SINT: A DHT based Secure P2P file sharing system

Subir Kumar Padhee
Department of Electrical and Computer Engineering
University of Colorado Boulder
Boulder, Colorado
Email: Subir.Padhee@colorado.edu

Sairam Udaya Janardhana Muttavarapu
Department of Electrical and Computer Engineering
University of Colorado Boulder
Boulder, Colorado
Email: Sairam.Muttavarapu@colorado.edu

Shrivathsa Keshava Murthy
Department of Electrical and Computer Engineering
University of Colorado Boulder
Boulder, Colorado
Email: Shrivathsa.Murthy@colorado.edu

***Abstract*—P2P Systems are widely used in the internet today. This project implements Peer to Peer protocol to build an application that involves file transfer using DHT. As the design involves multiple peers, there is a big security threat as malicious peers can access files and crash the network. We discuss security challenges involved in the process like Peer to Peer authentication, encryption, Distributed Denial of Service(DDoS) and possible attacks in distributed systems. Also, we implement solutions like asymmetric RSA encryption, DDoS defence system adapted from various papers with modification of our own to mitigate security challenges.**

I. Introduction

We have developed a Distributed Hash Tables based Peer to Peer file sharing application using the Kademlia based tomP2P framework, on the lines of popular file sharing application, Bit Torrent. Our application SINT, which stands for SINT Is Not Torrent, allows users to share files with and download from peers over a TCP based network shared by all the peers. We have adopted the indirect storage model of DHTs for this project wherein no file content is stored on the DHTs. Only the addresses of peers sharing the files are stored on the DHTs paired with the keys for the files shared. Apart from adding a level of security, this allows the load to be shared by multiple nodes. Every file is downloaded in chunks which are encrypted using AES symmetric encryption. The encryption key is further RSA encrypted. A Trust Factor parameter is used to rank peers based on their past behavior. Apart from acting as a level of security, the trust factor helps in load balancing the network. There is an authentication process every peer undergoes at the time of joining the network.. Also, we have implemented another packet sniffer module that captures DDoS attacks and notifies the application of a potential threat. The application takes positive action by blacklisting such malicious peers and mitigates DDoS attacks.

II. Design

A. Overview

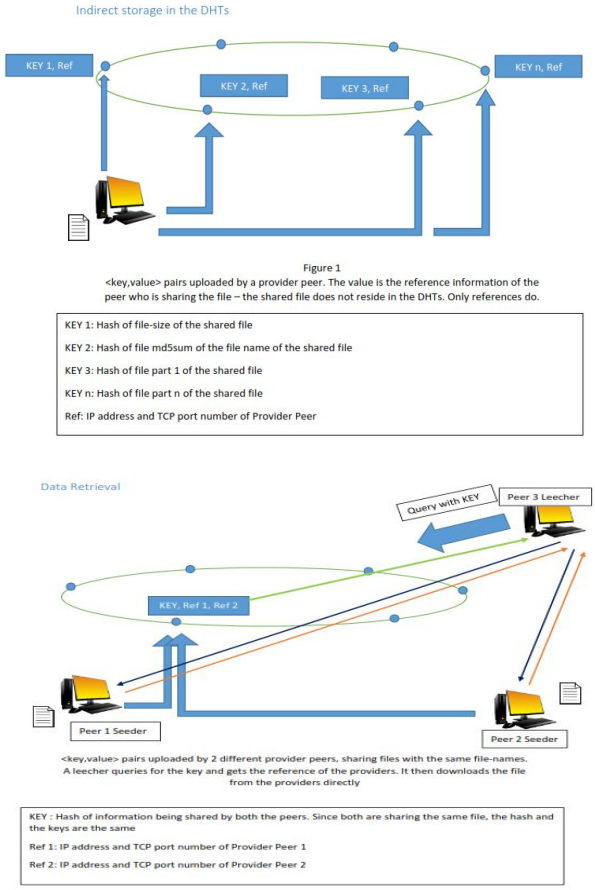
The components of the system can be divided into two main parts.
1. The Central server: A Central server is hosted on the cloud which serves as a bootstrap server and initializes the system. It hosts the central database that stores user credentials like email address as unique identity, password, name and IP address for identity. It is responsible for new user signup and public-private key generation for the user which is used for file encryption. The public key is stored in the database while the private key is emailed to the user. Periodic heartbeat messages are sent to all participating peers from the central server and some security features enabled from here, as will be elaborated in later sections. All the information storage in the central server is done on a MySQL Relational Database. A PHP based interface that supports HTTP requests is used for storing and querying values to and from the database by the participating peers.
2. The Participating peers: A participating peer is one that has a SINT client running on it. During initialization it connects to the central bootstrap server. For file sharing and downloading, a participating peer connects with the central server for getting the public RSA key for file encryption, querying and updating the Trust Factors of peers. The SINT client provides the user with an interactive user interface built using the Standard Widget Toolkit in Java.
Precisely, the central server has a role to play for system initialization and whenever its a security related function to be performed. We consider the central server highly resistant to security threats with Kademlia framework and strict IP-table rules.

B. P2P File Sharing

We have built our system, SINT on top of the Kademlia based tomP2P framework. Every peer that shares files (and hence is called a Provider), acts as a tracker for the files it shares in the system. The contents of the shared files are not stored in the DHTs. Only references to the sources of the shared files are stored in the DHTs. Therefore, whenever a Provider shares a file, it puts the hash of the file name on the DHT and pairs it with its own address which includes IP address, port numbers among other information. References of multiple Providers of the same file are paired with the same key. Whenever a peer seeking a file sends a query with the key of the file (hash of the filename), it is served a response with the addresses of all the Providers of the file.

The peer then iterates through all these provides retrieving chunks of the file from all of them. The peer uses the Trust factor of a Provider peer to decide whether to download and how many chunks to download from it. This model of file sharing is called the Indirect Storage in DHTs model.

There is an alternate model called the Direct Storage in DHTs where-in a peer willing to share a file uploads a key for the file and pairs it with a value which is the content of the file. This way, the shared file is stored on a node in the DHT whose Node ID is closest (distance metric computed by XOR operation) to the key of the shared file. Any peer seeking the file can thus download the file directly from the DHTs without having to contact any peer separately. We have used the former-indirect- model since it is more scalable and is less constrained by the memory in the participating peers.



C. Load Balancing

We have defined a metric called Trust factor (referred to as TF hence forth) that SINT uses to decide the proportion of file content to be downloaded from each peer. For a file download, a SINT client runs through the list of Provider peers (peers sharing the requested file) and queries the TF of each of them from the central database. Based on the TF of the peers, using threshold limits of 1, 3, and 7, SINT dynamically sorts the peers into three groups- best (TF 7- TF 10), good (TF 3- TF 7) and bad (TF 1- TF 3). Peers with TF lower than 1 are automatically dropped out and have to rejoin as new peers after going through the authentication procedure required for a new peer. The size of each chunk and hence the number of chunks that a file has to be divided into in order to be uploaded or downloaded is also computed dynamically based on the size of the file. For files smaller than 1 MB, a chunk size of 1 KB is chosen. Files larger than 1 MB but smaller than 100 MB are transferred in chunks of 1 MB. For files larger than 100 MB, SINT uses chunks of 50 MB in size. This dynamic allocation of chunk size ensures that the file download is well distributed. Further, only 60% of the files chunks are downloaded from the best group of peers, while 30% and 10% are downloaded from the good and bad groups respectively. If a particular group has no members, its quota of peers is taken up by the group immediately higher ranked than the group in question. This goes on cumulatively. This allocation ensures that the best group of peers are not overburdened by network traffic and the good and bad peers keep getting chances to improve their Trust Factors and keep using SINT. If a peer falls below the TF threshold of 1, it is dropped and can neither share nor download files.

D. Trust Factor

While the Trust Factor of participating peers plays the central role in balancing the load during a download, it serves as the reputation of the peers as well. A Peer has to maintain high TF in order to ensure that it does not fall behind and is eventually dropped out of the system when its TF falls below 1. This acts as a deterrent for malicious peers and a potential sieve to filter them out over time. Trust Factor for a peer is measured and stated on a scale of 0 to 10, 0 being the most undesirable value and 10 the most sought after. TF is combination of two components, the first which has three sub components - a, b and c, is automatically computed by the SINT client and carries a 90% weight in the TF. The second component is user feedback which is sought from the user at the end of a download and counts for 10 The first sub-component a concerns the speed of download from the concerned peer with the size of file being downloaded factored in. As the file chunks are being downloaded from a peer, SINT computes the speed of download for each chunk downloaded. These speed values are multiplied by the size of the entire file downloaded and averaged over the number of transaction (which is same as the number of chunks downloaded from the peer for the concerned file). The averaged value is normalized on a scale of 10 by factoring in a precomputed Best-Case value which we derived empirically. This component carries a weight of 30 The next sub-component b concerns data integrity and counts for 50% weight of the 90%. Clearly, data integrity is of prime importance in a file sharing system. Files with spurious content are of no use to the downloader and their download eats into network traffic unnecessarily. Hence, a data integrity check failure gets the concerned peer a zero score for the component, while a success gets it a full score. The SINT client stores the data integrity check status of every peer the concerned file is downloaded from and is

therefore able to find out which peer was responsible for the compromise in case of a failure. The md5sum hash of the file contents is used for this data integrity check. However, a reference value is needed for every file downloaded to run the check against. SINT trusts the group of best peers (peers with TF higher than 7) to provide the reference md5sum hash value. All the peers in the best peers group are queried for the md5sum hash of the concerned file and only if all the values returned are equal, is the reference value stored and the process continued. If the values returned by all the best peers do not match each other, the downloaded file is deleted and the user is informed.

The third sub-component c is the existing TF value of the concerned peer. It is queried from the central database and carries a weight of 20% of the 90% part. This component ensures that the TF computation is cumulative and the history of a peers behavior has a bearing on its future.

The other component in TF computation is the user feedback and counts for 10% of the overall Trust Factor. It is on a scale of 0-10. If the data integrity check for the downloaded file fails, the peers who caused it to fail are given a zero score, while the other peers are given the score that the user gave as feedback.

TF =[(ax0.3)+ (bx0.5)+ (cx0.2)] x 0.9+ feedback x 0.1

E. User Authentication and File Encryption

A new peer joins the system by signing up through the SINT client. The central server generates a RSA public-private key pair for every new user/peer that signs up. The private key is immediately sent to the concerned user through the email address provided and is not stored anywhere. The SINT client takes the private key from the user locally before download starts if it doesnt have it already. The public key, on the other hand is stored on the mySQL database of the central server, for any participating (Provider) peer to access using the email address/IP address of a downloader peer as the key. All Provider peers encrypt the chunks of files they provide to the downloader peer using a randomly generated AES symmetric key. This AES key is encrypted with the RSA public key specific to the downloader peer and the encrypted AES key is also sent over to the peer downloading the file which then decrypts it using the private key it had received during sign up. It then decrypts the file chunk contents using the decrypted AES key.

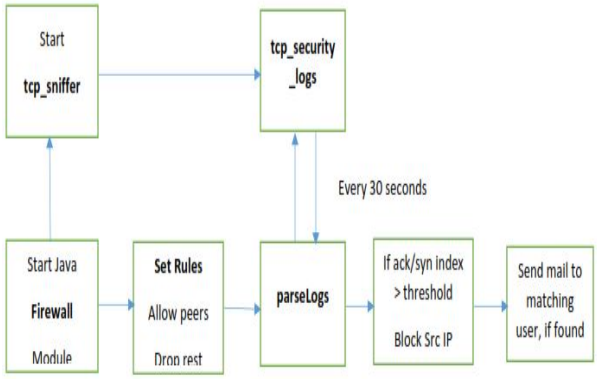
F. DDoS

The Kademlia nodes use TCP port 4002 which is secured by the library. Our attempt to overflow the network failed when we targeted this port used by the library. But, our P2P application also uses TCP port 4003, to send and receive files, which is vulnerable to potential DDoS attacks. The following section discusses our implementation to mitigate such attacks.

DDoS is one of the deadliest attacks in the internet today. An ideal solution is to mitigate the attack at the source. This would actually reduce large volumes of traffic at the destination. But this requires access to infrastructure at the ISP level. So, we do our best to mitigate at the application. We arrived at two solutions.

1. Analyze SYN/ACK flood traffic at the operating system level. We could add code in the ip_input.c file of the linux kernel. At the ip_recv function, main IP receive routine, we could add a module to drop packets after detecting DDoS attack.
2. Write an application to sniff incoming packets and log data. This will be read by our application to block malicious traffic.

We decided to implement the latter as earlier solution involves a customized operating system and deploying such operating system at every node is not an efficient solution.



As the application starts at the peer nodes, the Firewall class is called which will monitor traffic for potential DDoS attack. At this point, we know the IPs of all other nodes including boot peer. The firewall service allows tcp traffic at port 4003 from only peer nodes and drops packets from any other source. This is set by the following iptables system call.

iptables -A INPUT -p tcp –dport 4003 -s peer_ip -j ACCEPT
iptables -A INPUT -p tcp –dport 4003 -j DROP

This ensures traffic destined to port 4003 is from one of the other nodes and eliminates the possibility of a DDoS attack by an outsider.

TCP sniffer

This C daemon program is called from the java application in the very beginning. The program sniffs incoming packets and looks for DDoS patterns.A recvfrom function with a TCP socket not bound to any port or IP when put in an infinite while loop, gets all incoming tcp packets. This logic is used to sniff packets.

SYN flood An attacker sends a SYN packet but never sends ACK for the SYN ACK received. Instead, an RST is sent by the attacker. We maintain an index called single SYN flood index which counts the number of such incidents. This is the case the attacker uses his own IP throughout the period which is unlikely most of the times. In case of random source IP spoofing, we maintain another index called random

SYN flood index. Here, attacker just sends SYN by spoofing random source IPs. In both the cases, an incident with an index increasing exponentially high in short period, can be concluded as a potential attack.

ACK flood As explained above, another index is maintained for ACK flooding. Here, we look for continuous ACK packets which causes ACK based DDoS attack.

UDP flood Usually UDP flood attack will have either source or destination port as zero. UDP socket and another index are used to report such attacks. Even though the motive is not to attack a particular application, it floods the node with redundant packets and the processing capability is gradually decreased.

ICMP echo flood Monitors consecutive type 0 and 8 ICMP ping requests using ICMP sockets. The magnitude of such packets will be in the order of tens of thousands in case of an attack.

All these information is logged in a file. After setting the initial rules, the Java application constantly looks for these logs in 30 second interval and blacklists the source IP in case one of the indices goes high beyond threshold. Since we have already set the rules initially, such incident proves that the attacker is one of the peers as we have allowed only peers at this point. But, there is a good possibility that an outsider can spoof his source IP to that of a peer node and this might cause a potential blacklist of a genuine peer. As a solution to this problem, we query the database to see if there exists a matching user for the source IP. If found, we send an e-mail to the peer to reactivate. If this occurrence happens frequently, i.e. a malicious peer is trying to attack another peer, the malicious peer gets bad rating and will be permanently removed from the network.

The firewall service is restarted every time the application starts. Also, every time the log crosses a threshold, a warning message is displayed to the user in the user interface. This notifies the user of any potential attack and external measures can also be taken by the user to mitigate such attacks. We discuss setting indices threshold in the evaluation section.

III. Evaluation

P2P File sharing with Trust Factor As discussed earlier, SINT clients configure download and upload parameters dynamically so as to achieve an optimized sharing of files. These parameters include, chunk size and number based on size of file to be downloaded, grouping of peers into ranks based on trust factor and number of chunks to be download from each group based on size of the group. Further, the user feedback is also added to the trust factor depending on whether data integrity check succeeded or failed.

We tested with files of different sizes and also with some peers with wrong/less content under the same file names to simulate varying conditions. We did the same with different file types. However, all our peers had IP addresses in the same subnetwork, so we could not simulate geographically distant peers. The results were pretty impressive. We observed high download speeds in the range of 30-60 MB/sec for large files of size in the 1-2.5 GB range. The downloads were slower for files smaller than 1 MB since the number of chunks was very high in such cases. The download speed increased considerably when the number of peers sharing the same file increased. We observed the trust factor updating as expected. The peer with the altered file had its trust factor reduce pretty rapidly from 10 to under 7 and then below 1 after which it was no longer part of the system. This reduction in trust factor only happened for the peer with the altered file and not for the ones with the correct file. The effect of user feedback was likewise.

User Authentication and Encryption The system of user authentication and file encryption provides multiple layers of security.

1. Private key is transferred directly to the user via e-mail and not on the SINT network We trust the email clients security features to guard against possible email hacking.
2. Each chunk is encrypted using a randomly generated AES key and therefore, a malicious software will have to intercept every chunks encrypted AES key and crack the RSA key to be able to recompose the entire file.

DDoS Simulating an attack to analyze DDoS traffic pattern was challenging. Initially, we wrote a web server and hosted on our home network by exploiting port forwarding feature of the router. The website was accessible from another geographical location before the attack. We used hping3 utility to carry out SYN, ACK, UDP and ICMP based DDoS attacks. The website was totally inaccessible from another location after a minute the attack was launched. We noted down the ACK, SYN, UDP and ICMP indices to calibrate the threshold. The SYN/ACK index was less than 50 under normal scenario and it took about 5 seconds to load the homepage. During the attack period, index was increasing exponentially. It reached 100000 in less than 30 seconds and the system started to slow down.

Then, the same experiment was repeated on one of peer nodes and we were unable to download any file from the infected node due to the intensity of the attack on TCP port 4003. Since some routers and Amazon AWS have DDoS defense mechanism installed by default, attacking the node from another location was difficult. Hence we attacked from another node inside the same network.

The following causes SYN flooding (-S) at 172.31.43.144 and port 4003 by spoofing random source IPs.

Hping3 S d 120 w 64 –flood p 4003 172.31.43.144 rand-source

Similarly A for ACK flooding, –udp, –icmp options can be used to carry out other DDoS attacks. SYN Flood index threshold is set to 300 and ACK index flood threshold is set to 4000. ACK is higher because consecutive ACK responses will be higher even under normal scenario which is acceptable.

The solution had a positive impact when we deployed on our application. The packets were getting dropped when we tried

to attack the nodes from outside network as it allows traffic only from peer nodes. Also, when we tried to attack from one of the nodes in the network, tcp sniffer was able to recognize the source attacker and dynamically add it to blacklist. We had less than 1% SYN Flood false positives. Even though there was enough traffic in the pipe, packets were getting dropped before reaching the application and thus application resources were saved which establishes our DDoS defense goals.

IV. Related Work

Peer Authentication All the papers we surveyed in the field of security in Distributed Systems unanimously agree on the need for a peer authentication mechanism in any system that is based on a P2P network. The peer authentication mechanisms proposed by all of the papers is built on a feedback scheme where peers rate the authenticity/trustworthiness/reputation of peers they have transacted with. Whenever a peer needs to interact with another peer, it checks the peers reputation or trustworthiness and decides whether to proceed or not.

The paper Reputation-based trust model for peer to peer ecommerce communities [3], talks about quantifying and comparing trustworthiness of peers based on transaction-based feedback system in e-Commerce communities like eBay auction system and the like. The authors propose five different parameters to consider while computing the trustworthiness of a peer. The first is the amount of satisfaction peers receive while transacting/dealing with the peer in question. While this factor sounds pretty subjective, the remaining factors aim to streamline it more. The second factor is the number of transactions, which is used to average out the trust value over the number of transactions. By bringing this in, it can be ensured that a peer may not increase its trust value by increasing its transaction volume and use it to hide the fact that it frequently misbehaved on a number of occasions. The third parameter is credibility of the feedback, which aims to weigh the feedback from more credible peers more than those from the lesser credible ones. While credibility can be a complex parameter to measure, the paper suggests using the current trustworthiness of a peer or a function of it as a measure of credibility as well. The last two parameters weigh the context of transaction and the community to evaluate the trust factor. As an example of importance of context of transaction, a malicious peer can score good on small transactions only to play foul in a bigger transaction. Likewise, in a network representing a community for sharing music, a recent history should have more weightage so as to reflect the current trend. The authors then present a formula to compute the trustworthiness factor by combining the mentioned factors with the desired weight. The paper also talks about providing incentives to peers who provide feedbacks, in the form of increase in trust factor (by adding this as the 6th parameter in the formula).

The paper Eigen-Trust Algorithm for Reputation management in P2P networks [2] also talks about computing a global trust value for each peer by combining individual feedbacks from transactions like the previous paper discussed. It also suggests allowing peers to add an individual component to the global value of a peer based on individual experiences with that peer. This would allow peers to bias downloads based on own experiences. Additionally, it stresses the need for a P2P based system to be truly decentralized when it comes to policing and management in the group and the need to keep the identities of the peers anonymous and opaque to the trust factors. It mentions that while computing the trust factor, the newcomers should not be blindly trusted and given any advantage because that can act as an easy way in for malicious peers. It further adds that the computation of the trust value should not cause an overhead for the network in terms of computation, infrastructure, storage and message complexity. The paper drives in the point that the trust value of peer should not be stored on the peer it concerns as this allows malicious peers the opportunity to alter their values. The paper proposes a novel strategy to choose peers not directly based on the trust factor (which would cause overloading of the most trustworthy peers), but probabilistically based on the trust values so that the load is distributed over peers.

The paper Managing trust in a peer-2-peer information system [1], talks about the importance of managing the trust and the trust factor values in a decentralized fashion instead of a central server, because only that makes the distributed system truly scalable. It advocates use of a P-Grid[5] to maintain the trust data in a distributed manner.

The paper Trust-Me: Anonymous Management of Trust Relationships in Decentralized P2P Systems [4], talks about the importance of securing the trust information of all the peers and that of maintaining anonymity of the trust host and the trust querying peer. It proposes the use of asymmetric cryptography schemes to store and access the trust values of the peers.

Distributed Denial of Service Security plays a key role in todays world of internet and cloud computing. Cloud computing is one of the technologies where a large amount of data is stored online with distributed services to offer. Because they are distributed in nature, it has become an easy target for hackers to attack peer nodes and exploit information. Denial of Service (DoS) is the largest threat to the internet and internet of things. This a technique that involves in flooding a server or node by redundant traffic beyond its capabilities which eventually forces the node to shut down and thus denying service for legitimate users. Distributed denial of service (DDoS) is a similar technique that involves multiple sources and are initiated by master and a network of remotely controlled, well-structured and widely dispersed nodes called Zombies.

Different types of DDoS attacks are listed as follows.

1. Bandwidth depletion Attacks This consumes the bandwidth of the victim by flooding unwanted traffic.

a. Flood attacks These are instigated by ICMP and UDP packets. An attacker sends a large number of UDP packets

to victims random or specified port. When the victim finds out they do not belong to an application, an ICMP destination unreachable message is sent back. Now the return address is spoofed and it does not go to the zombie. This way available bandwidth is depleted. Similarly, ICMP_ECHO_REPLY packets are used to initiate ICMP flooding.

b. Amplification attacks Attacker sends large number of packets to a broadcast IP range. As a result, nodes in the broadcast range reply to the spoofed victim address thereby resulting in malicious traffic.

2. Resource depletion attacks This is targeted to strap victims system resources, so that the legitimate users are not serviced. Different types of such attacks are listed below. a. Protocol exploit attacks This uses a specific protocol to flood the traffic. TCP SYN attack is one such example. TCP SYN is not acknowledged and this makes the victim to flood the network by repeatedly sending SYN ACK packets.

b. Malformed packet attacks Attacker crashes the victim by sending packets wrapped with malicious information. This can be achieved by altering optional field in the IP packet.

Countermeasures and Mitigating policies against DDoS attacks

1. DDoS intrusion prevention Ingress and Egress filtering, route based distributed packet, history based IP filtering, Changing IP address, Load balancing and honey pots

2. Anomaly detection This method detects DDoS attacks by recognizing the anomalies in the systems performance. Normal behavior of the system is recorded previously. This is then compared with current system performance to make a decision.

3. Misuse detection - This method detects DDoS attacks by maintaining a database of well-known signatures and patterns. If one such event is noticed in future, it is reported.

4. Response to intruder detection Once the DDoS attack has been detected, source address and history of the attacker is maintained to take future preventive measures.

5. IP and ICMP trace back Path traversed by the attacker is reverse traced to find the attacker.

Some of the DDoS attacks can be detected but not prevented. In such cases, the impact caused by such attacks can my minimized by increasing the fault tolerance and quality of service of the nodes. DDoS attacks in the cloud environment contribute to 14%. The paper discusses such attacks and prevention techniques. Some of them are listed below.

Name of the attack Cause/ Definition Prevention technique

1. Virtual Machine attack Vulnerabilities in the hypervisor that runs VM - Advanced cloud protection system

2. DNS Attack redirect victim to a different site while resolving hostname DNS Security extensions

3. Port scanning available open ports Port encryption, Firewalls

4. SQL Injection malicious code injected in SQL query parameterized queries, user input validation

5. CAPTCHA breaking audio system to beak captcha letter overlap, variable fonts, lengthy strings

6. Dictionary attack Brute Force attack Encrypted words for passwords, challenge response

7. Google hacking track sensitive information avoid custom authorization, backup policies

Encryption and Data Integrity Symmetric and Asymmetric encryption [7] are two kinds of encryption techniques that secure file transfer protocols generally employ.

In symmetric encryption technique, both the sender and receiver should have the copy of the same key for encryption and decryption of the file. Some of the symmetric encryption algorithms include: AES (Advanced Encryption Standard), Blowfish, DES (Data Encryption Standard), Triple DES etc.

In Asymmetric encryption technique, sender and receiver have two keys (public and private key) for encryption and decryption of the file. Public key is used for encryption of the file and the private key is used for decryption of the file. Most widely used asymmetric key algorithms are RSA, DSA and Diffie-Hellman Key Exchange.

There is a tradeoff between both the kinds of algorithms in terms of computational capabilities and ease of key distribution. Symmetric encryption takes less computation time when compared with asymmetric encryption because of the large key numbers involved in computation. It is easy to crack a symmetric encryption within less time when compared with asymmetric encryption. As asymmetric encryption involves more time in computation, there is high possibility of denial of service (DoS) attacks. Because of the involvement of private key, the peers can easily share their public keys on the network for secure file transfer. PAuth [8] is an authentication protocol involving the central server in which users identity is verified by its peers designed against both DoS attacks based on the high computational cost of password verification using hash function. In this protocol, the server will authenticate a user based on analysis of verification responses from randomly selected peers using asymmetric encryption. Because of the set of verifying peers is chosen randomly by the server for each authentication attempt, it is thus unlikely that any one peer would be overloaded with asymmetric encryption computation. There is a tradeoff of computational cost on any one peer with the communication overload of server.

V. Conclusion

While many P2P file sharing systems do exist today, security is a major concern in most of such systems. Our aim in this project was to first develop a stable file sharing application and then add levels of security to it. We have been able to address some of the security concerns and have taken steps toward addressing some more. The Data Integrity check in the form of md5sum hash check, data security through double encryption - Asymmetric encryption of the symmetric key and the peer authentication system through the use of a communication medium (the email) exterior to the network used by the system are formidable layers of security. The Trust Factor which is partly automatic and uses user feedback for the other part is another layer of security which works as a sieve to filter out malicious peers in quick time. Distributed Denial of Service is another security threat we have tried to tackle. Although it may not be possible to eliminate this threat, our application

serves to mitigate the risks to a certain extent. The application takes a very conservative approach in populating the IP-tables of the Central server as well as those of the participating peers. IP-port numbers of only known peers (user authenticated) are allowed to connect. The Packet sniffer firewall service module captures DDoS attacks with less than 1% false positives and the application takes necessary action to blacklist malicious nodes which helps in mitigating DDoS attacks.

ACKNOWLEDGMENT

We wish to thank authors of various papers we referred to. Also, we would like to thank Dr. Shivakant Mishra for his guidance throughout the project.

REFERENCES

[1] K. Aberer and Z. Despotovic. *Managing trust in a peer-2-peer information system*. CIKM, 2001.

[2] S. Kamvar, M. Schlosser, and H. Garcia-Molina. *EigenTrust Algorithm for Reputation management in P2P networks*. Twelfth International World Wide Web Conference, 2003.

[3] L. Xiong and L. Liu. *A reputation-based trust model for peer to peer ecommerce communities*. Third International Conference on Peer-to-Peer Computing, 2003.

[4] A. Singh and L. Liu. *TrustMe: Anonymous Management of Trust Relationships in Decentralized P2P Systems*. Ninth International Conference on Cooperative Information Systems (CoopIS 2001), 2001

[5] K. Aberer. *P-Grid: A self-organizing access structure for PdP information systems Proc*. Ninth International Conference on Cooperative Information Systems (CoopIS 2001), 2001.

[6] B Prabadevi and N Jeyanthi. *Distributed Denial of service Attacks and its effects on Cloud Environment- a Survey.* Networks, Computers and Communications, The 2014 International Symposium, 2014.

[7] John Carl Villanueva. *Symmetric vs Asymmetric Encryption*, Internet: http://www.jscape.com/blog/bid/84422/Symmetric-vs-Asymmetric-Encryption, Mar 15, 2015.

[8] Zijing Gao, Thomas Lu and Anand Srinivasan. *PAuth: A Peer-to-peer Authentication Protocol.* March 20, 2015..

[9] *Asymmetric Encryption Algorithms, Diffie-Hellman, RSA, ECC, El-Gamal, DSA*. Internet: http://www.omnisecu.com/security/public-key-infrastructure/asymmetric-encryption-algorithms.php, 2016.

[10] Joel Dubin. *Choosing the right public key algorithm: RSA vs. Diffie-Hellman.* Internet: http://searchsecurity.techtarget.com/answer/Choosing-the-right-public-key-algorithm-RSA-vs-Diffie-Hellman, April 2007.

[11] George Oikonomou, Jelena Mirkovic and Peter Reiher. *A Framework for A Collaborative DDoS Defense.* In Computer Security Applications Conference, 2006.

[12] Silver Moon. *SYN Flood DOS Attack with C Source Code (Linux).* Internet: http://www.binarytides.com/syn-flood-dos-attack/.

[13] *RSA Public Key Encryption System.* Internet: https://globlib4u.wordpress.com/2013/10/16/rsa-public-key-encryption-system/.