RSYNC over HTTPS for Linux and Windows with Seamless data transfer

Ajay Tanpure, Akshay Patil, Anup Bansod, Aniket Kulkarni

Abstract- Transfer of private, sensitive or confidential data over WAN requires protection against eavesdropping, unauthorized access, use, modification, perusal, disruption assuredly and efficiently. The available open-source utilities like Rsync reduces time and bandwidth required to update the file across the network. It uses the interactive protocols that detects the changes in a file and sends only changed data but it uses weak SSH encryption for secured transfer even over WAN. It is not a secure mode of the data transfer as there are well known attacks like brute-force attack, dictionary attack on SSH. Every Organization requires their data to be transferred securely and they rely heavily on tools which are secure and can be used efficiently without much constraints. As HTTPS is being used heavily and tested thoroughly, it will provide the required high level of security. Another aspect of solution is solving the problem that industry is stuck with these days which is seamless data transfer securely. What we have proposed and implemented is a solution that will transfer data over highly secure HTTPS between linux and windows seamlessly, using the existing delta transfer algorithms for detecting the changes in file. It will not implicate or impose any more requirements/dependencies in comparison with SSH. Our solution will not require any software installed or additional permissions to be enabled for data transfer except the port to be open. Linux host initiating the syncing process will push the binaries on windows host, which will run as a service and can communicate with another processes on the linux host. Moreover, our final tool can be used seamlessly between linux-windows for data backup, recovery, synchronization or plain file copying. We believe our solution is of great importance in the scenarios like OS-images synchronization in cloud infrastructure products, server mirroring in addition to its uses for general purpose file transfer securely.

Index Terms- Rsync, Delta Transfer algorithm, HTTPS, SSH

I. Introduction

Rsync is arguably one of the most powerful tools you can have in your arsenal as a systems administrator or user. It does not matter if you take care of one system or thousands, rsync can make your life easier. Rysnc makes the efficient file synchronization a reality. It enables administrators to propogate changes to files or directory trees. To save the bandwidth and time, rsync moves the minimum amount of the data by identifying common region between the source and targer file. When synchronizing files, rsync sends only the portion of the file that have changed and copies unchanged data from the previous version already on the target. Faster and more efficient methods for syncronizing copies make it easier to manage manage destributed replicas. Rsync uses the rsync algorithm which provides th emethod for briging remote files into sync.

An important feature of rsync not found in most of the similar programs/protocols is that the mirroring takes place with only one transmission in each direction. Rsync can update the whole directory trees and filesystem, optionally preserves the symbolic links, hard links, file ownership, permissions, date and time. Rsync requires no privileges to install and internal pipelining reduces latency for multiple files. Rsync algorithm consists of basics steps:

Signature generation: A signature block describing an existing file is generated by breaking a file up into equally sized pieces and generating to checksums, weak and strong for each block. Weak checksum is calculated using the Adlers-32 algorithm and strong checksum is calculated using Message Digest-4 hashing algorithm. These checksums are concatenated to form the signature block.

Signature Search: The difference between the data that the signature block was generated on and the new data are found by one-pass search on the new data, computing a rolling checksum and hash table to find blocks of data at any alignment in the new data that match any block in the old data. The differences are encoded in terms of literal data and block references.

Reconstruction: The computed differences can be applied to the old data to generate the new data by writing literal data and blocks from the old data to new data. Rsync can be used to upload and download the files. While uploading the files or directory, remote side sends the checksum of files to client side. Client side in turn, by comparing the checksum sent by remote side with checksum generated at local side, sends only difference to remote side. In downloading mechanism, client side sends the checksum of files which it has to server side. Server by comparing the checksum sent by client with checksum of files which it has, generate the difference and send to client which in turn reconstructs the updated files. Rsync can be used for downloading (Fig-1) and uploading (Fig-2) files from/to remote host

II. BACKGROUND

Rsync usually works in two modes as daemon mode and non daemon mode. In the daemon mode, rsync daemon on server side continuously listens to the TCP port 873 for incoming connections. Syntax of rsync command in daemon mode is

```
rsync [OPTION...] [USER@]HOST::SRC... [DEST]
rsync [OPTION...] SRC... [USER@]HOST::DEST.
```

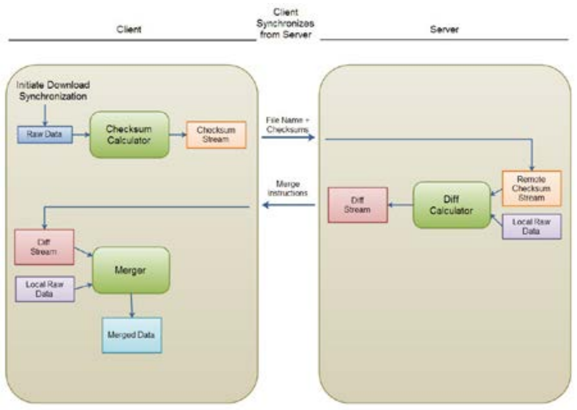

Fig. 1: Download process

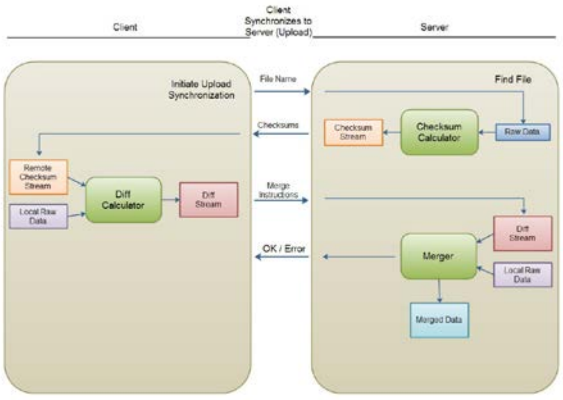

Fig. 2: Upload process

In the non-daemon mode, rsync connects to remote side side using remote shell SSH or RSH.Client side rsync starts the rsync process on the server side using SSH.Syntax of rsync command in non- daemon mode is

rsync [OPTION...] [USER@]HOST:SRC... [DEST] rsync [OPTION...] SRC... [USER@]HOST:DEST

In non-daemon mode is widely used as it provides the en- cryption of data. Despite of these features of the rsync, its shortcomings regarding the data security preclude its use. We address one specific shortcoming. The encryption of the SSH is too weak to use over the network as there are many well known attacks like bruteforce attack, dictionary attack. To overcome this security threat of the existing rsync we are trying to make data transfer over HTTPS. HTTPS is widely used these days and it tested thoroughly. It provides high level of security as it uses SSL/TLS protocol and provide authority certificates. HTTPS has not much dependency on client side or server side and works on all platforms. Also there are no tools which can be used with both unix-like and windows-like operating system seamlessly with no dependency on target side.

III. DESIGN

Design revolves around the original rsync algorithm for synchronizing the files at remote location. HTTPS is better option to overcome the security issue in data transfer over

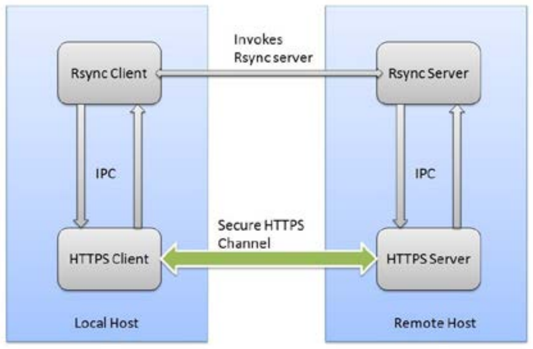

Fig. 3: Linux-Linux

the network. Our design can be used with linux and windows provided the process of syncing initiated by the linux side only.

Linux-Linux: In this case, both the local and remote sides are linux. In existing rsync, using remote shell, rsync process is invoked on the remote side, and data transfer takes place over SSH or RSH. In our design, rsync process is started on the remote side using SSH as existing rsync does. Rsync contains embedded HTTPS server which runs as the thread when invoked. When we are uploading the files to remote side, new HTTPS server process is started inside rsync process on remote side. Once HTTPS server process is started on the remote side, handshaking is done between the HTTPS server and local side. HTTPS server should have the authority certificates. Remote side sends the checksum of files to local side. By comparing the checksum sent by the remote side with checksum of file present at local side, local rsync generates the difference and send it to remote side. All these communication takes place over HTTPS instead of SSH. Main thread of rsync process has functionality to reconstruct the files from the old version and the differences sent over HTTPS. In the case of downloading the files from the remote side, HTTPS server will be started on local side instead of remote and invoked rsync on remote side. Once handshaking is done between the HTTPS server at local side and the remote rsync process, actual data starts. Local side sends the checksum of the files to the remote side. Remote side compares the checksum sent by local side with checksum of the files which it has and generates the differences. These differences are sent to local side using which; local side rsync generates the new files from old versions and difference. All the communication takes place over HTTPS. Once data transfer is over, processes rsync and HTTPS server are terminated and connection between the local side and remote side is closed. Refer Fig.-3.

Linux-Windows: In this case, local side is linux and remote side is windows. In our design, there is no prerequisite of any software installed on windows so it can be used seamlessly. User will provide w option in the rsync command mentioning that remote side is the windows.

Linux side uploads the binaries of rsync on the windows side. These binaries runs as service named rsyncd. Service rsyncd contains logic for rsync same as running on the linux side and HTTPS server. Samba API are used for uploading binaries on the windows side run as service.

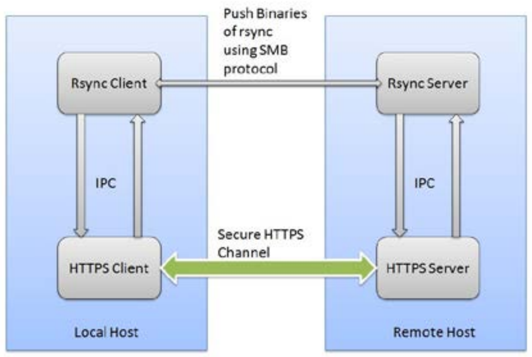

Fig. 4: Linux Windows

Admin share is needed to be enabled on the windows side and user control access level should be set to lowest level. While uploading the file, after the handshaking is done between the linux side rsync and HTTPS server on windows which is running as service, rsync on linux sends the HTTPS request for the checksum of the files. Linux side rsync compares the checksum with checksum of files present at local side and generate the difference. These differences are sent to windows side HTTPS server which in turns passes it to the rsync which is also running as service and reconstruct the undated file. When downloading files, HTTPS server is started on the linux side and it sends the checksum of files to windows. Rsync on windows compares checksum sent by linux with checksum of files which are present at local side and send the differences to HTTPS server on the linux. Linux side Rsync generates the updated files from old version and differences. Once the syncing process is complete, rsync on linux side stops and uninstalls the rsyncd service running on the windows. Refer Fig.-4.

IV. SYNCHRONIZATION

HTTP protocol works on the request-response mechanism. To get some data from the HTTPS server, HTTPS client need to send the request to server, server in response send the actual requested data back to client. But rsync protocol sends the complete bolck of checksum or differences of checksum. So, we have added a new control character say,Kc , in the rsync protocol. When data is not ready on the pipe from the rsync client to HTTPS client or from rsync server to the HTTPS server, HTTPS client or HTTPS server will send control character Kc , so other ends process i.e HTTPS server or HTTPS client can negect that and check for the pipes from the Rsync processes. Because of it, we make rsync protocol work like request-response of HTTP.

V. FUTURE WORK

Currently rsync algorithm uses the Adlet-32 algorithm for calculating rolling checksum and MD5 algorithm for calculating the hash value of block of the data. MD5 algorithm is not the collision resistance. MD5 algorithm generates the 128 bits of the hash value for given data block. There is possibility that two blocks of data can have the same hash value. The probability of having same hash value is very less equal to 1-128. Thus other hashing algorithm can be used in rsync algorithm. As per our design, updated rsync will work for the linux and windows operating system, this constraint can extended to many other platforms like Solaris, macOS etc as updated rsync will be used in WAN and many hosts can different operating system. For the synchronization purpose we have added the control data Kc , there is the probability that, actual rsync data is same as the control data. Though there is very less probability of being same data 216 , It can make rsync protocol behave abnormally. This can be resolved by adding some hash function to generate control data.

VI. CONCLUSION

Updated rsync will work over HTTPS which ensures the security of the data being transferred. As windows operating system is widely used, and there is no need of any software installed on the windows host, it will prove efficient to use in WAN. Windows user will not need anything installed on the windows system except the admin share enabled. As the only the differences between the files will be sent as per the rsync algorithm, bandwidth will saved.

REFERENCES

[1] A. Tridgell Efficient Algorithms for Sorting and Synchronization ,Australian National University 1999.

[2] Zahed, K.S.; Rani, P.S.; Saradhi, U.V.; Potluri, A. Reducing storage

requirements of snapshot backups based on rsync utility, Communication Systems and Networks and Workshops,COMSNETS 2009

[3] Ghobadi, A.; Mahdizadeh, E.H.; Yong Lee Kee Pre-processing directory structure for improved RSYNC transfer performance. Advanced Communication Technology (ICACT), 2011

[4] Hao Yan; Irmak, U.; Suel, T Algorithms for Low-Latency Remote File Synchronization, INFOCOM 2008. The 27th Conference on Computer Communications

[5] Irmak, U.; Mihaylov, S.; Suel Improved single-round protocols for remote file synchronization FOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies

[6] David Rasch and Randal Burns In-Place Rsync: File Synchronization for Mobile and Wireless Devices Department of Computer Science, Johns Hopkins University

[7] P. Vo, and W. F. Tichy An empirical study of delta algorithms 6th Workshop on Software Configurat ion Management, March 1996

AUTHORS

First Author – Ajay Tanpure, ajaytanpure92@gmail.com
Second Author – Akshay Patil, akshay29patil@gmail.com
Third Author – Anup Bansod, anup.bansod@gmail.com
Fourth Author – Aniket Kulkarni,