Securing User Authentication using Single Sign-On in Cloud Computing

Ashish Revar
Assistant Professor, CE/IT Department,
Marwadi Education Foundation, Rajkot, India
ashishrevar@gmail.com

Madhuri Bhavsar
Associate Professor, CSE Department,
Nirma University, Ahmedabad, India
madhuri.bhavsar@nirmauni.ac.in

Abstract—In past three decades, the world of computation has changed from centralized (client- server not web-based) to distributed systems and now we are getting back to the virtual centralization (Cloud Computing). This paper aims to design and implement an optimized infrastructure for secure authentication and authorization in Cloud Environment. SSO (Single Sign-On) is a process of authenticate once and gain access of multiple resources. Aim of SSO is to reduce number of login and password in heterogeneous environment and to gain balance in Security, Efficiency and Usability. This paper leads to implementation of Cloud for Storage and Virtual Machines Images to run the SSO on the top layer of Cloud. This has entailed a review and comparison of existing single sign-on architectures and solutions, the development of a new architecture for single sign-on, an analysis of single sign-on threats within a Cloud context, a derivation of single sign-on objectives in Cloud, leading up to the security requirements for single sign-on in Cloud. Security and functionality are the main driving factors in the design. Others factors include performance, reliability, and the feasibility of integration.

Index Terms—Cloud computing, Cloud security, Single Sign-On, User authentication.

I. INTRODUCTION

A. Understanding Cloud Computing

Cloud computing [1] is a complex infrastructure of software, hardware, processing, and storage that is available as an on-demand service with pay-as-you-go manner through internet. Cloud computing is a computing paradigm in which tasks are assigned to a combination of connections, software and services accessed over a network. This network of servers and connections is collectively known as ”the cloud”. Cloud computing platform dynamically provisions, configures, reconfigures servers as needed.

B. Single Sign-On

Presently, Users are having multiple accounts in various Service Providers with different usernames accompanied by different password. Therefore the vast majority of network users tend to use the same password wherever possible, posing inherent security risks. The inconvenience of multiple authentications not only causes users to lose productivity, but also imposes more administrative overhead. Enterprises today are seriously considering the use of Single Sign On technology[2] to address the password explosion because they promise to cut down multiple network and application passwords to one.

II. UBUNTU ENTERPRISE CLOUD

Ubuntu Enterprise Cloud (UEC)[5] brings Amazon EC2-like infrastructure capabilities inside the firewall. The UEC is powered by Eucalyptus, an open source implementation for the emerging standard of the EC2 API. This solution is designed to simplify the process of building and managing an internal cloud of any size, to create own self-service infrastructure.

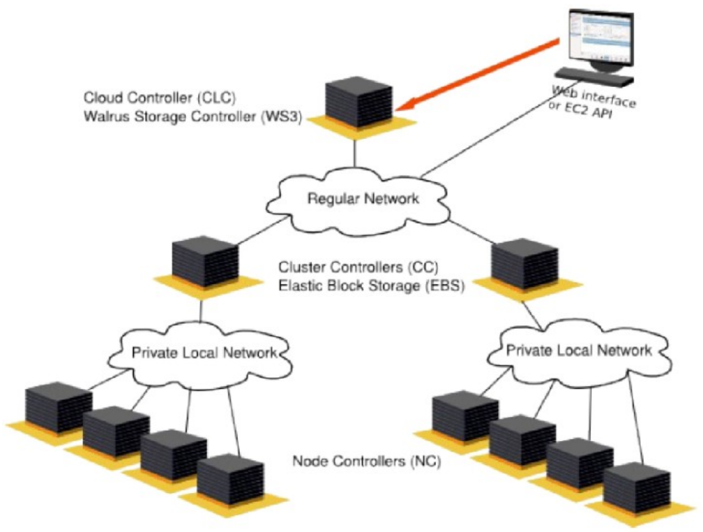

Fig. 1. UEC Architecture[5]

(SSO)

A. UEC elements

The architecture of Eucalyptus[6], which is the main component of Ubuntu Enterprise Cloud, has been designed as modular set of 5 simple elements that can be easily scaled:

Cloud Controller (CLC)
Walrus Storage Controller (WS3)

Elastic Block Storage Controller (EBS)
Cluster Controller (CC)
Node Controller (NC)

Each element is acting as an independent web service that exposes Web Service Description Language (WSDL) document defining the API to interact with it. It is a typical web service architecture.

III. System Desing and Implementation

A. Kernel Based Virtual Machine

KVM (for Kernel-based Virtual Machine) is a full virtualization solution for Linux on x86 hardware containing virtualization extensions (Intel VT or AMD-V)[8]. It consists of a loadable kernel module, kvm.ko, that provides the core virtualization infrastructure and a processor specific module, kvm-intel.ko or kvm-amd.ko. KVM also requires a modified QEMU although work is underway to get the required changes upstream. UEC has support of KVM, so I can run multiple virtual machines running unmodified Linux images[7], [8].

B. Cloud Topology

The configuration shown in Fig.2 puts all of the user facing components (CLC/Walrus) and back-end control components (CC/SC) on a single system, and uses the second for VM hosting.

1) **machine A**: CLC/Walrus/CC/SC
2) **machine B**: NC

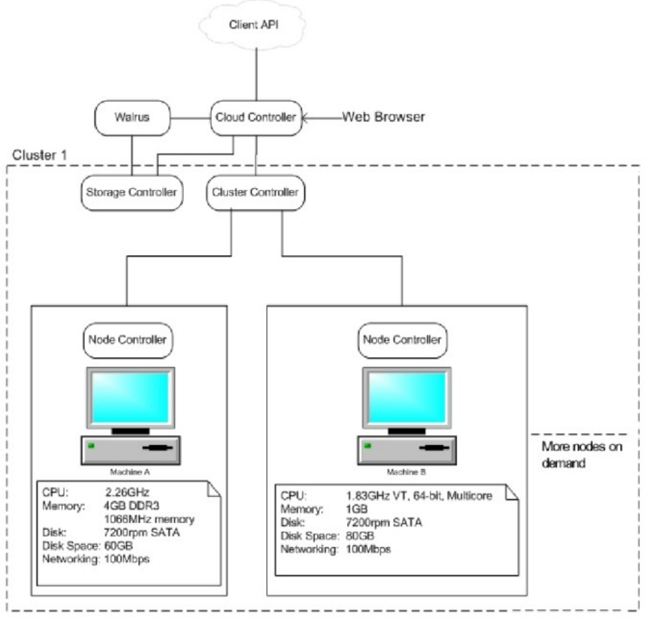

Fig. 2. Cloud Topology

C. Cloud Implementation

NC expects a bridge, so I have created a bridge and added eth0 as an interface. Bridge was automatically config-ured with the IP allocated to it. This installation is having DHCP server in network, then STATIC/SYSTEM mode will nerver work. So, I have to remodify the VNET _MODE in /etc/eucalyptus/eucalyptus.conf.[9]

```
auto eth0
 iface eth0 inet manual auto br0
 iface br0 inet static address
 192.168.1.103 gateway
 192.168.1.1 bridge_ports eth0
 bridge_fd 9 bridge_hello 2
 bridge_maxage 12 bridge_stp
 off
```

After installing CC and NC, I needed to retrieve their credentials for cloud. This can be done either from a web browser or from command line. I used command line and executed the following commands:

```
$ mkdir -p ~/.euca
$ chmod 700 ~/.euca
$ cd ~/.euca
$ sudo euca_conf --get-credentials mycreds.zip
$ unzip mycreds.zip
$ cd -
```

After obtaining the credentials, I setup EC2 API and AMI tools on the server using X.509 certificates. For this, sourced the included ”eucarc” file to set up the Eucalyptus environ-ment. This added to the /.bashrc file, so that the Eucalyptus environment is set up automatically when I log in.

```
$ . ~/.euca/eucarc
$ echo "[ -r ~/.euca/eucarc ] &&
. ~/.euca/eucarc" >> ~/.bashrc
```

To register the node to the cluster and then to find the availability zone of my cloud I have performed the following step.

```
$ sudo euca_conf --no-rsync --discover-nodes
$ euca-describe-availability-zones    verbose    AVAILABILITYZONE    uec-cloud
192.168.1.103
AVAILABILITYZONE  |- vm types      free / max      cpu     ram   disk
AVAILABILITYZONE  |- m1.small      0002   / 0002   1       128      2
AVAILABILITYZONE  |- c1.medium     0002   / 0002   1       256      5
AVAILABILITYZONE  |- m1.large      0001   / 0001   2       512     10
AVAILABILITYZONE  |- m1.xlarge     0001   / 0001   2      1024     20
AVAILABILITYZONE  |- c1.xlarge     0000   / 0000   4      2048     20
```

Before running an instance of the image, I have first created a keypair that can be used to log into the instance. The default security group configured to allow SSH access (port 22).

```
if [ ! -e ~/.euca/mykey.priv ]; then touch
      ~/.euca/mykey.priv
      chmod 0600 ~/.euca/mykey.priv euca-add-keypair mykey >
      ~/.euca/mykey.priv
fi
$ euca-describe-groups
$ euca-authorize default -P tcp -p 22 -s 0.0.0.0/0
$ euca-authorize default -P tcp -p 3389 -s 0.0.0.0/0
```

So all set to launch the instance. From euca-describe-images, get the EMI ID. Now launch the instance.

```
$  euca-describe-images
IMAGE     eri-0D4D117E           image-store-1268999725/ramdisk.manifest.xml
IMAGE     eki-F8BF1114           image-store-1268999725/kernel.manifest.xml
IMAGE     emi-E2861098           image-store-1268999725/image.manifest.xml
$ euca-run-instances emi-E2861098 -k mykey -t c1.xlarge --addressing private
```

Every instance has 2 addresses associated with it.

Private address: Address inside the VM

Public address: Address on the CLC and port forwarded (DNAT) to the instance

When the instance is fully started, the above state became ’running’. IP address assigned to this instance in the output, then connect to it:

```
IPADDR=$(euca-describe-instances  |  grep  emi-E2861098
|   grep  running
|   tail -n1 | awk ’{print $4}’) ssh -i ˜/.euca/mykey.priv ubuntu@192.168.1.103
```

D. Setting up Single Sign-On

The flow of Cloud based SSO is elaborated in 3, in which user authenticates to the central authentication server, and this authentication server itself supplies the user credential (e.g., username and password) to the appropriate server whenever the user requests to use an application on another server. This module is developed using PHP language, [11], [10]

which enables user to get them register with this server and store their credentials. This authentication proxy server uses a database to maintain all the credentials for the registered users.

Procedure of authenticating users is through cURL[12] and SSL which makes itself robust by creating a secure channel over an insecure network. This ensures reasonable protection from eavesdroppers and man-in-the-middle attacks, provided that adequate cipher suites are used and that the server certificate is verified and trusted. When users are creating SSO agents with server, information is passes through http requests and uses RSA algorithm to encrypt and store.[11], [10]

E. Authentication using cURL

```
$username="admin"; $password="blog"; $url="http://wordpressblogURL/";
$cookie="cookie.txt";

$postdata = "log=". $username ."&pwd=". $password ." &wp-
submit=Log%20In&redirect_to=". $url . "blog/wordpress/wp-
admin/&testcookie=1";
$ch = curl_init(); curl_setopt ($ch, CURLOPT_URL, $url . "blog/wordpress/wp-
login.php");

curl_setopt ($ch, CURLOPT_SSL_VERIFYPEER, FALSE); curl_setopt
($ch,
CURLOPT_USERAGENT, "Mozilla/5.0 (Windows; U; Windows NT 5.1;
en-US;
rv:1.8.1.6) Gecko/20070725 Firefox/2.0.0.6"); curl_setopt ($ch,
CURLOPT_TIMEOUT, 60); curl_setopt ($ch,
CURLOPT_FOLLOWLOCATION, 1);
curl_setopt ($ch, CURLOPT_RETURNTRANSFER, 0);
curl_setopt ($ch,
CURLOPT_COOKIEJAR, $cookie); curl_setopt ($ch,
CURLOPT_REFERER, $url
.  "blog/wordpress/wp-login.php");
```

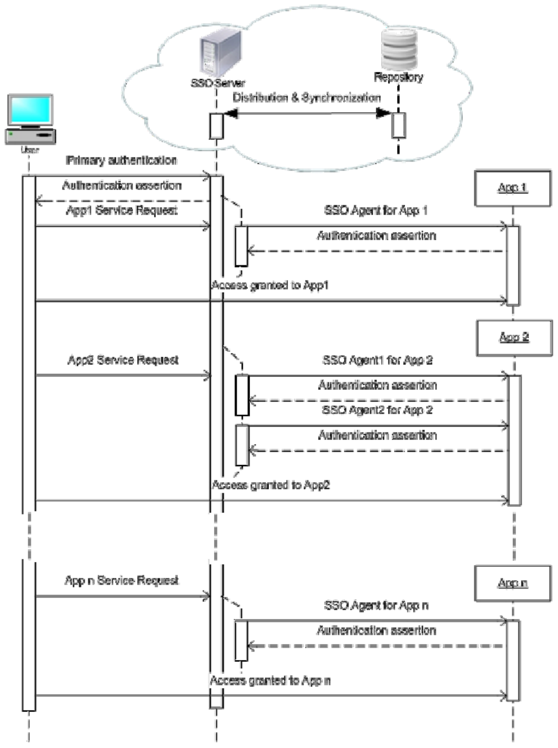

Fig. 3. Single Sign-On Flow

```
curl_setopt ($ch, CURLOPT_POSTFIELDS, $postdata); curl_setopt
($ch,
CURLOPT_POST, 1); $result = curl_exec ($ch);
curl_close($ch);
echo$result;

exit;
```

IV. Result

The objective of this analysis is to measure the value, in terms of effectiveness and efficiency, of a specific security initiative to implement a Single Sign-On system. Effectiveness is measured in terms of password compliance, access related incidents, and the time required to provision and de-provision accounts. Efficiency is measured in terms of support workload and effort and a simple ROI calculation.

The simple ROI formula[13] used is:

$$\frac{\text{(reduction in effort + (reduction in incidents} \quad \text{cost of incidents))}}{\text{(system cost)}} \qquad (1)$$

Current ROI: 53% Projected ROI:97%

Assumptions:

Dollar cost per Account Compromise: $10,000

Loaded Hourly effort cost: $53/hour

Monthly averages used for projection

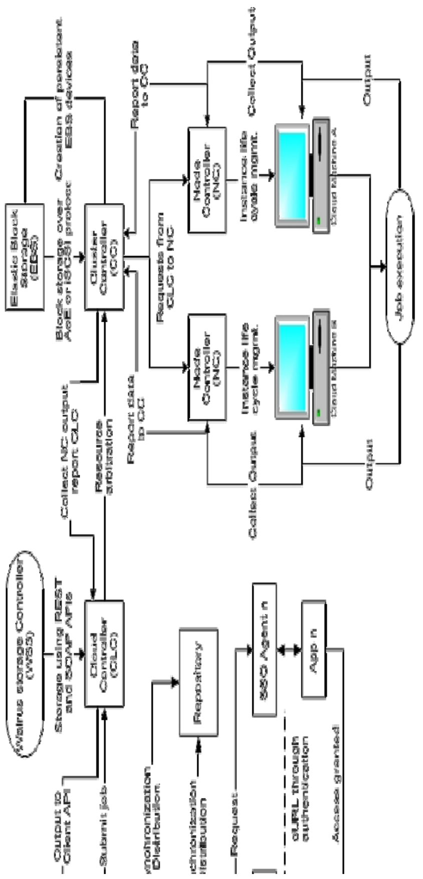
Elastic Block Storage (EBS)
Creation of persistent EBS devices
Block storage over AoE or iSCSI protocol
Cluster Controller (CC)
Report data to CC
Collect Output
Node Controller (NC)
Instance life cycle mgmt.
Cloud Machine 1
Cloud Machine 2
Output
Job execution
Requests from CLC to NC
Collect NC output report CLC
Resource arbitration
Walrus storage Controller (WS3)
Storage using REST and SOAP APIs
Cloud Controller (CLC)
Output to Client API
Submit job
Synchronization Distribution
Repository
Request
SSO Agent n
App n
URL through authentication
Access granted

Fig. 4. Data flow diagram of SSO deployed on SSO

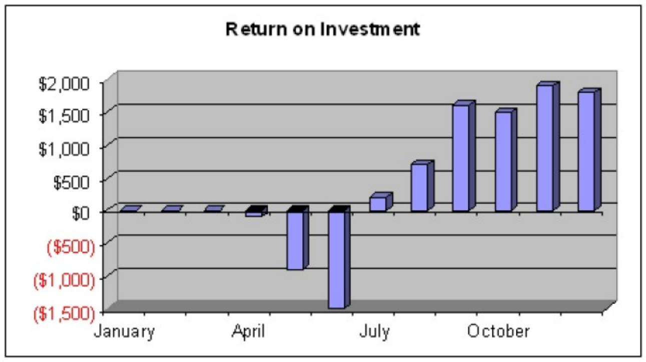

Fig. 5. Return on Investment[13]

V. Conclusion

A high level design of the Cloud based SSO solution has been developed, which involved designing a new architecture. The new architecture consists only the
authentication mechanism on the SSO Server, resulting in moving computation and storage to the Cloud. Because the Authentication Server is the central component in providing
single sign-on functionality, the AUS (SSO Server) was the focus of study for the development of the security
requirements for the SSO system.

Single sign-on greatly enhances the usability of the Cloud environment by allowing users to authenticate once to access applications on multiple machines. Such capabilities may lead to greater acceptance of these solutions in large community. Cloud server can activate and retire resources as needed, dynamically update infrastructure elements, and move workloads to improve efficiency without having to worry about creating new infrastructures. If the number of SSO users increases, this feature will be fruitful.

REFERENCES

[1] Rich Maggiani, Markoff, John, ”Cloud Computing Is Changing How We Communicate”, Communication Consultant Solari Communication.

[2] Rion Dutta, ”Planning for Single SignOn”, White Paper, MIEL e-Security Pvt.

[3] Jan De Clercq, ”Single Sign-on Architectures”, Proceedings of Infras-tructure Security: International Conference, InfraSec 2002, Bristol, UK, pg 40-58, October 1-3, 2002.

[4] Rajesh Kumar Singh and Alwyn R Pais, ”Secure Web Based Single Sign-On (SSO) framework using Identity Based Encryption System”, International Conference on Advances in Recent Technologies in Com-munication and Computing, IEEE Computer Society, 2009.

[5] Simon Wardley, Etienne Goyer & Nick Barcet, ”Ubuntu Enterprise Cloud Architecture”, Technical White Paper, August 2009, Canonical 2009.

[6] Johnson D, Kiran Murari, Murthy Raju, Suseendran RB, Yogesh Giriku-mar, ”Eucalyptus Beginner’s Guide - UEC Edition”, v1.0, 25 May 2010, CSS Corp. Pvt. Ltd.

[7] Eucalyptus Documentation, ”Managing Eucalyptus Images”, available at http://open.eucalyptus.com/wiki/EucalyptusImageManagement_v1.5.

[8] KVM, ”Kernel Based Virtual Machine”, available at http://www. linux-kvm.org/page/Main Page._

[9] Kiran Murari, ”UEC: CC and NC on a single machine”, March 19, 2010, available at http://kiranmurari.wordpress.com/2010/03/19/ uec-cc-nc-single-machine/.

[10] http://techportal.ibuildings.com/2009/03/31/php-and-the-cloud/

[11] http://ycscripts.co.cc/free-php-scripts/1/User-Management-System

[12] http://curl.haxx.se/

[13] Andrew Sudbury, Director, Security Metrics Design & Best Practices,
”Highlights of a Security Scorecard Project”, ClearPoint Metrics.