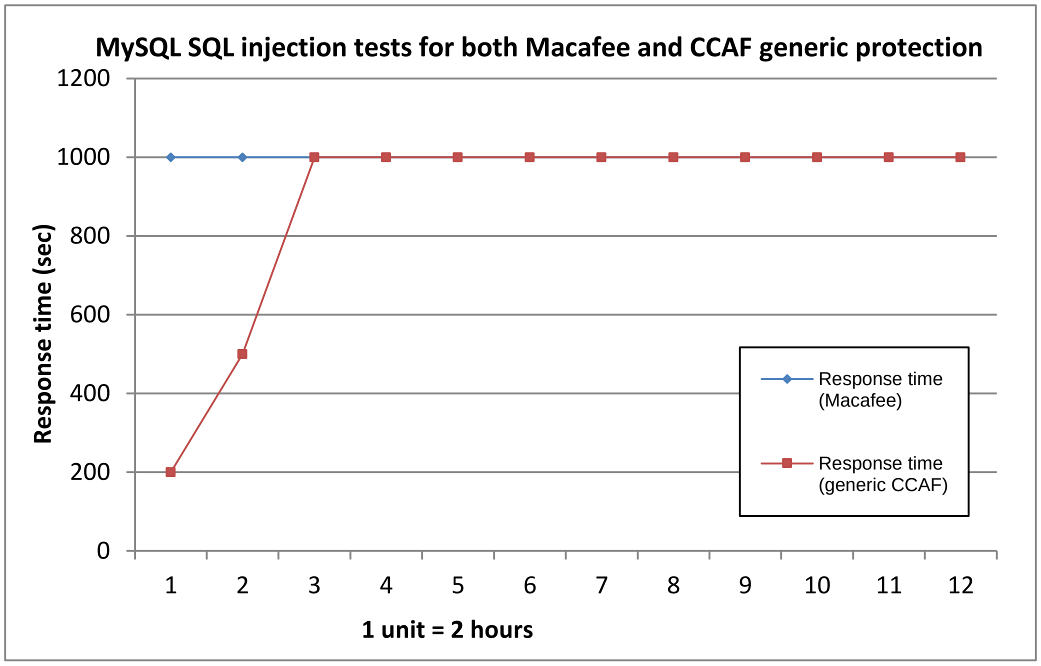


Figure 18: MySQL SQL injection tests with McAfee and generic CCAF protection test

Figure 19 shows MySQL and MongoDB SQL injection tests with MacAfee and three CCAF protection tests, similar to the repeat of experiments for Figure 16 and Figure 17 for 24 hours. MySQL with CCAF protection can withstand 24 hours. Similarly, MongoDB tests with McAfee, generic CCAF protection and full CCAF protection can withstand 24 hours. It shows that either the organization adopting business clouds should use MongoDB as database services. During the migration to NoSQL database, the existing MySQL servers should use full protection such as CCAF full security to ensure reliable services.

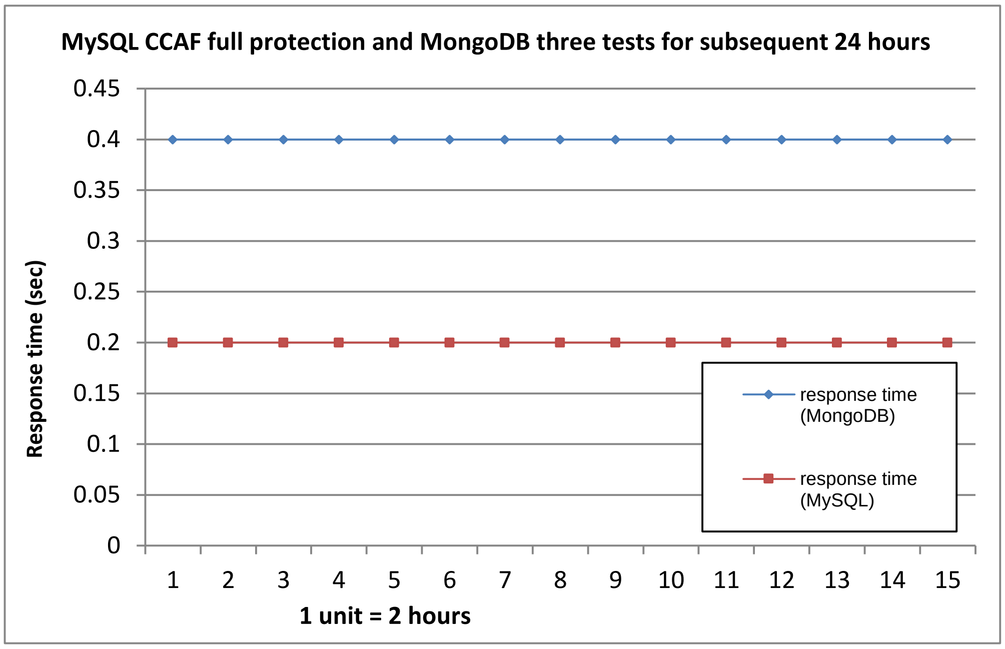


Figure 19: MySQL and MongoDB SQL injection tests with McAfee and three CCAF protection tests

4C Data simulations for CCAF multi-layered security

This section describes how data can be managed in the multi-layered security. While tests in Section 4A and 4B are focused on the identifying and blocking malicious files, often a large amount of clean and safe data is transferred through services and servers (in VMs and/or physical machines). Hence, experiments on the use case of transferring and handling a large quantity of clean data is required for the following reasons:

- The CCAF multi-layered security will not raise false alarm, since false alarm may cause damage to resources, time and reputation.
- The CCAF multi-layered security will identify the clean and safe data and have an excellent performance measurement to allow a large volume and quantity of data can be used in the services and servers.

In order to demonstrate that CCAF multi-layered security, system design and experiments on both aspects have been undertaken. The process involved has been described as follows. First, the system design to process data has been simulated to ensure that all the clean and safe can be checked and processed as fast and efficient as possible. Second, the experiments on using the real data have been tested to verify that no false alarm can take place and have short execution time to complete all checking of the data. The CCAF multi-layered security has always been up-to-date with the latest signature to identify the viruses, their sources of origins, behaviors, infection methods. However, it should take within seconds to identify whether each file is suspected to be malicious. In order to verify this, real data of between 10 TB and 50 TB have been used to validate the capabilities of the CCAF service. Prior using the real data for experiments, simulations on the data should be carried out first, so that the organizations that adopt the CCAF service can understand how long it will take to read all the data and check that they are not infected with viruses and trojans.

To facilitate simulation for predicting the data management, Business Process Modeling Notation (BPMN) has been used to simulate the data management under the CCAF multi-layered security. Figure 20 shows the BPMN model for analyzing the Cloud data security. There are three types of status for data: Data in Use; Data at Rest and Data in Motion. Data in Use means that data is used for services or users require them for their work. Data at Rest means that data is not in used for work and is archived in storage.

Data in Motion means either the data is about to change status from "Data at rest" to "Data in Use", or data has been transferred from one place to another successfully.

All types of the data need to satisfy all these three conditions at some point. The process starts with a data status decision (diamond symbol) passes that data based on that decision to any one of the paths of the cloud storage processes (data at rest, data in use, and data in change/transition). This in turn passed on to a data security pool which is a separate lane with dedicated security processes (such as data security area and data center update) to study security controls in place before it ends. Simulation in Figure 20 can be completed within seconds for analyzing terabytes of data. To support this further, experiments on simulating 10 TB, 20 TB, 30 TB, 40 TB and 50 TB of data with BPMN were conducted and their execution time was recorded.

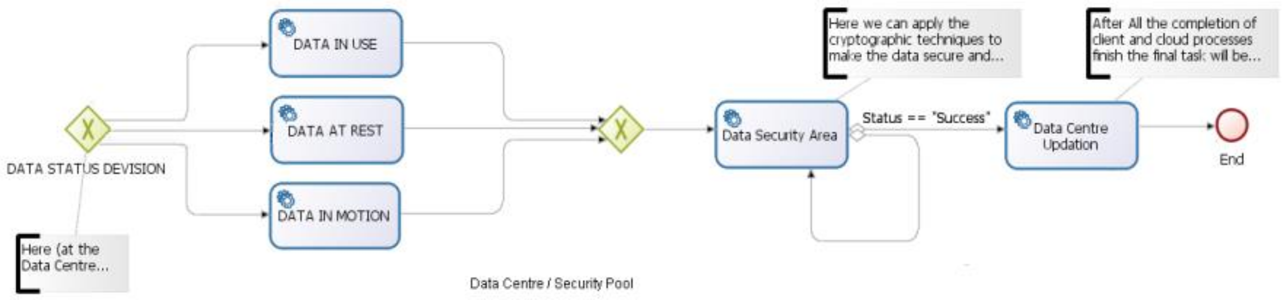

DATA IN USE
DATA AT REST
DATA IN MOTION
DATA STATUS DEVISION
Here (at the Data Centre...
Here we can apply the cryptographic techniques to make the data secure and...
After All the completion of client and cloud processes finish the final task will be...
Data Security Area
Status == "Success"
Data Centre Updation
End
Data Centre / Security Pool

Figure 20: BPMN model for three types of Cloud data security

Figure 21 shows the execution time to simulate processing of 10TB-50TB of clean data. All the time taken was between 1,250 seconds and 6,742 seconds and the relationship between all the datapoints presented to be in a linear regression, indicating that the larger the data, the more execution time in proportion to be simulated.

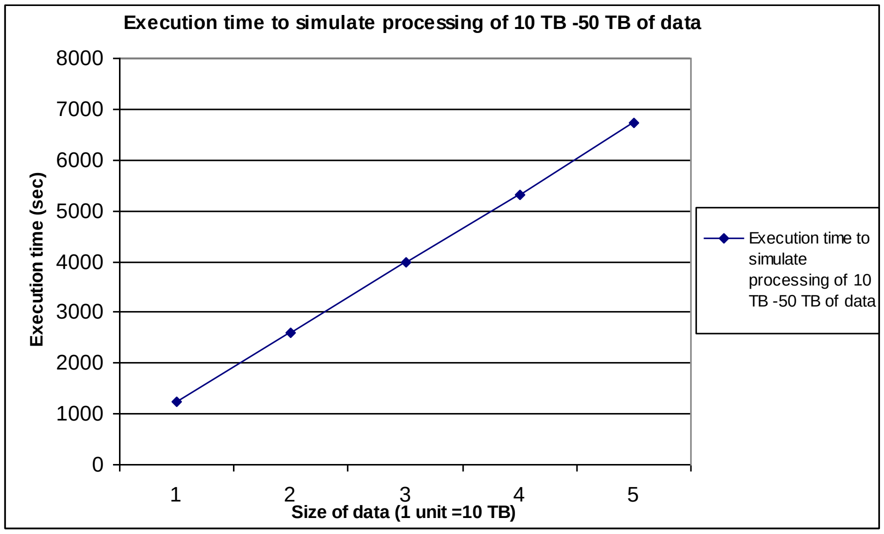


Figure 21: The execution time to simulate processing of 10TB-50TB of clean data

To demonstrate whether the CCAF multi-layered security could be subject to false alarm, 10 TB of real data was used for experiments. The 10 TB of data included 2 TB on documents and 8TB of multimedia data including graphics, images, and videos. For the purpose of testing, all documents were zipped up as 10 GB each, as well as the multimedia. The total number of documents used was 50,000. The total number of multimedia files was 10,000, with a total of 60,000. Each multimedia file was large in size and thus it could easily reach 8 TB in size. Before the tests, all files were checked by several anti-virus tools several times which reported there was no malicious file. Methods in Section 4A such as the use of precision, recall and F-measure of formulas 1 to 3 were used to measure the robustness and effectiveness of no false alarm results from the CCAF security. Experiments were performed three times to get the average values of the false alarm detected.

After the end of three experiments, results showed that 59,996 files were reported clean every time. However, 4 were reported suspected files but not viruses and trojans. The reason was that those multimedia files were altered by forensic tools and the altered files behaved differently than ordinary images. These files were considered as warning rather than false alarm. However if a strict rule was set, the warning was considered as a partial false alarm since the service could not fully differentiate between a working file and an infected file, leaving the user's own judgment. Hence, two types of results are presented in Table 3. All the values were 1 for test including warning and 1, 0.9999 and 0.9999 form test excluding warning. CCAF is excellent not to pick up false alarm as supported by the experimental results.

Table 3: Testing whether no false alarm was detected for CCAF multi-layered security

Services	Precision	Recall	F-Measure
CCAF (including warning)	**1**	**1**	**1**
CCAF (excluding warning)	**1**	**0.9999**	**0.9999**

5 Discussion

This section presents six topics for discussion. The first one is the security policy required to deliver the CCAF security framework to the fullest extent. The second topic is on aligning businesses with governance and information security and discussion about framework approach. The third topic is about integrating the secure EFSS system for the CCAF multi-layered security and the forth one is the general discussion about Big Data in the Cloud. Following that, our research contributions will be summed up to demonstrate that the proposed CCAF multi-layered security is a valid and working recommendation for business clouds, whereby businesses and institutes should always consider adopting multi-layered security approach to ensure that their data and services are guarded with up-to-date protection. Finally, research limitations of this research and general approaches from CCAF are discussed.

5A CCAF Security Adoptation

The key to successful transformation to cloud computing is to design flexibility for cloud business to be able to customize and add new security policies. Therefore, we believe the adoption of the Secure EFSS system – and the security issues that it attempts to address – sits at a critical organizational intersection, between the cost of appropriate counter-measures, versus the convenience and lower capital cost of operating a business cloud. Within the cloud environment, there has not yet been a standardized approach to security planning. Perhaps this reflects the shift away from distributed systems to a more logically centralized architecture, which is the hallmark of the cloud. Yet, there is very little discussion of the role that centralized versus system-specific or issue-specific policies play in the design of secure cloud environments. At the core of a cloud security plan – which should be aiming to reduce risk within budgetary limits, to the extent that management are prepared to accept residual risk after countermeasures have been implemented – lies the security policy. Going to market with security policies that are more or less restrictive could be the basis of competitive advantage. In the business of cloud computing market, there has not yet been any systematic approach to security planning and security policy changes that can be introduced in a systematic manner as it is important according to NIST [11, 31], a cloud security plan should encompass:

- Policies, or the set of decisions that management have made to defend the organization against perceived or measured threats, which can be expressed through standards, guidelines or procedures;
- Roles and responsibilities, or the matrix which maps who (or what) is responsible for which tasks that need to be undertaken to implement security policy;
- Planning, or the means by which security will be implemented during each stage of a system's lifecycle. In the case of cloud storage, this may relate primarily to data lifecycle, which imposes a set of even stricter standards, particularly relating to privacy;
- Assurance, or determining the extent to which the cloud environment is actually secured; and
- Accreditation, or the process of residual risk acceptance by management.

The multi-layered security approach for the CCAF security framework outlined in Section 4 can be mapped from the technical requirements to a set of organizational needs to minimize risk and customization of security policies. At present, this mapping is often ad hoc within cloud environments, relying on the tact knowledge of operational staff to link the two together. Future research will focus on how to formally specify these requirements, and providing the real-time means to provide assurance against organizational goals. For example, penetration testing should be performed at a level and frequency commensurate with the value of the data being protected.

5B Aligning businesses with governance and information security

A working framework is to align business with governance and information security, so that all the businesses processes and services can minimize risk, reduce the level of errors, improve efficiency and collaboration and enhance the business opportunities, reputation and deliveries of services [32]. For example, Cloud Computing Business Framework (CCBF) has demonstrated that aligning businesses with corporate, operational and research activities can provide businesses improvement in efficiency, revenue and customer satisfaction. There are projects delivered by CCBF to demonstrate that those organizations that adopt Cloud Computing can provide services critical to the businesses such as health informatics as a service, business intelligence as service and bioinformatics as a service. Real life examples for financial modeling and risk simulations to investigate what caused financial crisis in 2009 have been demonstrated [33]. Risk analysis and simulations are parts of enterprise security to help organizations to understand what risks are, how to measure, how to analyze and how to respond based on the results. Lessons learned from CCBF have been blended with our current CCAF security to fully integrate multi-layered security with governance and information security. All types of risk and security breaches can be informed and alerted in real-time to allow swift remedy actions. A board of governance body can ensure all the data and services are protected with security updates. Our proposal of the CCAF security is essential to business clouds since CCAF transforms security concepts into real practice. Examples demonstrated by CCAF can be used by organizations such as University of Southampton and Leeds Beckett University to ensure all data are safe and protected from real threats.

5C Integrating the Secure EFSS system for the CCAF multi-layered security

The concrete instance and core technologies of CCAF security presented in Section 3 were based on the Secure EFSS system, which could be deployed to an on-premise OpenStack cloud environment. One advantage is to offer a Dropbox-liked service that could backup and transfer the data. Differ from existing EFSS systems, the designs of the Secure EFSS system take enterprise security concerns into consideration and it has also addressed the opened EFSS security issues identified in this paper. To sum up, key contributions of the proposed integrated security approach are shown as follows:

(1) The firewall friendly secure deployment protects the services and enterprise data;

(2) The isolated personal storage space protects enterprise employees' privacies as well as increases system scalability;

(3) The Distinct Share Link supports secure sharing with internal enterprise colleagues and external business partners;

(4) Personal and shared files are encrypted by different keys which are further protected and managed by built-in key cascading management processes for an secure cloud file sharing environment;

(5) The sandbox-based cloud file synchronization extends the enterprise domain to employees' endpoint devices to prevent data leaks caused by file synchronization; and

(6) Finally, the out-of-band authentication method uses proxy-based authentication to protect the enterprise directory (e.g., AD or LDAP) and also prevent the employees' sensitive data to be logged by untrustworthy services.

5D Big Data in the Cloud

This section describes how the CCAF multi-layered security is relevant to the special issue: Big Data in the Cloud, a popular topic in the service computing and scalable computing. All the security solutions and proposals should allow organization that can adopt Cloud computing to offer data security and ensure all services are up-and-running with the maximum level of protection. OpenStack security has been integrated into our CCAF multi-layered security. The first layer was focused on the firewall and authentication. The second layer was focused on the identity management and OpenID played an important role enforcing identity management. The third layer was focused on encryption technologies as described in Section 3.

A large scale penetrating testing involved with 10,000 trojans and viruses were undertaken to ensure that the CCAF multi-layered security. Our CCAF multi-layered security can handle a sudden surge of malicious files and have an extremely high accuracy to block them. Similarly CCAF multi-layered security can handle with 50 TB of safe and clean data with all the execution time below 7,000 seconds. All the experiments had been successfully designed and conducted for Big Data in the Cloud.

5E Our research contributions

This section summarizes our research contributions as follows.

First, we develop a security framework that can deliver from theory to practice and make a conceptual framework into an architectural framework. We demonstrate the core technologies and explain how they can work together. We have performed experiments to ensure that our design and prototype are robust enough to withstand penetration testing and SQL injection.

Second, we implement a CCAF multi-layered security and blend it with core technologies from OpenStack and our existing work. The outcome of the experiments demonstrate that our service can detect and block viruses and trojans much better than existing tools since we have a F-measure of 0.99. Our CCAF multi-layered security has 100% not to report false alarm as supported by our experiments.

Third, our CCAF multi-layered security offer research contribution to volume, velocity and veracity of Big Data science. It is relevant to volume since CCAF security can handle 10,000 of malicious files. For clean and safe files, CCAF security can read 10 – 50 TB of files with ease. CCAF security is relevant to velocity since it can detect and block viruses and trojan in penetration testing and SQL injection ethical hacking. It can respond quickly to unpredicted events that lead to a surge of files and attacks. CCAF security offers veracity since all experimental results have a high percent of accuracy supported by our analysis and results in penetration testing and SQL injection. No attack can be achieved with our NoSQL database protected by CCAF security.

5F Limitations of this research and general approaches from CCAF

The major research limitation is the use of viruses and trojans for penetration testing. They are the 2013 known vulnerabilities. The most up-to-date versions are unavailable for testing samples as yet. Collaborators who can provide more up-to-date testing will be sought after for our future work. Other types of penetration testing will be adopted to ensure a better coverage of testing results. The lesson learned from the general approaches used by CCAF is that the multi-layered security for all Cloud and Big Data services are always better and safer than a single security solution adopted by many other services.

6 Conclusions and Future Work

An integrated security approach and its implementation – a CCAF multi-layered security was demonstrated in this paper. The motivation and the related literature about the CCAF security were explained and the core technologies were described, which considered enterprise security concerns and addressed the EFSS security issues. CCAF security was presented with the integration of the three layered security: firewall, identity management and encryption. To demonstrate CCAF multi-layered security as a working framework for business clouds, experiments were designed. Results show that CCAF multi-layered can detect and block 9,995 viruses and trojans during penetration test and could block above 85% of attacks for 100 hours shown in Figure 11. Each of the three layers could detect and block viruses and trojans between 0.015 and 0.105 per second. F-measure is 0.9975 which is well above other services. SQL injection was undertaken for MySQL and MongoDB. Results in the first hour and 24 hours tests showed that a full CCAF multi-layered security protection could block and prevent SQL injection for MySQL and MongoDB. All response time stayed as 0.20 and 0.40 seconds rather than 1,000 seconds per simple request. CCAF multi-layered security had achieved 100% rate for not detecting false alarm. It took less than 7,000 seconds to read all of 50 TB data in the experiments. CCAF security policy could work with real life examples and could also align with businesses to protect assets and data. Our work also demonstrated our three major research contributions and explained how our work could offer added value for volume, velocity and veracity of Big Data service in the Cloud. Our future work is to strengthen our international collaboration with partners in different countries and develop different proofs-of-concepts, prototypes, services and research consultancy with our partners.

Acknowledgment

Part of this study is conducted under the “Big Data Technologies and Applications Project (1/4)” of the Institute for Information Industry which is subsidized by the Ministry of Economic Affairs of the Republic of China. Part of this study has been redesigned and experimented with testing and experiments offered by Dr. Victor Chang’s private resources conducted in England.

References

[1] X., Zhang, M., Nakae, M. J., Covington & R., Sandhu. "Toward a usage-based security framework for collaborative computing systems". ACM Transactions on Information and System Security (TISSEC), 11(1), 3, 2008.

[2] H., Takabi, J. B., Joshi, & G. J., Ahn. "Securecloud: Towards a comprehensive security framework for cloud computing environments". In 2010 IEEE 34th Annual Conference, Computer Software and Applications Workshops (COMPSACW) (pp. 393-398), 2010, July.

[3] T., Zia, & A., Zomaya. "A security framework for wireless sensor networks". In Proceedings of the IEEE Sensors Applications Symposium, pp. 49-53, 2006, February.

[4] T., Mather, S., Kumaraswamy, & S., Latif, "Cloud security and privacy: an enterprise perspective on risks and compliance". O'Reilly Media, Inc., 2009.

[5] S., Marston, Z., Li, S., Bandyopadhyay, J., Zhang, & A., Ghalsasi, "Cloud computing—The business perspective. Decision Support Systems", 51(1), 176-189, 2011.

[6] V., Chang, R. J., Walters, & G., Wills. "Cloud Storage and Bioinformatics in a private cloud deployment: Lessons for Data Intensive research". In Cloud Computing and Services Science (pp. 245-264). Springer International Publishing, 2013.

[7] M. Ramachandran, & V., Chang. "Cloud Security proposed and demonstrated by Cloud Computing Adoption Framework". In 2014 Emerging Software as a Service and Analytics workshop, 2014.

[8] M., Ramachandran, V., Chang, & C.S., Li. "The Improved Cloud Computing Adoption Framework to deliver secure services", In, Emerging Software as a Service and Analytics 2015 Workshop (ESaaSA 2015), in conjunction with CLOSER 2015, Lisbon, Portugal, 20 - 22 May 2015.

[9] R. K., Ko, P., Jagadpramana, M., Mowbray, S., Pearson, M., Kirchberg, Q., Liang, & B. S., Lee. TrustCloud: A framework for accountability and trust in cloud computing. In 2011 IEEE World Congress on Services (SERVICES), pp. 584-588, July 2011.

[10] S., Pal, S., Khatua, N., Chaki, & S., Sanyal. "A new trusted and collaborative agent based approach for ensuring cloud security". arXiv preprint arXiv:1108.4100. 2011.

[11] NIST, "Framework for Improving Critical Infrastructure Cybersecurity", technical report, Version 1.0, February 2014.

[12] S. Subashini and V. Kavitha, “A Survey on Security Issues in Service Delivery Models of Cloud Computing,” Journal of Network and Computer Applications, vol. 34, no. 1, pp. 1-11, January 2011.

[13] M., Basso and J. Mann, “MarketScope for Enterprise File Synchronization and Sharing,” Gartner, 12 Feburary 2013.

[14] Y.H., Kuo, Y.L., Jeng, and J.N., Chen, “A Hybrid Cloud Storage Architecture for Service Operational High Availability,” Proceedings of the 37th IEEE Annual International Computers, Software & Applications Conference (IEEE COMPSAC 2013), pp. 487-492, July 2013.

[15] J., Sanders, “Dropbox and Box Leak Files in Security Through Obscurity Nightmare,” TechRepublic, http://www.techrepublic.com/article/dropbox-and-box-leak-files-in-security-through-obscurity-nightmare/, Retrieved on August 15th, 2014.

[16] S.A., Baset, “Open Source Cloud Technologies,” Proceedings of the Third ACM Symposium on Cloud Computing (SoCC), October 2012.

[17] R., Cattell, “Scalable SQL and NoSQL Data Stores,” ACM SIGMOD Record, vol. 39, no. 4, pp. 12-27, December 2010.

[18] J., Han, E., Haihong, G., Le, and J., Du, “Survey on NoSQL database,” Proceedings of the 6th International Conference on Pervasive Computing and Applications (ICPCA 2011), pp. 363-366, October 2011.

[19] Y.H., Kuo and Y.L., Jeng, “Apparatus, Method, and Non-transitory Computer Readable Storage Medium Thereof for Controlling Access of a Resource,” U.S. Patent Application (Priority No.: 13/954,885).

[20] Y.H., Kuo and Y.L., Jeng, “Cloud Storage Server and Management Method Thereof,” U.S. Patent Application (Priority No.: 14/032,440).

[21] D., Dolev and A.C., Yao, “On the Security of Public Key Protocols,” IEEE Trans. on Information Theory, vol. 29, no. 2, pp. 198-208, March 1983.

[22] U.M., Maurer and J.L., Massey, “Cascade Ciphers: The Importance of Being First,” Journal of Cryptology, vol. 6, no. 1, pp. 55-61, March 1993.

[23] C., Percival, “Stronger Key Derivation via Sequential Memory-hard Functions,” in The BSD Conference (BSDCan), May 2009.

[24] Y.H. Kuo and Y.L. Jeng, “Secure Synchronization Apparatus, Method, and Non-Transitory Computer Readable Storage Medium Thereof,” U.S. Patent Application (Priority No.: 14/292,901).

[25] K.Y. Wang, Y.H. Kuo, and Y.L. Jeng, “Synchronization Apparatus, Method, and Non-Transitory Computer Readable Storage Medium,” U.S. Patent Application (Priority No.: 14/300,955).

[26] V., Chang, & M. Ramachandran, "Towards Data Security with the Cloud Computing Adoption Framework", IEEE Transactions on Services Computing, forthcoming, 2016.

[27] N., Antunes, N., & M., Vieira, M. "Assessing and Comparing Vulnerability Detection Tools for Web Services: Benchmarking Approach and Examples". IEEE Transactions on Services Computing,, 8(2), 269-283, 2015.

[28] A., Kieyzun, P. J., Guo, K., Jayaraman & M. D., Ernst, M. D. "Automatic creation of SQL injection and cross-site scripting attacks". In IEEE 31st International Conference on Software Engineering, 2009. ICSE 2009. pp. 199-209, 2009, May.

[29] S., McClure, S., Shah, & S., Shah, "Web hacking: Attacks and defense". Addison-Wesley Longman Publishing Co., Inc., 2002.

[30] OWASP article, Testing for NoSQL injection, technical report, accessible on ttps://www.owasp.org/index.php/Testing_for_NoSQL_injection, July, 2015.

[31] NIST. "An Introduction to Computer Security". Publication 800-12, 1996, downloaded from http://csrc.nist.gov/publications/nistpubs/800-12/handbook.pdf

[32] M. D., Harris, D., Herron, & S., Iwanicki. The business value of IT: managing risks, optimizing performance and measuring results. CRC Press, 2008.

[33] V., Chang. "A proposed Cloud Computing Business Framework". ISBNs: 9781634820172 (print), Nova Science Publisher, 2015.