

A Novel System Defined Network Solution for User Legitimacy Assurance in the Docker Containers

Pinkpreet Kaur¹, Anupama Gupta², Er. Harpreet Kaur³

Computer Science and Engineering, LLRIET, Moga, India^{1,3}

Computer Science and Engineering, GZSCCET, Bathinda, India²

Abstract: The aim of the work is to provide an overview of virtualization methods, while focusing on Docker. This new virtualization stage, which incredibly contrasts from the customary, virtual machine based way to deal with virtualization, has started gigantic enthusiasm for Linux holders. Accordingly, numerous Docker based activities have risen extending from straightforward order line apparatuses to whole working frameworks. To pick up a more profound comprehension of Docker's inside components, the proposed work shows working with Docker's Remote API. In this work, an attempt has been made to address the imperative security issues of the Docker compartments. Additionally, security calculations and strategies have been proposed to address DoS attacks related issues in the Docker holder innovation. The preparatory investigations and testing of the security strategies are promising. Results show that the Validation, Authentication and security is noticeably increased in proposed model as comparison to existing model.

Keywords: Information Security; Docker Technology; User Authentication; AES; Cryptography

1. INTRODUCTION

The web in straightforward terms is characterized as an interconnected arrangement of computer systems. The extent of web in day to day life is immeasurable. It provides an extensive variety of data, administrations and assets which enable all areas to be connected all around. As the need of web is emerging with time, different issues identified with its security comes in to existence. The purpose behind web frailty is fundamentally its plan in light of the fact that the preeminent concern was its usefulness instead of its security. The security of web is a major concern these days. The issues identified with web security are confirmation, respectability, accessibility, classification and non-repudiation. In this paper, fundamental core interest is on weakness to accessibility. The accessibility implies that the data, the processing frameworks, and the security controls are all available and operable in conferred state at a few irregular purposes of time (Tipton and Krause, 2004). Among all security attacks, the Distributed Denial of Service (DDoS) attacks are those which prevent customers and clients to get to all points of interest of administrations accessible to them from server side. DDoS attack brings about long framework timeouts, lost incomes, expansive volumes of work to recognize attacks and to plan sufficient reaction measures (Sardana and Joshi, 2009). The Denial of Services (DoS) attack is Distributed Denial of Services (DDoS) attack since it is propelled simultaneously to various machines. DDoS attacks are not new unsettling influence to web, they returned late in August 1999 and after that relentlessly their seriousness is developing. Some perceived DoS attacks are SYN Flood, tear, smurf, ping of death (Sachdeva et al. 2010). There have been expansive scale assaults focusing on numerous prominent sites (Yuan and Mills, 2005). Fig. 1 describes the concept of DDOS attacks.

These locales incorporate twitter, facebook, Amazon and so forth. (Mirkovic et al. 2004). In any case, the most widely recognized type of DDoS attacks is a parcel flooding attack, in which a substantial number of apparently honest to goodness TCP, User Datagram Protocol (UDP), or Internet Control Message Protocol (ICMP) bundles are coordinated to a particular goal. DDoS attacks can't be distinguished and halted effortlessly in light of the fact that manufactured source addresses and other strategies are utilized to disguise assault sources (Kumar, Joshi and Singh, 2006).

According to Peng et al. 2007, assurance against these assaults is testing for mostly two reasons. To start with, the quantity of zombies included in a DDoS attack is substantial and misuse of these zombies traverses substantial land zones. The volume of movement sent by a solitary zombie may be little, however the volume of collected activity touching base at the casualty host is overpowering. Second, zombies ordinarily parody their IP addresses under the control of assailant, which makes it extremely hard to follow the attack movement back even to zombies. In this paper, an outline of DDoS attack and different counteractive action and alleviation systems for DDoS attacks alongside their favorable circumstances and weaknesses is talked about.

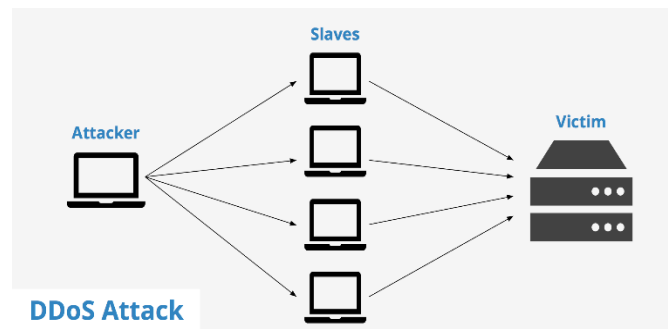


Figure 1: DDoS Attack

DDoS attack does not depend on specific system configuration or framework shortcoming. It essentially abuses the enormous asset asymmetry between the Internet and the casualty. Since Web engineering is open in nature, any machine appended to it is publicly noticeable to another machine connected to empower the correspondence. The programmer or aggressor group takes the unfortunate preferred standpoint of this open nature to find any shaky machine associated with the Internet. The found machine is in this way tainted with the attack code. The tainted machine can further be utilized to find and contaminate another machine associated etc. The aggressor subsequently progressively prepares another attack. Contingent on the assaulting code, the programmers send control directions to aces which thus control specialists. The zombies under the control of aces transmit assault bundles which focalize at casualty to debilitate its assets.

DDoS attack essentially focuses on casualty's computational or communicational assets, for example, data transmission, memory, CPU cycle, record descriptors and cushions and so forth. The enlist stage is first stage in event of DDoS attack in which the assailant finds the powerlessness in the casualty framework and select different specialists, these numerous operators additionally called as bots or zombie. These zombies frame a botnet including all such arranged machines which are dependable to run attack code under normal charge what's more, control by the aggressor. The second stage is the adventure stage in which the weakness is misused in the selected zombies. In the third stage, known as contaminate stage, bots are tainted with the attack code. Last stage is the utilization stage which utilizes the specialists to send the attack code to casualty framework. Arora et al. (2011) have classified DDoS attacks into two general classes: flooding attacks and helplessness attacks. Flooding DDoS attacks devour assets, for example, arrange transmission capacity. These attacks surge the system with such a high volume of movement which expends their accessible system assets and authentic client demands can't get past, bringing about debased efficiency. Powerlessness assaults utilize the normal conduct of conventions, for example, TCP and HTTP to the assailant's scope.

Network assaults surge a PC with such a high volume of association solicitations, that all accessible working framework assets are devoured what's more, the PC can no longer procedure honest to goodness client demands. Nonetheless, assets of interfacing system are not an issue in the event of business servers as these are facilitated by the ISPs. Be that as it may, server assets, for example, handling limit, cushion restrain and so on., are put under worry by surge of apparently genuine solicitations produced by DDoS attack zombies. Each ask for devours some CPU cycles. Once the add up to demand rate is more than the administration rate of server, the demands begin getting cradled in the server, and after a few times because of support over run, approaching solicitations are dropped. The clog and stream control signals compel true blue customers to diminish their rate of sending solicitations, though assault parcels continue coming. At last, a phase comes when as it were assault movement is coming to at the server. In this way, administration is denied to honest to goodness customers. Carlson et al. (1998) expressed that as assault quality develops by utilizing various sources, the computational necessities of sifting activity of malevolent streams turn into a weight at the objective. The present work shows how the usage of Docker has improved the security of the communication. The work is evaluated using various performance metrics and a valid comparison is made between the existing work and the proposed work.

Reference architectures (RA) in the field of software architecture or enterprise architecture can be seen as a template for a particular system domain. It provides a common vocabulary with which implementations are discussable and commonalities being identifiable. An RA generalizes a set of solutions Kratzke (2014). These solutions are structured for the depiction of one or more concepts based on the harvesting of a set of patterns that have been observed in a number of successful implementations. The fastest computers listed in the Top 500 are able to execute 1016 FLOPS. The next generation of computer clusters will move into new dimensions and be a hundred times faster. Such exascale computers will not have significantly more nodes, but considerably more cores per node. It is predicted that this increase of CPU performance will not be matched by other resources resulting in an imbalance between CPU performance on the one hand and I/O performance on the other hand Pickartz et al. (2014).

2. RELATED WORK

Ranjan (2014) proposed that there are a few lacks in the current Platform as a service (PaaS) stage. PaaS stage of the application facilitating condition is single and give just the working condition specific programming dialect or scripting dialect. Furthermore, the parts of PaaS stage are shut. Finally, virtual machines devour over the top assets. So, the PaaS stage proposed web applications will concentrate on and address the accompanying issues. The stage ought to give the runtime condition to an assortment of uses. It not just backings prevalent programming and scripting dialects and furthermore give more grounded similarity and more adaptable working condition. So, the virtual machine is additionally given as a runtime situation to application. Furthermore, the stage cannot just give the capacity that giving an open get together system to clients and enable an outsider to give abilities considering this stage. Bansal and Kaur (2015) proposed the light weight virtualization techniques for the issue of mapping of customers' demand to a particular web server and tie the solicitations originating from customers to port 80 with 8081 and 8082 and furthermore utilized apparatuses for contrasting the execution of linux compartments and virtual machines. This method contained various pivotal focal points like live movement, security and so forth.

Yang (2015) proposed that containers are convenient, the applications can be packaged in to a solitary unit and can be conveyed to different situations without rolling out any improvements to the holder. They utilized some benchmark apparatuses for measuring the quality and shortcomings on various stages as far as capacity, handling and system. Their outcomes demonstrate that holders by and large accomplish preferred execution over hypervisors. They likewise presented how docker is not quite the same as linux holders. They examined and measured boot speed and CPU execution of every stage and said Docker doesn't squander CPU assets and capacity is little, and boot-time, time of creating and circulating pictures are short in docker which is the benefit of utilizing docker in cloud. Docker can be utilized with the stage as an administration or application organization to lessen the overhead of system administrator/framework administrator. Anderson et al. (2015) discussed that OS level based virtualization is an approach in which working arrangement of the host is imparted to the virtual example. This implies the host bit is imparted to the virtual example and that virtual occasion is known as compartments, that is the reason occasionally it is known as containerization or lightweight virtualization.

Different apparatuses are utilized to accomplish lightweight virtualization, for example, OpenVZ , Linux holders (LXC), Docker, Rocket and Kubernetes. Authors have utilized Docker for execution and keeping up the servers on a similar host. Docker is a lightweight virtualization apparatus which is utilized for the application facilitating in a cloud domain. Different cloud suppliers utilized docker for giving administrations to purchasers, for example, Platform as a Service (PaaS) and Software as a Service (SaaS). Joy (2015) presented the design for the usage of servers utilizing lightweight virtualization utilized Docker. Two web servers are executed on docker with a port tie strategy, in this Webserver-1's port 80 is tied with the host port 8081 and Webserver-2's port 80 is tied with 8082 port of host and both the servers are taking volume from the host i.e. an index on the host is mounted as volume for the servers. Yu and Huan (2015) suggested that the lightweight virtualization procedure has influenced virtual machines more compact, to work more effective and less demanding to administration. The season of live relocation can be enormously lessen when connected to compartments in view of the lightweight virtualization system, which is extremely valuable to server farms. This exhibited a logging and replay approach for live relocation of Docker compartments. With this approach, both the down time and aggregate movement time are lessened. Yin and Shang (2015) proposed research and usage PaaS stage in light of Docker. PaaS is a type of distributed computing administration assets and gives application improvement condition. As the current PaaS stage facilitating and virtual machine condition disentanglement of the issue of unnecessary utilization of assets, they have directed inside and out research on the PaaS stage. Authors proposed a component to make PaaS stage in light of Docker.

The complexity of cloud-based analytics environments threatens to undermine their otherwise tremendous values. In particular, configuring such environments presents a great challenge. Zhang et al. (2015) alleviate this issue with an engine that recommends configurations for a newly submitted analytics job in an intelligent and timely manner. The engine is rooted in a modified k-nearest neighbor algorithm, which finds desirable configurations from similar past jobs that have performed well and apply the method to configuring an important class of analytics environments: Hadoop on container-driven clouds. Preliminary evaluation suggests up to 28% performance gain could result from this method. Alampalayam et. al (2016) discussed the idea of containerization and virtualization is getting footing in the cloud based IT conditions. Docker motor is prominent usage for disentangling and streamlining containerization innovation. IT industry understands various mechanization and increasing speed highlights and offices through the embracement of the Docker-supported containerization worldview, which is a working framework (OS) level and lightweight virtualization. However, the security issues influence the far reaching and certain utilization of Docker stage.

3. PROPOSED WORK

DoS has been a major issue in the IT world by making gainful IT frameworks and assets unusable through a cleverly planned endeavors of bulldozing the frameworks with undesirable stuffs and demands. Holders are likewise not taken off alone and subsequently there is a request for advanced answers for going around DoS attacks Sardana & Joshi (2009) . In this section, an improved strategy to address DoS attacks and actualizing the technique to show its capacity in disposing of DoS attacks have been defined.

3.1 Methodology

The various steps that are followed are as follows:

Step1: Key exchange to create the session between server and client. This would be done at pre-phase to establish the connection. Keys would be generated into a random fashion. So, it would be non-predictable.

Step2: After step1, the container data would be stored in encrypted format with private key. So, the unauthorized user could not be able to read the data.

Step3: The files would be in read-only format which the recommended process requires to secure the containers content.

3.2 Work Flow

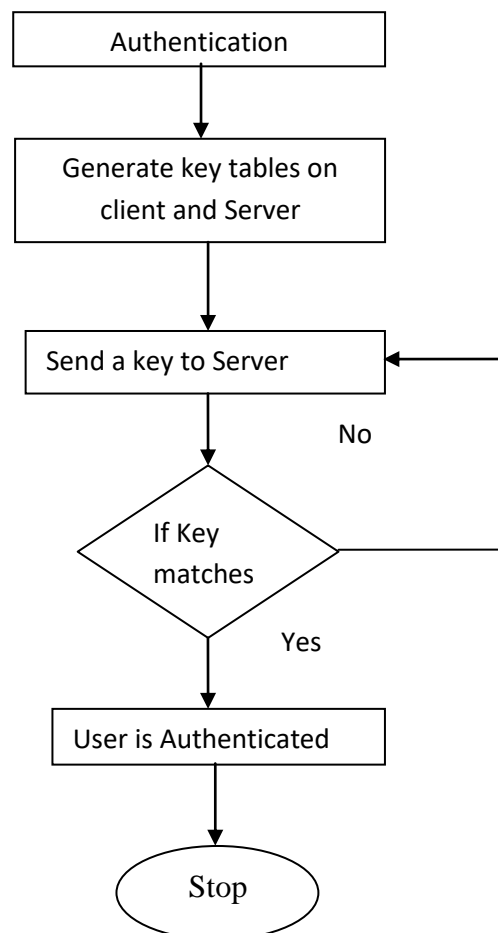


Figure2: Flow chart showing the process of authentication

The proposed workflow has been demonstrated with the help of flowcharts. First flowchart shown in Figure 2 depicts the process of authentication. In this process, key table is generated on both client and server side with random key generator. User tries to create a session after which the server demands key from client and client sends any key from key table. Then, the server matches that key with key table. If key matches, the user is authenticated. Otherwise, unauthenticated. Second flowchart (Figure 3) presents the concept of Docker and AES encryption process. This security process is done on server side.

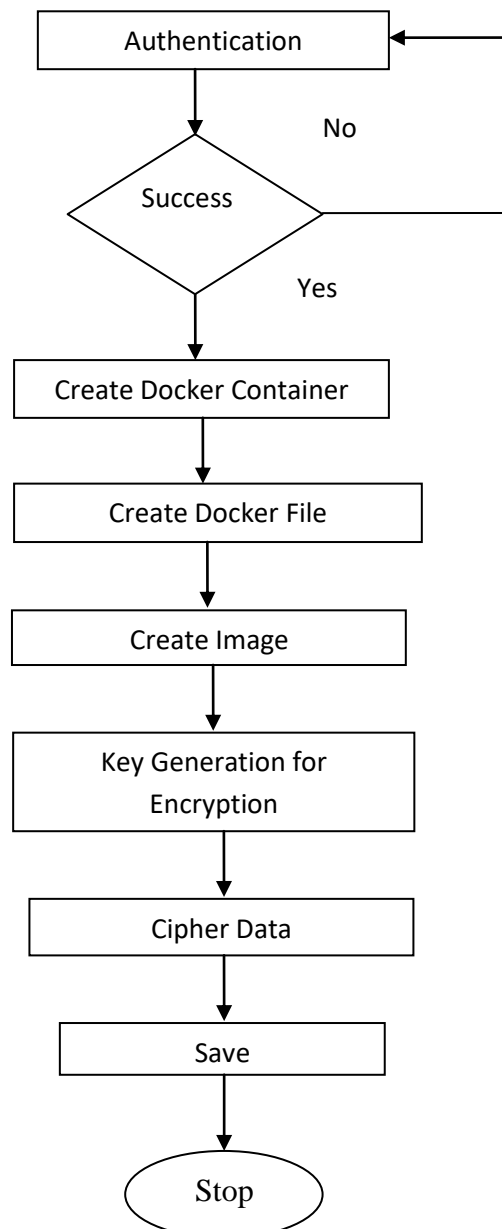


Figure 3: Flow chart showing the process of Docker and AES Encryption

3.3 Proposed Algorithm

Algorithm 1: Randomized Function to generate random number

1. Initialize the random number generator.
2. Create a radii value for each point in the sphere. These values are in the open interval (0, 3), but are not uniformly distributed. The values are created using following mathematical equation:

$$f(x) = 3 * \int_1^{1000000} \text{random} * \frac{1}{3}$$

3. Randomly select and concatenate the values to create one time password (OTP).
4. Return OTP.

Algorithm 2: Randomized Function to generate captcha

1. Initialize the random number generator.
2. Generate the alphanumeric string from an array set.
Array = A (Set of small, capital alphabets & numeric values 0-9)
Random number = R (with unique values).
Captcha String = C with 5 values.

- C = A[R, (A-Z)]
- C = A[R, (a-z)]
- C = A[R, (0-9)]
- C = A[R, (A-Z)]
- C = A[R, (a-z)]
- C = [A[R, (A-Z)], A[R,(a-z)], A[R,(0-9)], A[R,(A-Z)], A[R,(a-z)]]

Algorithm 3: AES Algorithm Encryption Process

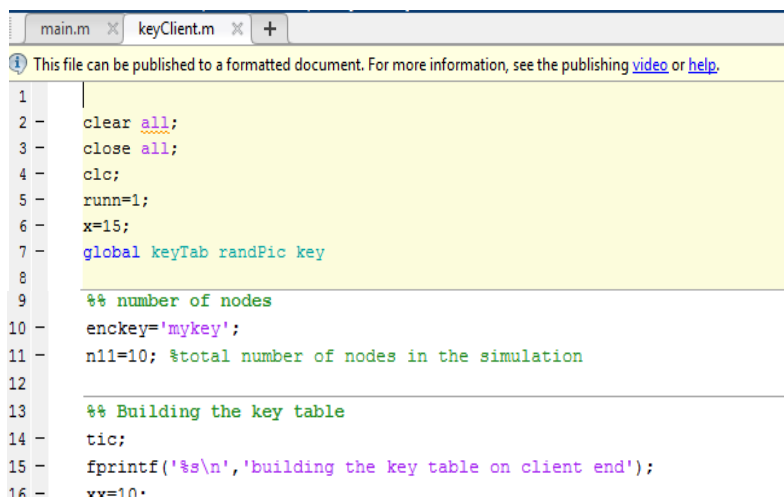
1. Input Image
2. Convert the image into Data Matrix (d)
3. Data Matrix Validation \rightarrow validate(d) \rightarrow d_M
4. Data Matrix Segmentation \rightarrow segment((d_M) \rightarrow d_mⁱ
5. Input Security Key (S_k)
6. Key Expansion(S_k)
7. Initial Round \rightarrow Add Round Key (S_k)
8. Rounds \rightarrow For Loop
 - a. Sub Bytes(d_mⁱ)
 - b. Shift Rows(d_mⁱ)
 - c. Mix Columns(d_mⁱ)
 - d. Add Round Key(d_mⁱ)
9. Rounds \rightarrow End For Loop
10. Final Round \rightarrow Mix Columns(False)
 - a. Sub Bytes(d_mⁱ)
 - b. Shift Rows (d_mⁱ)
 - c. Add Round Key(d_mⁱ)
11. Data Matrix Merger \rightarrow merge(d_mⁱ) \rightarrow dE_M
12. Data Matrix Reverse validation \rightarrow revalidate(dE_M) \rightarrow dE

4. RESULTS AND DISCUSSIONS

The proposed work has been done in MATLAB. The parameters required for simulation are listed in Table 1. The screenshot for building and encrypting key table is shown in Figure 4.

Table 1: Simulation Parameters

| Network Parameters | Values |
|----------------------|------------------------------|
| CPU | Intel Celeron 1.6 GHZ |
| RAM | 4 GB |
| HDD | 500 GB |
| Operating System | Microsoft Windows 7 (64-bit) |
| Software | MATLAB 2013 a |
| Programming Language | MATLAB |



```

1
2 - clear all;
3 - close all;
4 - clc;
5 - runn=1;
6 - x=15;
7 - global keyTab randPic key
8
9  %% number of nodes
10 - enckey='mykey';
11 - n11=10; %%total number of nodes in the simulation
12
13  %% Building the key table
14 - tic;
15 - fprintf('%s\n','building the key table on client end');
16 - xx=10;
  
```

```
Command Window
building the key table on client end
Encrypting the secure key
Key has been Encrypted
Sending secure key to Server
building the key table on server end
Decrypting the secure key
Key Match
```

Figure 4: Building and encrypting key table

The key tables are built on both client and server side and shows the encryption and decryption process with key. The client sends key from key table when server demands. If the key matches then user is authenticated. This is shown in Figure 5.

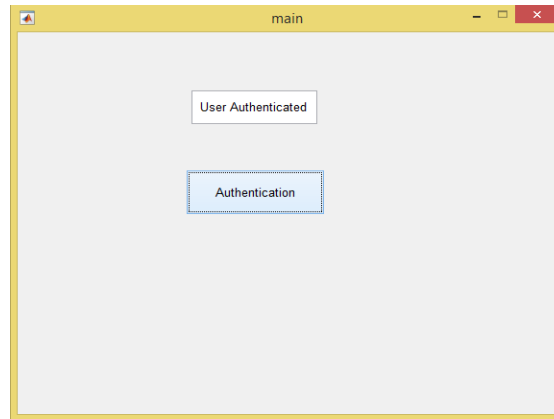


Figure 5: Authentication Process

Figure 6 shows the creation of session with Docker server. In this container, the commands are created and then written in docker file and then these commands and files are run. Subsequently, the commands are written in Docker file. This is shown in Figure 7. Key is generated for AES Cryptography as shown in Figure 8. This same key is then used for encrypting and decrypting the text.

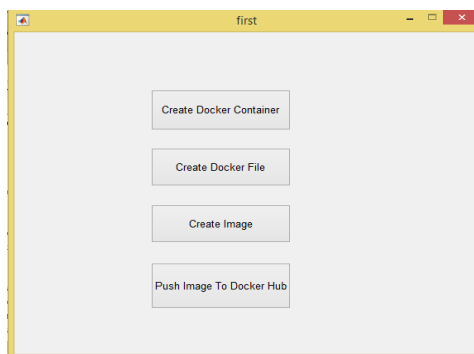
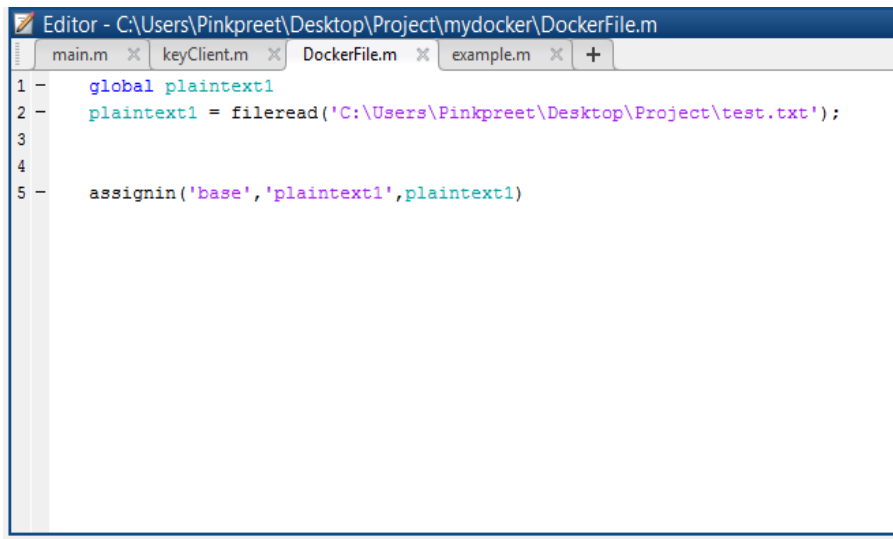


Figure 6: Create session with docker server



```
Editor - C:\Users\Pinkpreet\Desktop\Project\mydocker\DockerFile.m
main.m x keyClient.m x DockerFile.m x example.m x +
1 - global plaintext1
2 - plaintext1 = fileread('C:\Users\Pinkpreet\Desktop\Project\test.txt');
3
4
5 - assignin('base','plaintext1',plaintext1)
```

Figure 7: Docker file

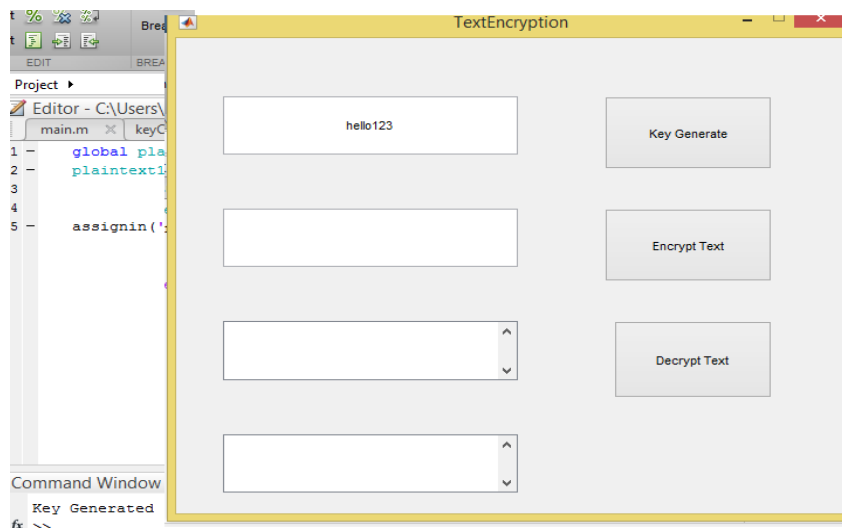
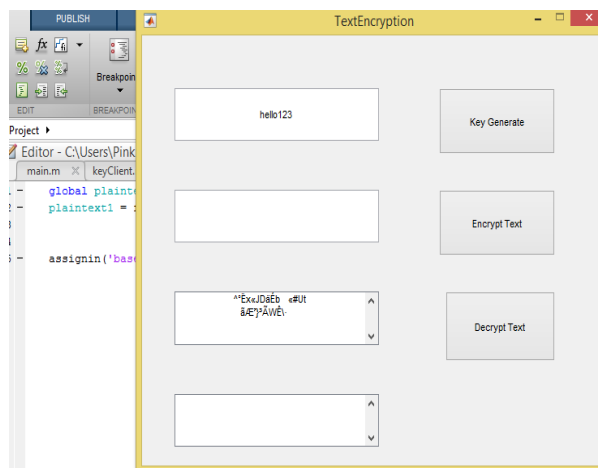


Figure 8: Key Generated for encrypting text

Text is encrypted using AES algorithm with the key that is generated in Figure 8. This encryption is done on server side. This encryption is shown in Figure 9. The text that is encrypted is that text whose path is in docker.m file.




```

Command Window
Total data size: 512.000000 kB
The Text message used : Hello World23
The Encrypted Message : ^° Èx«JDäÉb « #Ut
âÆ'}'Ä WÊ\ .
The time taken for encryption : 0.259460 seconds
Encryption speed: 1973.3 kB/s
    
```

Figure 9: Encrypting Text using AES algorithm

Text is decrypted using AES algorithm. This decryption is done on server side. The text that is encrypted in Figure 9 is decrypted with same key that is used for encrypting the text. This process is shown in Figure 10.

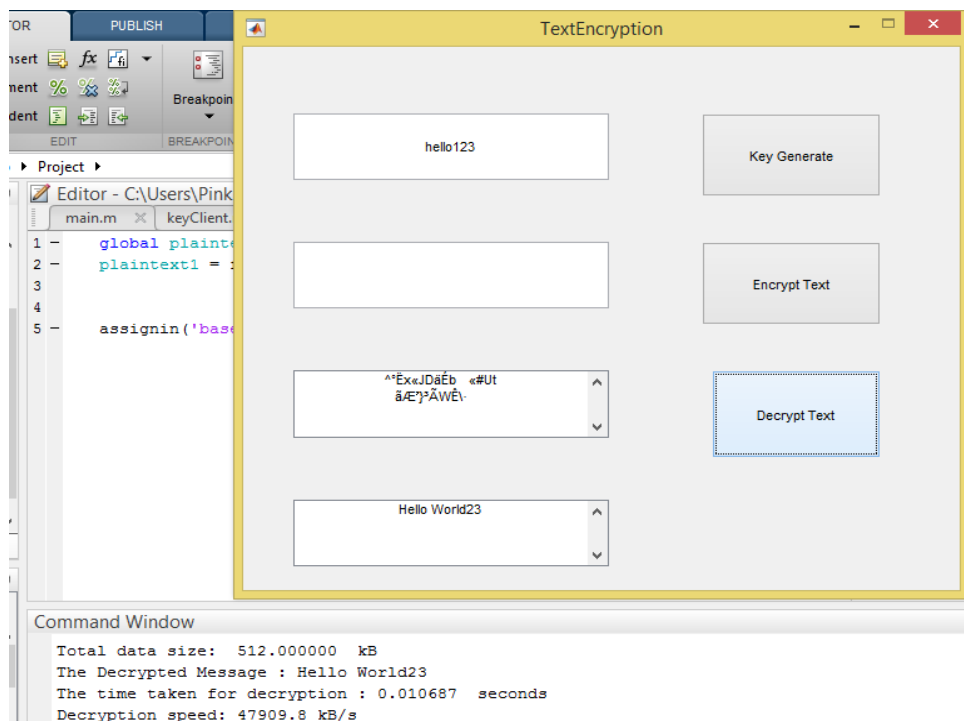


Figure 10: Decrypting Text using AES algorithm

The proposed system has been tested with the 25 random persons having age between 16 years to 42 years who logged into the system. The age variation has been counted as the factor towards testing the ease of access to the proposed authentication scheme. The number of successful attempts and success rate by the testing user set is shown in Table 2. The Bar Graph representing the login accuracy by the testing users of various age groups is shown in Figure 11. The security scheme has been tested for average Login success rate, probability of failed login attempts, choice of features, etc.

Table 2: The successful attempts and success rate by the testing user set

| Sr. No. | Age Group | Total Persons in the Age Group | Total Attempts | Successful Attempts | Success Rate |
|---------|-----------|--------------------------------|----------------|---------------------|--------------|
| 1 | <20 | 5 | 102 | 100 | 98.03% |
| 2 | <30 | 13 | 280 | 267 | 95.35% |
| 3 | <40 | 6 | 115 | 111 | 96.52% |
| 4 | <42 | 1 | 16 | 14 | 87.50% |

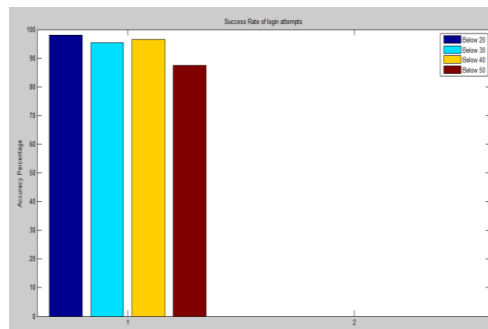


Figure 11: Bar Graph representing the login accuracy by the testing users of various age groups

CONCLUSIONS AND FUTURE SCOPE

The implemented work includes the new multi-level password scheme, which is designed to prevent the security risks of the currently popular graphical password schemes. The new scheme has been proposed for its use in the power user portal for data access on cloud server. This scheme is designed using the GUIDE (Graphical User Interface Development environment) tool of MATLAB. When a user enters the pattern passwords, a numerical code for the pattern password is generated on the basis of the grid point indexing numbers. For future work, further innovations on the existing solutions is needed to fortify the security requirements for containerized workloads and environments.

REFERENCES

- [1]. Arora, K., Kumar, K., & Sachdeva, M. (2011). Impact analysis of recent DDoS attacks. *International Journal on Computer Science and Engineering*, 3(2), 877-883.
- [2]. Braun, M., & Xie, G. G. (2002). A Feedback Mechanism for Mitigating Denial of Service Attacks against Differentiated Services Clients. *NAVAL POSTGRADUATE SCHOOL MONTEREY CA DEPT OF COMPUTER SCIENCE*.
- [3]. Carlson, M., Weiss, W., Blake, S., Wang, Z., Black, D., & Davies, E. (1998). An architecture for differentiated services. RFC 2475, December.
- [4]. Chang, R. K. (2002). Defending against flooding-based distributed denial-of-service attacks: a tutorial. *IEEE communications magazine*, 40(10), 42-51.
- [5]. Kargl, F., Maier, J., & Weber, M. (2001, April). Protecting web servers from distributed denial of service attacks. In *Proceedings of the 10th international conference on World Wide Web* (pp. 514-524). ACM.
- [6]. Khattab, S. M., Sangpachatanaruk, C., Melhem, R., Mosse, D., & Znati, T. (2003, August). Proactive server roaming for mitigating denial-of-service attacks. In *Information Technology: Research and Education, 2003. Proceedings. ITRE2003. International Conference on* (pp. 286-290). IEEE.
- [7]. KotagiriRamamohanarao, T. C. Protection from Distributed Denial of Service Attack Using History-based IP Filtering.
- [8]. Sardana, A., & Joshi, R. (2009). An auto-responsive honeypot architecture for dynamic resource allocation and QoS adaptation in DDoS attacked networks. *Computer Communications*, 32(12), 1384-1399.
- [9]. Tipton, H. F., & Krause, M. (2003). *Information security management handbook*. CRC Press.
- [10]. Yau, D. K., Lui, J., Liang, F., & Yam, Y. (2005). Defending against distributed denial-of-service attacks with max-min fair server-centric router throttles. *IEEE/ACM Transactions on Networking (TON)*, 13(1), 29-42.
- [11]. Yuan, J., & Mills, K. (2005). Monitoring the macroscopic effect of DDoS flooding attacks. *IEEE Transactions on Dependable and Secure Computing*, 2(4), 324-335.
- [12]. Zhao, W., Olshefski, D., & Schulzrinne, H. (2000). *Internet quality of service: An overview*. Columbia University, New York, New York, Technical Report CUCS-003-00.
- [13]. Jin, H., Deng, L., Wu, S., Shi, X., Chen, H., & Pan, X. (2014). MECOM: Live migration of virtual machines by adaptively compressing memory pages. *Future Generation Computer Systems*, 38, 23-35.
- [14]. Kratzke, N. (2014). A lightweight virtualization cluster reference architecture derived from open source paas platforms. *Open Journal of Mobile Computing and Cloud Computing*, 1(2), 17-30.
- [15]. Lee, B. Y., Park, J. H., & Yoo, J. S. (2012). Software Architecture of the Grid for implementing the Cloud Computing of the High Availability. *The Journal of the Korea Contents Association*, 12(2), 19-29.
- [16]. Ma, F., Liu, F., & Liu, Z. (2010, July). Live virtual machine migration based on improved pre-copy approach. In *Software Engineering and Service Sciences (ICSESS), 2010 IEEE International Conference on* (pp. 230-233). IEEE.
- [17]. Pickartz, S., Gad, R., Lankes, S., Nagel, L., Süß, T., Brinkmann, A., & Krempel, S. (2014, August). Migration techniques in HPC environments. In *European Conference on Parallel Processing*(pp. 486-497). Springer, Cham.
- [18]. Rad, P., Muppidi, M., Agaian, S. S., & Jamshidi, M. (2015, September). Secure image processing inside cloud file sharing environment using lightweight containers. In *Imaging Systems and Techniques (IST), 2015 IEEE International Conference on* (pp. 1-6). IEEE.
- [19]. Rey, J., Cogorno, M., Nesmachnow, S., & Steffanel, L. A. (2015, March). Efficient prototyping of fault tolerant Map-Reduce applications with Docker-Hadoop. In *Cloud Engineering (IC2E), 2015 IEEE International Conference on* (pp. 369-376). IEEE.
- [20]. Seo, K. T., Hwang, H. S., Moon, I. Y., Kwon, O. Y., & Kim, B. J. (2014). Performance comparison analysis of linux container and virtual machine for building cloud. *Advanced Science and Technology Letters*, 66, 105-111.
- [21]. Sun, M., & Ren, W. (2013). Improvement on Dynamic Migration Technology of Virtual Machine Based on Xen [J]. *International Form, IEEE*, 2(10), 124-127.
- [22]. Yoon, J., Park, C., & Song, U. S. (2013). Building the Educational Practice System based on Open Source Cloud Computing. *Journal of Digital Contents Society*, 14(4), 505-511.
- [23]. Zhang, R., Li, M., & Hildebrand, D. (2015, March). Finding the big data sweet spot: Towards automatically recommending configurations for hadoop clusters on docker containers. In *Cloud Engineering (IC2E), 2015 IEEE International Conference on* (pp. 365-368). IEEE.