**Data Protection in OpenStack**

I. **INTRODUCTION**   
  
Cloud computing is increasingly prevalent as organizations look to slash IT costs and to simplify management of IT resources. Many organizations also expect cloud computing to increase productivity and give them a competitive market advantage. Despite these benefits, security and privacy concerns remain major inhibitors to adoption for more than 60% of companies [1]. One recent study claims that as few as 1 in 10 cloud providers use even basic security features such as encrypting data at rest [2]

**II.BACKGROUND   
  
 A. OpenStack Services**

The following list identifies the major OpenStack services (each service’s codename appears in parentheses) and provides a brief description of the service.

**Block Storage (Cinder)** Cinder supports user-defined provisioning of block storage, typically based on commercial network-attached storage (NAS) or storage area network (SAN) products.

**Compute (Nova)** Nova provides on-demand, self-service provisioning and management of VMs. Many hypervisors are supported, including KVM, Xen, Hyper-V, and vSphere.

**Dashboard (Horizon)** Horizon provides a web interface to various features within OpenStack such as uploading VM images and booting a VM.

**B. Contribution Process**

OpenStack follows a structured process for contributions, The semi-annual OpenStack Summit initiates the development cycle and facilitates design discussions among contributors. Blueprints provide a high level description of new features and track their development. Specifications add design information and technical details, allowing OpenStack’s core reviewers (i.e., project maintainers) to provide direction prior to implementation. Merging code changes requires acceptance from two core reviewers in addition to passing automated unit and integration tests.

**III. SECURITY FEATURES**

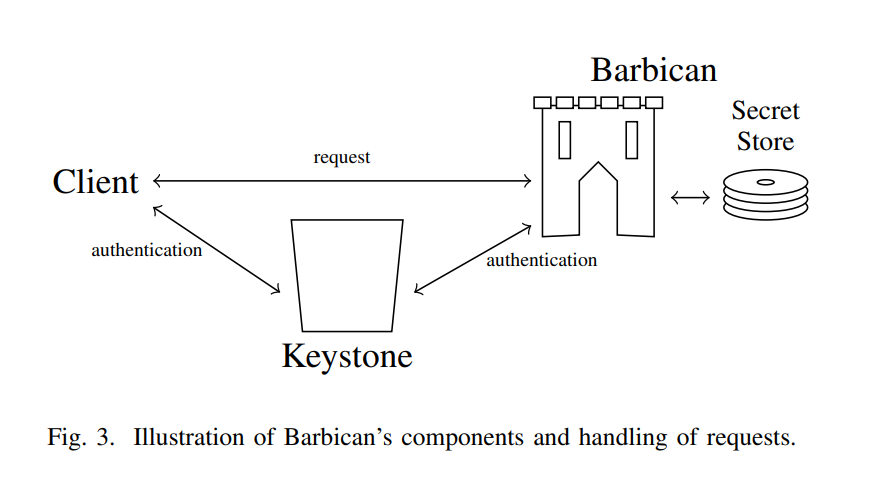
**Key Management**

Effective data protection requires strong cryptographic keys that are stored securely. A secure key manager is a critical part of cloud security. Key managers have a number of benefits, including

• the generation of cryptographically strong keys;

• performing cryptographic operations (e.g., signing) of data without the key leaving the device;

• centralization of sensitive data, which includes passphrases, keys, and certificates;



OpenStack’s key management service, **Barbican**, includes a REST interface for managing secrets, including symmetric keys, asymmetric key pairs, X.509 certificates, passphrases, and opaque data objects. OpenStack services integrate with Barbican via a simple Python interface. Our core contribution to Barbican was support for KMIP [3], a communication protocol defining how to encode key management messages and object data for exchange. The addition of KMIP support in Barbican required the creation of PyKMIP,2 an open-source Python library, which implements the KMIP specification. The library also includes a KMIP server application that can be used for testing and demonstration [6]. This server allows developers to test the full key management life cycle, defraying the upfront investment required for production-level key management appliances.

**Secrets API**

The secrets resource is the heart of the barbican service. It provides access to the secret / keying material stored in the system.

Barbican supports the storage of data for various content-types securely.

**How to Create a Secret**

**Single Step Secret Creation**

The first secret we will create is a single step secret. Using a single step, barbican expects the user to provide the payload to be stored within the secret itself. Once the secret has been created with a payload it cannot be updated. In this example we will provide a plain text secret. For more information on creating secrets you can view the POST /v1/secrets section.

**curl -X POST -H "content-type:application/json" -H "X-Auth-Token: $TOKEN" \**

-d '{"payload": "my-secret-here", "payload\_content\_type": "text/plain"}' \

http://localhost:9311/v1/secrets

**Two Step Secret Creation**

The second secret we will create is a two-step secret. A two-step secret will allow the user to create a secret reference initially, but upload the secret data after the fact. In this example we will not provide a payload.

**curl -X POST -H "content-type:application/json" -H "X-Auth-Token: $TOKEN" \**

-d '{}' <http://localhost:9311/v1/secrets>

**How to Update a Secret**

To update the secret data,the secret ref is used from the previous example. You will have to substitute the uuid after /secrets/ with your own in order to update the secret.

**curl -X PUT -H "content-type:text/plain" -H "X-Auth-Token: $TOKEN" \**

-d 'my-secret-here' \

<http://localhost:9311/v1/secrets/2a549393-0710-444b-8aa5-84cf0f85ea79>

**How to Retrieve a Secret**

To retrieve the secret we have created we will need to know the secret reference provided via the initial creation

**curl -H "Accept: text/plain" -H "X-Auth-Token: $TOKEN" \**

<http://localhost:9311/v1/secrets/2a549393-0710-444b-8aa5-84cf0f85ea79/payload>

**How to Delete a Secret**

To delete a secret we will need to know the secret reference provided via the initial creation

**curl -X DELETE -H "X-Auth-Token: $TOKEN" \**

<http://localhost:9311/v1/secrets/2a549393-0710-444b-8aa5-84cf0f85ea79>

**CONCLUSION**

This paper presents several data protection features available within OpenStack. These features include the PyKMIP library, block storage encryption

As part of our future work, we will investigate additional security features for OpenStack. However, much of our ongoing focus is to support and to improve the existing features that we described in this paper. In particular, we are gradually expanding PyKMIP’s support for KMIP’s Basic Baseline profile [3] and creating a dashboard plugin to enumerate a user’s keys and to retrieve their certificates. We also continue to add enhancements to the image integrity feature (e.g., the ability to sign and verify snapshots of VM instances).