ISSN(Online) :2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 9, September 2015

Multi-Tiered Lightweight Virtualization for Web Services

Mercy H[1], Dr. K.P. Kaliyamurthie[*2]

[1]Assistant Professor , Department of Computer Science And Engineering, Jerusalem College of Engineering, Chennai, India

[2]Assistant Professor, Department of Computer Science Engineering, Bharath University, Chennai, India

[*]Corresponding Author

ABSTRACT: With increasing reliance on internet services and applications enabling the management of personal information from anywhere, there has been increase in application and data complexity which led to the multi-tiered design of systems where web servers run the front end logic of the application and data stored/fetched from database or file servers. Since there has been huge surge in data theft, these applications over the years has become more vulnerable to intrusions. To identify the intrusions in a more accurate manner we implemented Double Guard using Apache web server and My SQL with lightweight virtualization. With the system in place over a period of few weeks, we were able to expose a wide range of attacks with 100% accuracy in both static and dynamic web services.

KEYWORDS*:* Multi-tiered, virtualization

I.INTRODUCTION

Today's users are not aware of how much web applications and web services they use in day to day activity for past few years. Activity such as banking, travel booking, entertainment ticket booking and social networking are done through web on daily basis. Such services typically employ a web server front end that runs the application user interface logic, as well as a back-end server that consists of a database or file server. Due to their ubiquitous use for personal and/or corporate data, web services have always been the target of attacks. These attacks have become more diverse as attention has shifted from attacking the front end to exploiting vulnerabilities of the web applications in order to corrupt the back-end database system (e.g., SQL injection attacks). There is very little work being performed on multi-tiered systems that generate models of network behaviour for both web and database network interactions. In such multi-tiered architectures, the back-end database server is often protected behind a firewall while the web servers are remotely accessible over the Internet. Unfortunately, though they are protected from direct remote attacks, the back-end systems are susceptible to attacks that use web requests as a means to exploit the back end. To protect multi-tiered web services, Intrusion detection systems have been widely used to detect known attacks by matching misused traffic patterns or signatures. A class of IDS that leverages machine learning can also detect unknown attacks by identifying abnormal network traffic that deviates from the so-called "normal" behaviour previously profiled during the IDS training phase. Individually, the web IDS and the database IDS can detect abnormal network traffic sent to either of them. However, we found that these IDSs cannot detect cases wherein normal traffic is used to attack the web server and the database server. For example, if an attacker with non-admin privileges can log in to a web server using normal-user access credentials, he/she can find a way to issue a privileged database query by exploiting vulnerabilities in the web server. Neither the web IDS nor the database IDS would detect this type of attack since the web IDS would merely see typical user login traffic and the database IDS would see only the normal traffic of a privileged user. This type of attack can be readily detected if the database IDS can identify that

ISSN(Online) :2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 9, September 2015

a privileged request from the web server is not associated with user-privileged access. Unfortunately, within the current multithreaded web server architecture, it is not feasible to detect or profile such causal mapping between web server traffic and DB server traffic since traffic cannot be clearly attributed to user sessions.

Here we present a system used to detect attacks in multi-tiered web services. Our approach can create normality models of isolated user sessions that include both the web front-end (HTTP) and back-end (File or SQL) network transactions. To achieve this, we employ a lightweight virtualization technique to assign each user's web session to a dedicated container, an isolated virtual computing environment. We use the container ID to accurately associate the web request with the subsequent DB queries. Our system can build a causal mapping profile by taking both the web server and DB traffic into account.

We have implemented our system container architecture using Open VZ, and performance testing shows that it has reasonable performance overhead and is practical for most web applications. When the request rate is moderate (e.g., under 110 requests per second), there is almost no overhead in comparison to an unprotected vanilla system. Even
in a worst case scenario when the server was already overloaded, we observed only 26 percent performance overhead. The container-based web architecture not only fosters the profiling of causal mapping, but it also provides an isolation that prevents future session-hijacking attacks. Within a lightweight virtualization environment, we ran many copies of the web server instances in different containers so that each one was isolated from the rest. As ephemeral containers can be easily instantiated and destroyed, we assigned each client session a dedicated container so that, even when an attacker may be able to compromise a single session, the damage is confined to the compromised session; other user sessions remain unaffected by it.

Using our prototype, we show that, for websites that do not permit content modification from users, there is a direct causal relationship between the requests received by the front-end web server and those generated for the database back end. In fact, we show that this causality-mapping model can be generated accurately and without prior knowledge of web application functionality. Our experimental evaluation, using real-world network traffic obtained from the web and database requests of a large centre, showed that we were able to extract 100 percent of functionality mapping by using as few as 35 sessions in the training phase. Of course, we also showed that this depends on the size and functionality of the web service or application. However, it does not depend on content changes if those changes can be performed through a controlled environment and retrofitted into the training model. We refer to such sites as "static" because, though they do change over time, they do so in a controlled fashion that allows the changes to propagate to the sites' normality models.

II.RELATED WORKS

Intrusion alerts correlation [14] provides a collection of components that transform intrusion detection sensor alerts into succinct intrusion reports in order to reduce the number of replicated alerts, false positives, and non relevant positives. It also fuses the alerts from different levels describing a single attack, with the goal of producing a succinct overview of security-related activity on the network. It focuses primarily on abstracting the low-level sensor alerts and providing compound, logical, high-level alert events to the users. Double Guard differs from this type of approach that correlates alerts from independent IDSs. Rather, Double- Guard operates
on multiple feeds of network traffic using a single IDS that looks across sessions to produce an alert without correlating or summarizing the alerts produced by other independent IDSs.

An IDS such as in [12] also uses temporal information to detect intrusions. Double Guard, however, does not correlate events on a time basis, which runs the risk of mistakenly considering independent but concurrent events as correlated events. Double Guard does not have such a limitation as it uses the container ID for each session to causally map the related events, whether they be concurrent or not.

In addition to this static website case, there are web services that permit persistent back-end data modifications. These services, which we call dynamic, allow HTTP requests to include parameters that are variable and depend on user input. Therefore, our ability to model the causal relationship between the front end and back end is not always deterministic and depends primarily upon the application logic. For instance, we observed that the backend queries can vary based on the value of the parameters passed in the HTTP requests and the previous application state. Sometimes, the

ISSN(Online) :2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 9, September 2015

same application's primitive functionality (i.e., accessing a table) can be triggered by many different web pages. Therefore, the resulting mapping between web and database requests can range from one to many, depending on the value of the parameters passed in the web request.

To address this challenge while building a mapping model for dynamic web pages, we first generated an individual training model for the basic operations provided by the web services. We demonstrate that this approach works well in practice by using traffic from a live blog where we progressively model nine operations. Our results show that we were able to identify all attacks, covering more than 99 percent of the normal traffic as the training model is refined.

III.SYSTEM ARCHITECTURE

We initially set up our threat model to include our assumptions and the types of attacks we are aiming to protect against[1]-[4]. We assume that both the web and the database servers are vulnerable. In our design, we make use of lightweight process containers, referred to as "containers," as ephemeral, disposable servers for client sessions. It is possible to initialize thousands of containers on a single physical machine, and these virtualized containers can be discarded, reverted, or quickly reinitialized to serve new sessions. A single physical web server runs many containers, each one an exact copy of the original web server. Our approach dynamically generates new containers and recycles used ones. As a result, a single physical server can run continuously and serve all web requests.

In Double Guard, the new container-based web server architecture enables us to separate the different information flows by each session. This provides a means of tracking the information flow from the web server to the database server for each session[5][6]. Our approach also does not require us to analyze the source code or know the application logic. For the static webpage, our Double Guard approach does not require application logic for building a model. However, as we will discuss, although we do not require the full application logic for dynamic web services, we do need to know the basic user operations in order to model normal behaviour.

CLAMP [13]] is an architecture for preventing data leaks even in the presence of attacks. By isolating code at the web server layer and data at the database layer by users, CLAMP guarantees that a user's sensitive data can only be accessed by code running on behalf of different users. In contrast, Double Guard focuses on modelling the mapping patterns between HTTP requests and DB queries to detect malicious user sessions. There are additional differences between these two in terms of requirements and focus. CLAMP requires modification to the existing application code, and the Query Restrictor works as a proxy to mediate all database access requests. Moreover, resource requirements and overhead differ in order of magnitude: Double Guard uses process isolation whereas CLAMP requires platform virtualization, and

CLAMP provides more coarse-grained isolation than Double Guard. However, Double Guard would be ineffective at detecting attacks if it were to use the coarse grained isolation as used in CLAMP. Building the mapping model in Double Guard would require a large number of isolated web stack instances so that mapping patterns would appear across different session instances[14].

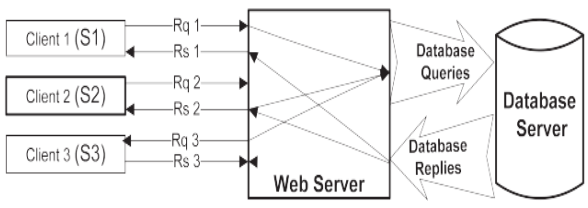


Fig. 1. Classic three-tier model.

ISSN(Online) :2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 9, September 2015

The web server acts as the front end, and database as the content storage back end.Fig. 1 illustrates the classic three-tier model. At the database side, we are unable to tell which transaction corresponds to which client request. The communication between the web server and the database server is not separated, and we can hardly understand the relationships among them.

A.Building the Normality Model

This container-based and session-separated web server architecture not only enhances the security performances but also provides us with the isolated information flows that are separated in each container session[15]. It allows us to identify the mapping between the web server requests and the subsequent DB queries, and to utilize such a mapping model to detect abnormal behaviours on a session/client level. In typical three-tiered web server architecture, the web server receives HTTP requests from user clients and then issues SQL queries to the database server to retrieve and update data. These SQL queries are causally dependent on the web request hitting the web server. We want to model such causal mapping relationships of all legitimate traffic so as to detect abnormal/attack traffic[7]-[10].

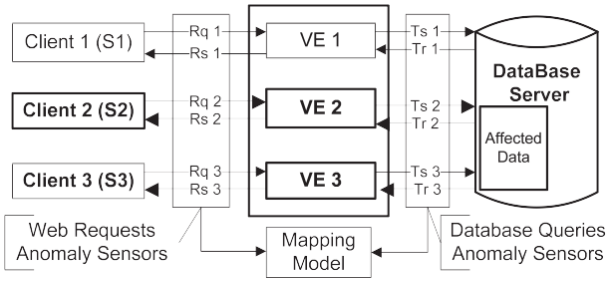


Fig. 2. Web server instances running in containers

Fig. 2 depicts how communications are categorized as sessions and how database transactions can be related to a corresponding session. According to Fig. 1, if Client 2 is malicious and takes over the web server, all subsequent database transactions become suspect, as well as the response to the client. By contrast, according to Fig. 2, Client 2 will only compromise the VE 2, and the corresponding database transaction set T2 will be the only affected section of data within the database.

Once we build the mapping model, it can be used to detect abnormal behaviours. Both the web request and the database queries within each session should be in accordance with the model. If there exists any request or query that violates the normality model within a session, then the session will be treated as a possible attack[11].

IV.OPEN VZ

Nowadays data are more valuable and hence database should receive the highest level of protection. Therefore, significant research efforts have been made on database IDS and database firewalls[16]. These softwares, such as Green SQL, work as a reverse proxy for database connections. Instead of connecting to a database server, web applications will first connect to a database firewall. SQL queries are analysed; if they're deemed safe, they are then forwarded to the back-end database server. The system proposed in composes both web IDS and database IDS to achieve more accurate detection, and it also uses a reverse HTTP proxy to maintain a reduced level of service in the presence of false positives. However, we found that certain types of attack utilize normal traffics and cannot be detected by either the web IDS or the database IDS. In such cases, there would be no alerts to correlate. In our system, the new container-based web server architecture enables us to separate the different information flows by each session. This provides a means of tracking the information flow from the web server to the database server for each session[17].

Virtualization is used to isolate objects and enhance security performance. Full virtualization and Para-virtualization are not the only approaches being taken. An alternative is a lightweight virtualization, such as Open VZ,

ISSN(Online) :2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 9, September 2015

Parallels Virtuoso, or Linux-V Server. In general, these are based on some sort of container concept. With containers, a group of processes still appears to have its own dedicated system, yet it is running in an isolated environment. On the other hand, lightweight containers can have considerable performance advantages over full virtualization or Para-virtualization. in our system, we utilized the container ID to separate session traffic as a way of extracting and identifying causal relationships between web server requests and database query events.

It is possible to initialize thousands of containers on a single physical machine, and these virtualized containers can be discarded, reverted, or quickly reinitialized to serve new sessions. A single physical web server runs many containers, each one an exact copy of the original web server. Our approach dynamically generates new containers and recycles used ones. As a result, a single physical server can run continuously and serve all web requests. However, from a logical perspective, each session is assigned to a dedicated web server and isolated from other sessions. Since we initialize each virtualized container using a read-only clean template, we can guarantee that each session will be served with a clean web server instance at initialization. We choose to separate communications at the session level so that a single user always deals with the same web server. Sessions can represent different users to some extent, and we expect the communication of a single user to go to the same dedicated web server, thereby allowing us to identify suspect behaviour by both session and user. If we detect abnormal behaviour in a session, we will treat all traffic within this session as tainted. If an attacker compromises a vanilla web server, other sessions' communications can also be hijacked. In our system, an attacker can only stay within the web server containers that he/she is connected to, with no knowledge of the existence of other session communications. We can thus ensure that legitimate sessions will not be compromised directly by an attacker.

V.ATTACK SCENARIOS

Our system is effective at capturing the following types of attacks:

A.Privilege Escalation Attack

Let's assume that the website serves both regular users and administrators. For a regular user, the web request ru will trigger the set of SQL queries Qu; for an administrator, the request ra will trigger the set of admin level queries Qa. Now suppose that an attacker logs into the web server as a normal user, upgrades his/her privileges, and triggers admin queries so as to obtain an administrator's data. This attack can never be detected by either the web server IDS or the database IDS since both ru and Qa are legitimate requests and queries. Our approach, however, can detect this type of attack since the DB
query Qa does not match the request ru, according to our mapping model.

B.Hijack Future Session Attack

This class of attacks is mainly aimed at the web server side. An attacker usually takes over the web server and therefore hijacks all subsequent legitimate user sessions to launch attacks. For instance, by hijacking other user sessions, the attacker can eavesdrop, send spoofed replies, and/or drop user requests. A session-hijacking attack can be further categorized as a Spoofing/Man-in-the-Middle attack, an Ex filtration Attack, a Denial-of-Service/Packet Drop attack, or a Replay attack. As each user's web requests are isolated into a separate container, an attacker can never break into other users' sessions.

C.Injection Attack

Attacks such as SQL injection do not require compromising the web server. Attackers can use existing vulnerabilities in the web server logic to inject the data or string content that contains the exploits and then use the web server to relay these exploits to attack the back-end database. Since our approach provides a two-tier detection, even if the exploits are accepted by the web server, the relayed contents to the DB server would not be able to take on the expected structure for the given web server request. For instance, since the SQL injection attack changes the structure of the SQL queries, even if the injected data were to go through the web server side, it would generate SQL queries in a different

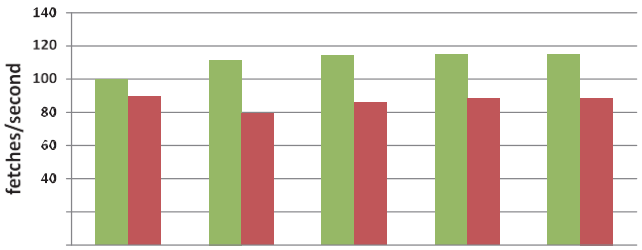
ISSN(Online) :2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 9, September 2015

structure that could be detected as a deviation from the SQL query structure that would normally follow such a web request.



D.Direct DB Attack

It is possible for an attacker to bypass the web server or firewalls and connect directly to the database. An attacker could also have already taken over the web server and be submitting such queries from the web server without sending web requests. Without matched web requests for such queries, a web server IDS could detect neither. Furthermore, if these DB queries were within the set of allowed queries, then the database IDS itself would not detect it either. However, this type of attack can be caught with our approach since we cannot match any web requests with these queries.

VI.PERFORMANCE EVALUATION

We implemented a prototype of Double Guard using a web server with a back-end DB. We also set up two testing websites, one static and the other dynamic. To evaluate the detection results for our system, we analyzed four classes of attacks.

This was deployed as part of our centre website in production environment and served 52 unique web pages. For our analysis, we collected real traffic to this website for more than two weeks and obtained 1,172 user sessions. To test our system in a dynamic website scenario, we set up a dynamic Blog using the Word press blogging software. In our deployment, site visitors were allowed to read, post, and comment on articles. All models for the received front-end and back-end traffic were generated.

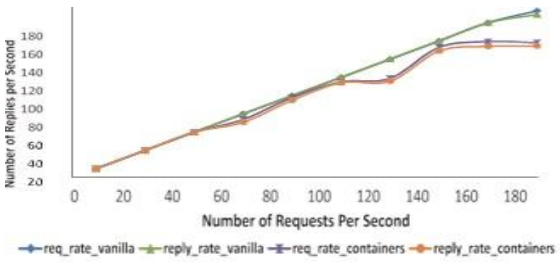


Fig. 4. Performance evaluation using auto bench.

For the http_ load evaluation, we used the rate of five (i.e., it emulated five concurrent users). We tested under the parameters of 100, 200, and 400 total fetches, as well as 3 and 10 seconds of fetches. For example, in the 100-fetches

ISSN(Online) :2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 9, September 2015

benchmark, http_ load fetches the URLs as fast as it can 100 times. Similarly, in the 10-seconds benchmark, http _load fetches the URLs as fast as it can during the last 10 seconds. We picked 15 major URLs of the website and tested them against both servers. Fig. 3 shows our experiment results.

Fig. 4 shows that when the rate was less than 110 concurrent sessions per second, both servers could handle requests fairly well. Beyond that point, the rates in the container-based server showed a drop: for 150 sessions per second, the maximum overhead reflected in the reply rate was around 21 percent (rate of 130). Notice that 21 percent was the worst case scenario for this experiment, which is fairly similar to 26.2 percent in the http _load experiment.

When the server was not overloaded, and for our server this was represented by a rate of less than 110 concurrent sessions per second, the performance overhead was negligible.

VII.CONCLUSIONS

We presented an intrusion detection system that builds models of normal behaviour for multi-tiered web applications from both front-end web (HTTP) requests and back-end database (SQL) queries. Unlike previous approaches that correlated or summarized alerts generated by independent

IDSs, system forms a container-based IDS with multiple input streams to produce alerts. We have shown that such correlation of input streams provides a better characterization of the system for anomaly detection because the intrusion sensor has a more precise normality model that detects a wider range of threats. We achieved this by isolating the flow of information from each web server session with a lightweight virtualization. Furthermore, we quantified the detection accuracy of our approach when we attempted to model static and dynamic web requests with the back-end file system and database queries. For static websites, we built a well-correlated model, which our experiments proved to be effective at detecting different types of attacks. Moreover, we showed that this held true for dynamic requests where both retrieval of information and updates to the back-end database occur using the web server front end. When we deployed our prototype on a system that employed Apache web server, a blog application, and a My SQL back end, system was able to identify a wide range of attacks with minimal false positives. As expected, the number of false positives depended on the size and coverage of the training sessions we used. Finally, for dynamic web applications, we reduced the false positives to 0.6 percent.

REFERENCES

1. SANS, “The Top Cyber Security Risks,” http://www.sans.org/top-cyber-security-risks/,2011.
2. National Vulnerability Database, “Vulnerability SummaryCVE-2010-4332,” http://web.nvd.nist.gov/view/vuln/detail?vulnId= CVE-2010-4332, 2011.
3. Selva Kumar S., Ram Krishna Rao M., Deepak Kumar R., Panwar S., Prasad C.S., "Biocontrol by plant growth promoting rhizobacteria against black scurf and stem canker disease of potato caused by Rhizoctonia solani", Archives of Phytopathology and Plant Protection, ISSN : 0323-5408, 46(4) (2013) pp.487-502.
4. National Vulnerability Database, “Vulnerability Summary for CVE-2010-4333,” http://web.nvd.nist. gov/view/vuln/detail? vulnId=CVE-2010-4333, 2011.
5. Subha Palaneeswari M., Abraham Sam Rajan P.M., Silambanan S., Jothimalar, "Blood lead in end-stage renal disease (ESRD) patients who were on maintainence haemodialysis", Journal of Clinical and Diagnostic Research, ISSN : 0973 - 709X, 6(10) (2012) pp.1633-1635.
6. Auto bench, http://www.xenoclast.org/autobench/, 2011..
7. Sukumaran V.G., Bharadwaj N., "Ceramics in dental applications", Trends in Biomaterials and Artificial Organs, ISSN : 0971-1198, 20(1) (2006) pp.7-11.
8. “VirtuozzoContainers,”http://www.parallels.com/products/pvc45/, 2011.
9. Selva Kumar S., Ram Krishna Rao M., Balasubramanian M.P., "Chemopreventive effects of Indigofera aspalathoides on 20-methylcholanthrene induced fibrosarcoma in rats", International Journal of Cancer Research, ISSN : ISSN: 1811-9727, 7(2) (2011) pp.144-151.
10. “Wordpress,” http://www.wordpress.org/, 2011.
11. “WordpressBug,”http://core.trac.wordpress.org/ticket/5487,2011.
12. C.Anley, “Advanced Sql Injection in Sql Server Applications,” technical report, Next Generation Security Software, Ltd., 2002.

ISSN(Online) :2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 9, September 2015

13. B.Parno, J.M. McCune, D. Wendlandt, D.G. Andersen, and A. Perrig, "CLAMP: Practical Prevention of Large-Scale Data Leaks," Proc. IEEE Symp. Security and Privacy, 2009.
14. B.I.A. Barry and H.A. Chan, "Syntax, and Semantics-Based Signature Database for Hybrid Intrusion Detection Systems," Security and Comm. Networks, vol. 2, no. 6, pp. 457-475, 2009.
15. D.Bates, A. Barth, and C. Jackson, "Regular Expressions Considered Harmful in Client-Side XSS Filters," Proc. 19th Int'l Conf. World Wide Web, 2010.
16. M. Christo dorescu and S. Jha, "Static Analysis of Executables to Detect Malicious Patterns," Proc. Conf. USENIX Security Symp.,2003.
17. A.Seleznyov and S. Puuronen, "Anomaly Intrusion Detection Systems: Handling Temporal Relations between Events," Proc.
18. Dr.K.P.Kaliyamurthie, D.Parameswari, Load Balancing in Structured Peer to Peer Systems, International Journal of Innovative Research in Computer and Communication Engineering, ISSN: 2249-2615,pp 22-26, Volume1 Issue 1 Number2-Aug 2011
19. Dr.R.Udayakumar, Addressing the Contract Issue,Standardisation for QOS, International Journal of Innovative Research in Computer and Communication Engineering, ISSN (Online): 2320 – 9801,pp 536-541, Vol. 1, Issue 3, May 2013
20. Dr.R.Udayakumar, Computational Modeling of the StrengthEvolution During Processing And Service Of9-12% Cr Steels, International Journal of Innovative Research in Computer and Communication Engineering, ISSN(Online): 2320-9801,pp 3295-3302, Vol. 2, Issue 3, March 2014
21. P.Gayathri, Assorted Periodic Patterns Intime Series Database Usingmining, International Journal of Innovative Research in Computer and Communication Engineering, ISSN(Online): 2320-9801, pp 5046- 5051, Vol. 2, Issue 7, July 2014.
22. Gayathri, Massive Querying For Optimizing Cost – CachingService in Cloud Data, International Journal of Innovative Research in Computer and Communication Engineering, ISSN(Online): 2320-9801,pp 2041-2048, Vol. 1, Issue 9, November 2013