CloudTurbine Security White Paper

CloudTurbine provides unique opportunities for secure data streaming between computers over local and wide area networks. CloudTurbine leverages data security provided by file storage and networking layers with which it interoperates. CloudTurbine is not a server nor an application per se, but a collection of modules that utilize CloudTurbine technology for storing and/or reading streaming data. Thus, CloudTurbine data security is best thought of as a system-integration issue.

By design, CloudTurbine writes and reads data only on the same local computer system upon which it operates. In this sense, the first layer of CloudTurbine data security is the same as local file system security. CloudTurbine enables streaming data via simple, regular files that in turn may be shared over networks. As such, the second layer of CloudTurbine data security is the same as that provided by your choice of file-sharing system network security.

Figure 1 illustrates the multi-layered security options to be considered for a CloudTurbine deployment. Note that as CloudTurbine utilizes files and file sharing as the medium and mechanism for data transport, it also utilizes and benefits from the associated file-related security infrastructure.

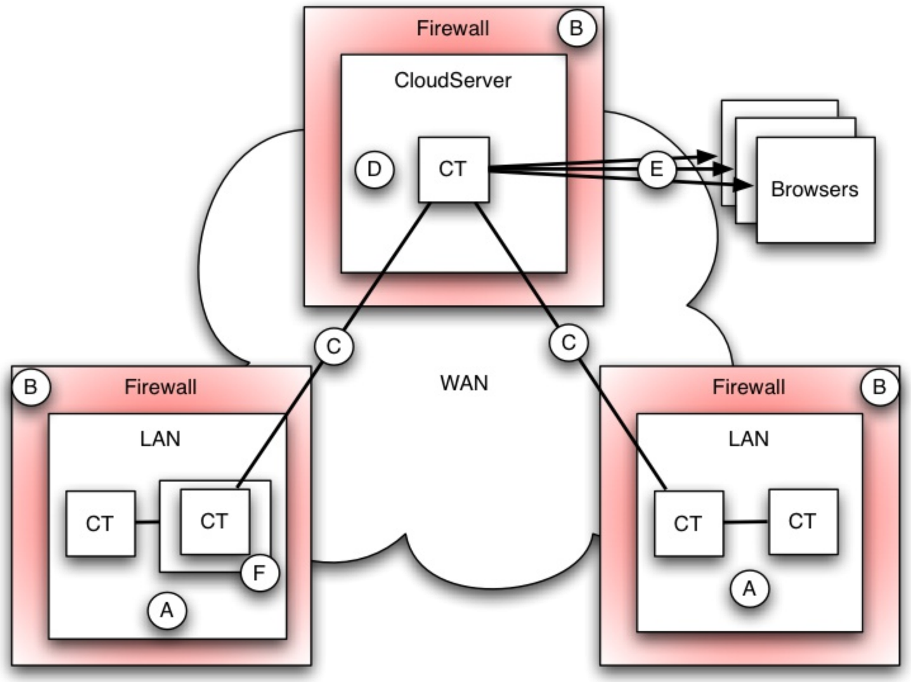

Figure 1: CloudTurbine Security Layers

From Figure 1, each layer has its unique features and opportunities for keeping a CloudTurbine deployment secure:

A) Local Area Network (LAN), data kept inside secure local area network. Optional data encryption at source.

B) Firewalls secure each site including connection authorization and authentication.

C) Internet (WAN) data transport, data secured in-transit with TLS, SSL, VPN, or end-to-end CT data encryption applied by CT API.

D) Security via trusted cloud service. Data at rest unencrypted (enabling path (E)), or encrypted at source (A).

E) Distributed browser access, secured in-transit with HTTPS, site certificates, password access.

F) Locally encrypting file layer; limits interoperability

In the discussion that follows, data security is considered in turn:

- Data at rest stored in a file system,
- Data in use by sources (writers) and sinks (readers),
- Data in transit between systems over network links.

CloudTurbine Data-at-Rest Security

On an individual computer, CloudTurbine security is file system security... so how does CT manage its files, and how best to manage data security of these files?

Full disk encryption

Full disk encryption protects an entire disk, typically requiring an authentication password at boot time. Running processes on such a system transparently see unencrypted data, so this level of protection is more for systems during powered-down state, such as a laptop being transported on business travel.

Firewalls

A network firewall can protect an individual system as well as a local area network. Depending upon the type (if any) of network file transport required, firewall configuration must consider opening associated ports for incoming/outgoing connections, IP filtering, and other firewall related security configuration issues.

Network Drives

Data access via local network configuration, such as network drives and network data backups must be considered. Network drive protocols include:

- SMB - Server Message Block, Windows, Mac, Linux
- AFP - Apple File Protocol, Mac
- NFS - Network File System, Linux
- CIFS - Common Internet Filesystem, Windows

Network file shares are inherently a security liability if access is possible by anyone other than a trusted collaborator. As such, they are almost always limited to within a firewall-secured local network.

CloudTurbine Data In-Use Security

File system security

Modern operating systems (Windows, MacOS, Linux, etc) provide file and folder level security via read/write/execute permissions associated with login users and groups. Since CloudTurbine transparently maps its data into simple files and folders, these OS file protection tools are readily applicable to protecting your CloudTurbine data.

For example, several collections of CloudTurbine data may be arranged into different top-level folders, each of which has associated file system user/group permissions. One collection may have read-only permission, whereas another may allow open read-write permission for users otherwise authorized to access the host computer system.

Figure 2 illustrates one such scenario. A top level folder ("CloudTurbine") contains all CT related data. Under this, several "Project" folders exist, e.g. for different test labs within an organization, or different subscribers to a CT file service. Each project may have various tests, perhaps sequenced by date. Finally, there are Source folders containing CT timestamped session data. Each branch of this hierarchy can be protected by operating system user/group ownership security mechanisms.

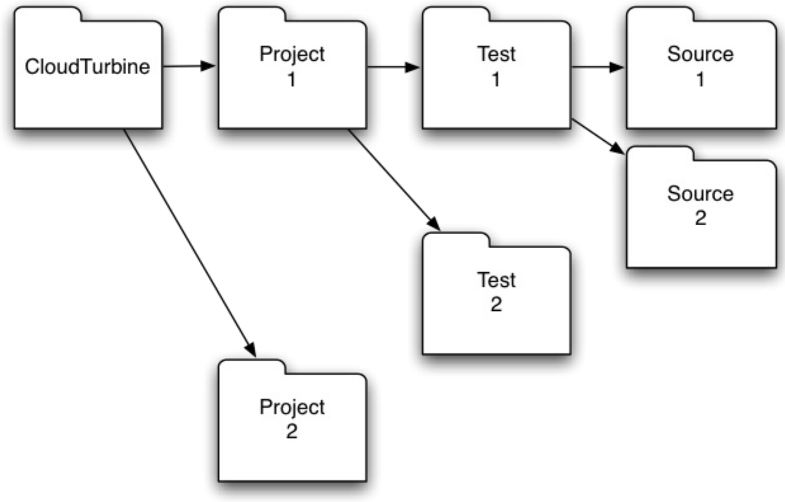

Figure 2: CloudTurbine Data Folder Hierarchy

Sources versus Sinks

With CloudTurbine, each data source has its own top-level data folder. Thus, each CloudTurbine data source may be protected using file-system user/group privileges associated with the corresponding CT source.

A data producer (source) must have write permission to its folders and files, but not necessarily read permission. A data viewer (sink) needs read permission, but not necessarily write permission. In some cases, a single application may be both source and sink, e.g. produces and processes data both. Setting file permissions to enable the minimal required access for each application limits exposure to potential security threats.

Symbolic Links

As CloudTurbine data may entail significant storage and or performance requirements (e.g. SSD drives), it may be convenient to utilize the symbolic link (aka shortcuts) to provide access to CloudTurbine data folders under a particular user's control/access, while retaining its physical location on specific hardware meeting performance constraints.

Figure 3 is an example of the use of symbolic links to connect a sandboxed data source to a CTweb folder that exposes selected data for remote viewing. Security advantages to symbolic linking CloudTurbine folders include being able to restrict access to specific data source folders to specific authorized users (e.g. via FTP access), while still being able to aggregate multiple sources for joint read access to a broader range of (separately authorized) users.

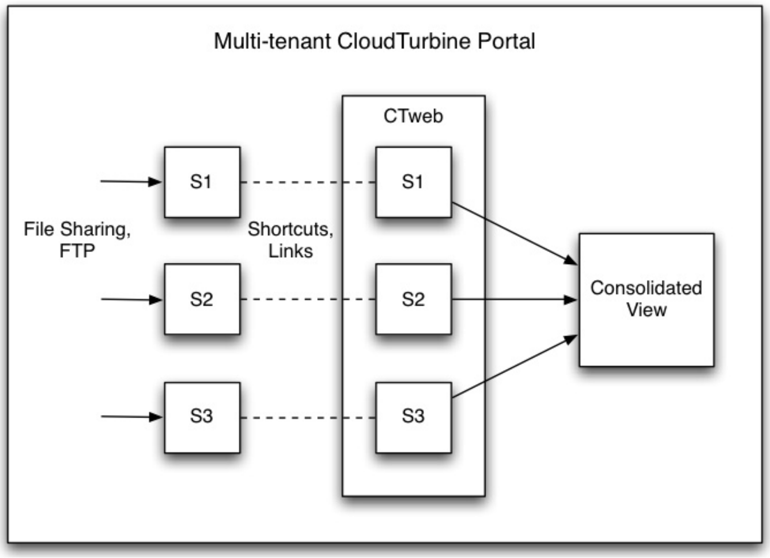

Figure 3: Symbolic Links Isolate Data Source from Viewers

Data Mirrors

Making copies of "raw" source data and allowing access only to the copies can protect against corruption of the original data. There are various mechanisms for producing such mirrors, a widespread one known as "file sharing".

File sharing entails elements of both data-in-use and data-in-transit security, the latter of which is discussed in the following section. File sharing data-in-use typically entails the same issues and considerations as for any other unshared locally stored data files. Depending upon the file sharing approach, it may also entail remote-access and control issues, such as:

- who is authorized to share the data
- how is authorization enforced
- can remote sharers add or delete data to you local store

CloudTurbine Data In Transit Security

CloudTurbine security primarily relies upon third party file layers (e.g. file sharing services) to provide in-transit encryption and security. Whereas the CloudTurbine Java API (CTlib) has some "built in" support for network data transfer via HTTP/S (CTweb) and FTP (CTlib API), these are not meant to be fully secure mechanisms for network data transport. A well designed CloudTurbine implementation (e.g. that utilizes locally secure files and appropriate third party file sharing), is able to provide enduring security without need for continual development and update of the core CloudTurbine API and clients.

Secure File Sharing

CloudTurbine is designed with special consideration given to utilization of file-sharing as the mechanism for distributing streaming data over networks. As such, file-sharing security is of particular interest to CloudTurbine network deployments.

Many file sharing services utilize less secure features, often to save bandwidth (sharing duplicate data), enhance value-added utility (such as document editing from web portal), or to mine content for marketing purposes:

- Encryption keys on server
- Cross-user deduplication

Less common, yet more secure approaches include:

- Explicit client side key holder (e.g. SpiderOak)
- Open source, peer-to-peer (e.g. Syncthing)
- Encryption as a layer (e.g. Boxcryptor)

The selection of a file sharing approach is key to secure CloudTurbine data in transit security. The many file sharing services available each address this issue in their own, often evolving way.

Legacy File Transfer Mechanisms

CloudTurbine network data distribution is designed to be compatible with modern file sharing services, but there are also legacy, often simpler means to share files that may be utilized.

Note that a significant difference with these methods is that they are typically unidirectional, whereas file-sharing systems tend to synchronize updates to/from all connected systems. For an application that only needs (or wants) one-way data flow, this can be advantageous as more efficient and secure.

Note also that these legacy methods typically do not include built-in scheduling. Scripting file transfer on a fixed schedule (cron, launchd) or file system modification (inotify) may be an added step left to the implementor.

FTP

FTP has been around since the early 1970s. The CT API "CTwriter" class has an "CTftp" subclass, which transparently pushes data over an FTP link to a remote (versus local) filesystem. Secure variants of FTP include:

- FTPS - relies on the same basic methodology as FTP, but adds SSL encryption
- SFTP - a more modern variant that sends files over Secure Shell (SSH) protocols

The CloudTurbine API supports FTPS as an option to its CTftp class for encrypted FTP. A secure VPN connection could be implemented to transport common FTP between sites. Alternately, CloudTurbine is amenable to layered file-transport approaches, for example using scripted SFTP.

For the security of SFTP without special programming, setting up a SSHFS (Secure-Shell File System) provides a secure, relatively low-latency way to connect distributed systems. SSHFS utilizes the FUSE (Filesystem in User Space) to network-mount a remote system folder, and automates SFTP file transfers between systems. See https://en.wikipedia.org/wiki/SSHFS.

Rsync

Rsync has been a staple file-synchronizing utility for Unix-like systems since the late 1990s. It uses delta-encoding to minimize network bandwidth, and is compatible with SSH for data security. It is a workhorse utility which requires more technical expertise but provides solid unidirectional file sync.

Web (RESTful) File Transfer

CloudTurbine, based upon files, is very compatible with HTTP (Web) interfaces for accessing data. The "CTweb" application provides a web portal to CloudTurbine data per Figure 1. Whereas early versions of CTweb offered read-only CloudTurbine data access, later versions added the ability to

HTTP-PUT data to the server as well. An example of a secure read-write CTweb server scenario is described in the following "Use Case" section.

The CTweb application is implemented using an embedded Jetty web server. It is designed to offer a browser interface to locally stored data; i.e. CTweb runs on the same machine as both CloudTurbine data and the viewer (browser) resides. It does offer HTTPS encryptions and basic (user/password) authentication options, and can be utilized for outward facing (Internet) access, but only where simple basic security is sufficient.

More sophisticated web interfaces to CloudTurbine are possible, such as via an Apache Tomcat servlet. However, note that a fundamental premise of CloudTurbine is that it offers an efficient mechanism for streaming data via files, where these files are made available at the point of viewing via secure file sharing mechanisms. A central web service over a network link to CloudTurbine data enchroaches on this premise and in so doing foregoes the robust security of data access via a locally secured file system.

End-to-End Encryption

The CloudTurbine Java API provides an end-to-end data encryption option, which when utilized as a layer on top of other security measures, provides very robust security. Features include:

- Open source java.crypto, AES encryption
- Data encrypted in memory by CT API
- Only data owner knows password/key
- Layer on top of OS/File/Cloud security

With a password or encryption key, channel data written to files (or within zip files) are encrypted using AES 128 bit encryption. A unique initialization vector ("nonce") is encoded with every encrypted file.

This means your data (sensor, text, audio, video, everything) is encrypted before it hits the disk, and along its entire path during any file sharing or networking. It is decrypted after being read from disk at authorized remote viewer application(s). Only you own and control the encryption key. This is in addition to any security provisions provided by your site admin or file sharing provider.

CloudTurbine encryption/decryption is performed via the "CTcrypto.java" class as part of the Open Source distribution, using standard published encryption algorithms. Along with all CloudTurbine code, there are no hidden features, magic, or back doors to access your data in-transit.

Multi-Factor Authentication

Per the CloudTurbine paradigm, network data transport is handled by third party services such as file-sharing. These in turn provide a variety of security mechanisms, including multi-factor authentication.

File sharing services typically provide security for both the data-host (provider), and for sharing data with collaborators. Both are important, with the former needed to secure remote administrative access to the account, and the latter to ensure only trusted collaborators have access to the data.

Multi-factor authentication is provided for administrative account access by most file-sharing services. For example, the popular Dropbox service provides for SMS codes, USB keys, and NFC devices in addition to passwords for administrative access to ones account:
https://www.dropbox.com/help/security/enable-two-step-verification

The invitation process for others to view a shared data folder typically involves sending an email or other electronic invitation to one or more known collaborators. Each collaborator needs their own authenticated file-sharing account. The invitation process is secure in that it is initiated by the data-host, and sent only to accounts known and trusted by the host. Different file sharing services have various levels of configurable data access restrictions, such as read-only access, limiting who may invite other users, etc.

Note the advantages of layered security. CloudTurbine streaming via an in-place, secure file-sharing service means simply using it as intended; writing and reading files. Both CloudTurbine and the file sharing service operate independently of each other. In this respect, CloudTurbine seamlessly leverages existing resources including security validation.

Security Use-Cases

Following are evolving discussions of specific use cases, including how CloudTurbine deployments leverage security layers provided by file sharing services such as two-factor authentication.

Case study: CloudTurbine.net AWS server setup

The cloudturbine.net server is hosted on Amazon Web Services (AWS) as an EC2 Linux Virtual Machine (VM). CloudTurbine data sources from multiple users are configured via a variety of connections, including:

- Dropbox
- Syncthing
- FTP

The layered security described for this site do not represent the only nor the most secure approach, but is meant to relate a variety of approaches to be considered when setting up a controlled semi-public data sharing server utilizing CloudTurbine as its backbone data streaming transport mechanism.

File Permissions

In setting up a cloud CTweb server with multi-user data writing privileges, care must be taken to ensure proper file permissions are set.

On the cloudturbine.net AWS server, all associated users are set to "cloudturbine" as their primary group, and "cloudturbine" is given file "rw" and directory "rwx" permissions. For example, users configured for data source access on the cloudturbine.net server include:

- Erigo
- Cycronix
- Vistra

To configure file access properties, run the following Linux commands for each [user]:

```
id -g -n [user]                   # print primary group
sudo usermod -g cloudturbine      # assign primary group
# From each home (login) dir:
sudo chgrp -R cloudturbine ~[user]    # set group all files/folders recursive
sudo chmod -R ug+rwX ~[user]          # rw all files, rwX all dirs, recursive
```

Administrative Access

Access to the AWS server (Linux) machine is via Secure Shell (ssh) using public-key cryptography key pairs. Firewall port configuration can lock and restrict access by port and IP. Optional multi-factor authentication is available, with all the security layers and mechanisms provided by AWS. Remote login is then available via a command such as:

```
ssh user-key-pair.pem ec2-user@12.345.6789
```

CloudTurbine Data Source FTP Access

Utilizing standard Linux FTP configuration, CloudTurbine data sources are enabled to send data to the server via FTP/SFTP to a restricted home sub-directory only. These data folders are linked to provide authorized remote monitoring as discussed earlier.

CloudTurbine Data Source File Sharing Access

Authorized data sources are provided shared folders on the server via associated file sharing service (Dropbox, Syncthing) security authentication/authorization protocols. These in turn are linked to specific sub-folders viewable within the CloudTurbine "CTweb" data hierarchy.

CloudTurbine CTweb Access

Per Figure 4, each user's data source folder is symbolically linked to a common "CTdata" directory which is monitored by the "CTweb" CloudTurbine web-access application. This provides a consolidated view of all configured data sources from a common web interface.

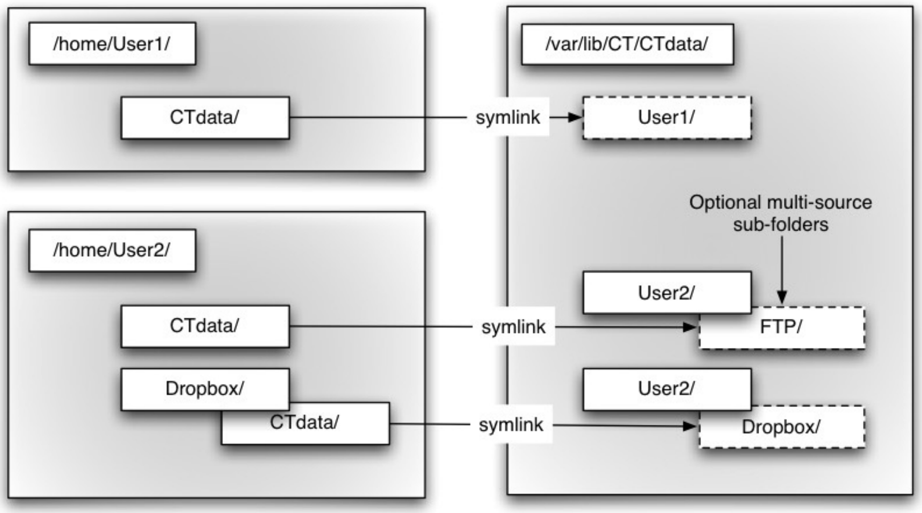

Figure 4: CloudTurbine AWS Server: Source to CTweb Folder Links

Individual or subset groups of data sources are given separate CTweb access via unique ports for each. Each instance of the CTweb application can be configured for HTTP and HTTPS (encrypted) access, and optionally for user name/password (BASIC) web authentication.

Case Study: Read/Write CTweb Server

Whereas the web servers nominally provide read-access to server data, HTTP provides write mechanisms via PUT and POST. Submitting information via web forms is one common example. The CloudTurbine API has extended the CTwriter method ("CThttp") to optionally send data via HTTP versus writing files to the local filesystem. In this configuration, layered security is possible as shown in Figure 5.

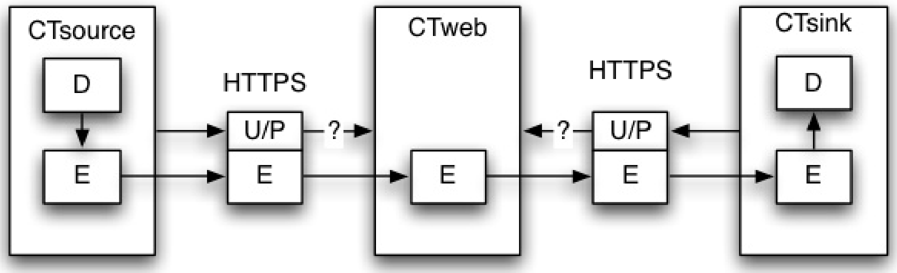

Figure 5: Secure Source-to-Sink Data Transfer Via Read/Write CTweb Server

In Figure 5, data "D" is encrypted "E" at the CTsource. Encrypted data along with user/password ("U/P") is sent securely over HTTPS to CTweb server. Authenticated U/P, encrypted data E is stored on CTweb server, and provided to properly authenticated sinks (viewers). Data from CTweb to CTsink is sent securely over HTTPS, and finally decrypted (E to D) at the end-viewer. The encryption key (password) is known only to the source and sink; the CTweb server has no need nor ability to decrypt the CloudTurbine data itself.

On the CTweb server, the file-based security mechanisms discussed earlier are also applicable. E.g. restricting users to pre-configured, user-specific CTsource folders with appropriate file-system read/write privileges adds another security layer.