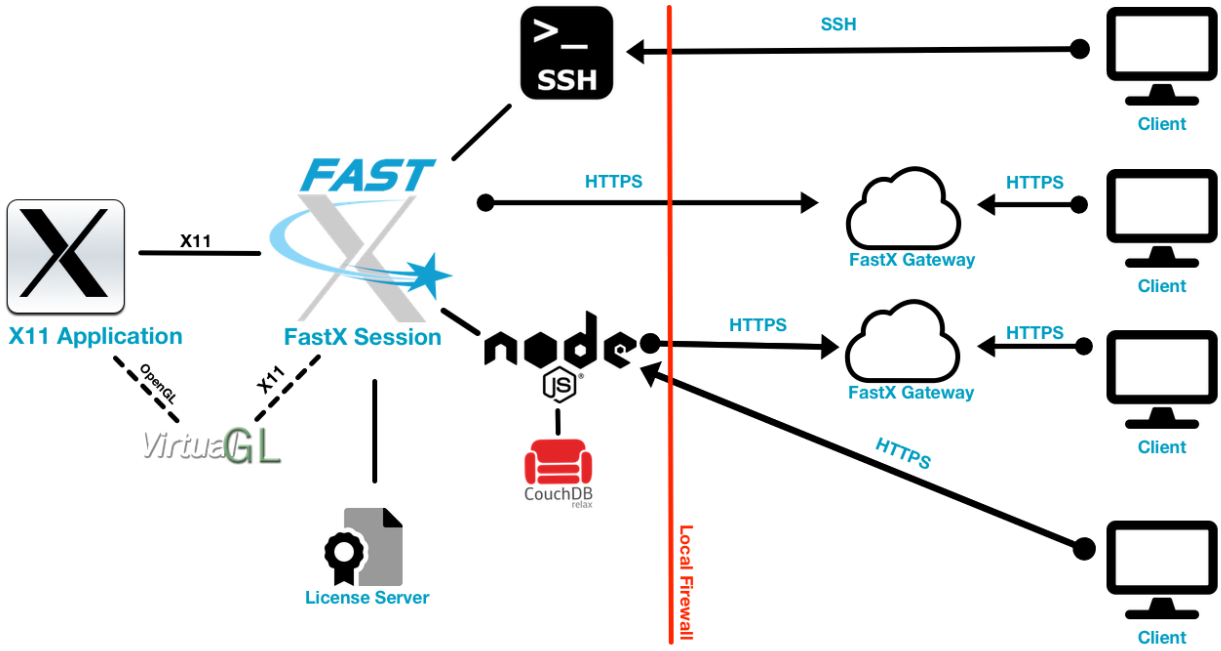
Security Considerations for FastX

Background

FastX is a highly configurable solution for accessing remote Linux desktops and applications from both a browser and a desktop client. FastX aims to maximize the quality of life of users connecting remotely, while maintaining a highly secure environment. Maximizing security in FastX comes with tradeoffs in ease of use, more configuration, and reduced features. Given the diverse nature of FastX users, there is no one configuration that will satisfy all customers. Rather, it is up to the system administrators to balance security concerns with quality of life. This guide aims to provide system administrators with the security considerations of certain features in order to find the right balance for their users.

Overview of the FastX Architecture



FastX consists of a series of components that work in conjunction to quickly and securely deliver Linux applications remotely. Each component has its own security implications so understanding the design can help you make the right decisions for your environment.

FastX Session (X11 Server)

The core component of FastX is the FastX Session. The FastX session is an Xorg X server with custom custom extensions that implement the FastX protocol. Each FastX session creates its own virtual DISPLAY which can be connected to.

SSH Daemon

FastX uses the SSH daemon already installed on your system.
Desktop clients can use SSH to connect to the FastX Session
The Web Server will also use the SSH daemon for user authentication.

NodeJS Web Server

FastX ships a standard NodeJS web server for HTTPS communication. User defined web servers can be configured.

CouchDB Database

FastX ships the CouchDB database for long term storage and cluster synchronization. The CouchDB database listens on the loopback interface only.

CouchDB configuration for FastX is stored by default in */usr/lib/fastx/var/local/db/couchdb.ini*

RLM License Server

FastX ships the RLM license server from Reprise Software for license management. License servers can be local on the system or centralized on a remote server.

VirtualGL

FastX ships the optional component VirtualGL for direct access to video card hardware.

X11 Application

The purpose of FastX is to deliver X11 applications from Linux Servers to Clients. FastX sessions run as long as there is at least on X11 applications connected to the X server.

FastX Gateway

In cluster configurations, multiple FastX installations communicate with each other

Client

FastX has a browser based client that is installed with the server as well as a downloadable Desktop client. The browser client communicates over HTTPS, while the Desktop client can connect over HTTPS or SSH.

Ports

FastX uses the following default ports

SSH: 22
HTTPS: 3300
RLM License server: 5053
StarNet ISV License: 57889

Web Server

The web server is an optional component of FastX. Disabling the web server also disables the majority of FastX features so we recommend running the web server. You may wish to disable the web server if

- you are the only user running FastX
- you want to download the Desktop client

Server Certificates

FastX ships with self signed certificates to get up and running quickly. When it is time to put your system in production, it is a good idea to install a valid certificate from a trusted third party.

Configure the certificates in the System > Local System > WWW section of the web server configuration.

TLS Versions

All network traffic in FastX is encrypted using the NodeJS server TLS module. While the default configuration should be adequate for most users, you can specify your TLS versions in the web configuration.

You can add your own TLS secure context object that the web server will use in
System > Local System > WWW > Https

See TLS createSecureContext for a list of options

Default TLS Min Version

Default TLS Max Version

Client Certificates

FastX does not check client certificates by default. You can force client certificate checks by settings the following two options in System > Local System > WWW > Https

requestCert: true
rejectUnauthorized: true

See TLS createServer for more information

HTTP Headers

Administrators may wish to add custom HTTP headers in FastX. Any header can be added to any web request. Configure custom HTTP headers in System > Web > HTTP Headers

Using a Different NodeJS Server

FastX periodically updates NodeJS web server that is shipped with the installation. You may wish to use a newer version of NodeJS to take advantage of the latest security updates and performance improvements. To use your own version of NodeJS,

edit the file */etc/sysconfig/fastx3*

Add the line:
*NODE_BINARY=**/path/to/your/node***

Other Options

FastX's web server takes all options in HTTPS createServer
You can configure this in System > Local System > WWW > Https

Require Web Server

The FastX web server provides extra security checks when interacting with FastX Sessions. If the web server is down, an attacker could gain previously unauthorized access to his own sessions.

Check *Require web server* in System > Local System > WWW to force the FastX Session to automatically reject any request when the web server is down.

Note: the web server is required for user A to interact with user B's FastX session. If the web server is down, user A will not be able to interact with user B's sessions.

Fixing Configuration Errors

A working https server configuration is required for the web server to run. Otherwise it will exit with an error. This poses a problem in fixing errors when using a web based configuration.

The web server configuration is stored by default in */usr/lib/fastx/var/config/www.json*
Remove this file to get back to the default configuration.

Running the Web Server on Port 443

The well-known HTTPS port is port 443. On Linux, root privileges are needed to bind to port numbers < 1024.

You can use your firewall to redirect traffic from port 80 to the FastX web server port. For example

```
sudo iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 443 -j REDIRECT --to-port 3300
```

User Authentication

A user needs to authenticate before having any access to the FastX infrastructure.
User authentication is documented in the following Guide to Logins

The different types of authentications are documented in Authentication Methods

Multi-Factor Authentication

Multi-Factor authentication (MFA) is supported through the SSH module. If you have set up MFA in your SSH server, FastX will automatically pick that up when authenticating. MFA works when connecting via SSH and HTTPS

Single Sign On with OpenID Connect

Administrators can configure OpenID Connect to allow Single Sign On for the browser client. OpenID Connect creates a Pure Web Token for FastX which means it does not have an associated link process running on the server which is used for launching new sessions. The web server will launch a link command. See Link Daemon.

Link Daemons

A link daemon is a process running as the user that the web server uses to execute commands as the user. Most typically this is used for starting new sessions. Link daemons stay running in the background waiting for connections from the web server. The web server and link daemon have secret shared JSON Web Tokens used for authorization. Link daemons will typically run until the JWT expires (default 1 week). Interacting with the link daemon will refresh the expiration.

When a user starts a session, the following occurs

- IF the user is connecting using SSH and starting a session on the same system, start the session using fastx-protocol (the FastX SSH session runner)
- ELSE if there is a running link daemon on the system connect and launch the session
- ELSE try to use sudo to launch the link daemon
- ELSE try to use ssh@localhost to launch the link daemon
- ELSE fail

The user generates a link daemon automatically when the user authenticates using the web-ssh method.

A link daemon stays running until it times out. This means that users do not need to double authenticate every time they launch a session. Only if there is no link daemon running. This would typically be a week without link interaction (logging in, launching a new session etc).

When launching sessions in a cluster, or when using the OpenID Connect authentication method, the only way to prevent periodic double authentication is to launch a sudo link

Sudo Links

If the owner of the fastx process has proper permissions, it will sudo to a different user and try to launch a link. To enable this feature, the “fastx” user must be given permission to run the link script as a “regular” user.

A typical system will have the following line in */etc/sudoers*

#includedir /etc/sudoers.d

Create a file */etc/sudoers.d/fastx3*

Add the following lines

```
Runas_Alias FASTX_USERS = ALL, !%root, !%bin, !%daemon, !%adm, !%mail
fastx ALL=(FASTX_USERS) NOPASSWD: /usr/lib/fastx/3/scripts/link
```

If you want to disable the sudo check

1. go to System > Local System > Settings
2. Check: Disable sudo commands

Permissions

FastX 3.1 implements a permissions system for FastX users. Permissions allow for fine tuning of user actions limiting what a user can do. Permissions are based on Linux user group names. For more information on permissions see the Permissions Guide.

Executing Commands on a Session

Users with the Exec permission can execute arbitrary commands on their session. Admins who are attempting to limit what a user can do should disable this permission.

Connecting to Someone Else's Session

Users with the manager or admin level permissions can connect to a running session of another user. This allows the manager to take control of the user session and act as the other linux user. This level of permission should be restricted to as few users as possible (for example the wheel group). Admins can disable the manager ability to connect to other user sessions in System > Users > Permissions > Manager Permissions

Note: Admins always have all permissions enabled.

Sessions

The abilities of a session are defined in the session profile System > Sessions > Profiles. Different sessions can have different profiles. Profile options with security considerations are documented below.

Enable Client Clipboard

This option allows the FastX client (or browser) to send clipboard data to the FastX server. Most administrators allow data to flow in to the server, but not out. This option can typically stay enabled.

Enable Server Clipboard

This option allows the FastX server to send clipboard data to the FastX client. A client can then take that data and use it on the client machine. Administrators who worry about data leakage should disable this option.

Server Clipboard Max Bytes

Administrators may wish to limit the length of the clipboard data that can be sent from server to client. Set this property to the maximum number of bytes you want to limit.

Enable X11 TCP Support

X11 servers listen for incoming X connections from X clients to display graphics. Most applications connect to the DISPLAY on the local system. However some applications (most notably load balancers) may request a remote display. This option enables the FastX X11 server to listen for incoming TCP connections from remote systems. Connecting to an X11 DISPLAY allows an application to capture the contents of the display, as well as display windows. If an attacker can do this, he would be able to see potentially confidential information, or display a fake password prompt. The X11 system has an authorization “cookie” that prevents this, but disabling this option, which limits connections to local, adds additional security.

Extensions

Extensions are implemented by the ClientComm channel. This is a general purpose channel that allows applications on the Linux system to use a running session as a channel to execute actions on the Client machine. Administrators may consider disabling the channels that are not in use. For more information see the ClientComm guide

Event Scripts

Event scripts are arbitrary scripts that are executed when some event happens on the session. It is important to note that the scripts are executed **as the user** of the FastX session.

Clustering

Grouping multiple FastX installations across multiple servers allows FastX to form a cluster. Cluster members securely communicate with each other in order to interact with sessions, load balance etc. For a full discussion of clustering see our Clustering Guide.

Database

FastX uses a distributed syncing CouchDB database instance for clustering. In order to be part of the cluster, a cluster member must either connect to the installation, or the installation will connect to the cluster member. Database connections communicate over HTTPS. The database must be kept secure as a compromised database can affect all members of the cluster.

The database is used for syncronizing sessions, updating valid servers and storing configuration.

Messages

Messages between two cluster members are either sent directly as an HTTPS request, or indirectly if the server is marked as external. External servers maintain a Server Sent Event stream with all members of the cluster.

All messages are encrypted via HTTPS. Also, all message data is further encrypted using the private key of the gateway. The session server decrypts the message with the gateway's public key and then handles the request.

Database Connections

These are the outgoing connections to other database members. As the database is shared, you do not need to connect to all cluster members. 2 - 6 database connections is typical to ensure proper syncing and fault tolerance.

HTTP Agent

By default, Database requests do not check server certificates. This is to simplify configuration. In production, all cluster members should have valid SSL certificates. When creating a database connection set

```
 {
   "rejectUnauthorized": true
}
```

to force certificate checks.

Cluster Keys

Cluster members use shared secret keys to enable access and communicate with each other. The shared secret creates a JSON Web Token (JWT) that is transferred as an HTTP header so the secret never leaves the system. The web server will create a JWT for each shared key in

the cluster configuration. A cluster member is considered valid if at least one of the JWTs is valid.

Client Verification

After a JWT has been verified, the request will be forwarded to the custom verify function. Here you can add custom checks to verify that the client server is valid.

Server is external

External servers are documented in the clustering guide. Create an external server to allow public access to cluster members on private networks. The cluster members must be able to connect to the external server. All HTTP requests will initiate from the cluster members.

Disable Direct Client Connections

Clients will try to short circuit their cluster connections by taking the most direct route to the end point. If a session server is on a private network, this is guaranteed to fail. Enable this option to force all connections through a gateway and never short circuit the connection.

Require valid SSL certificates for all cluster members

This option requires that all cluster members have valid SSL certificates. This option also has a side effect of disabling the Xorg server from connecting out to make a connection to the gateway. Instead a NodeJS sub process will act as a proxy.

Use Authorized Keys directory for cluster keys

By default, cluster members store their public keys in the shared synced database. The database is assumed secure. However, if the database is compromised, an attacker can put overwrite the public keys with his own. The attacker can also change the server id and in general cause problems with the database. In order to mitigate some of the damage, an administrator can enable the public keys to be stored off line in a directory. The cluster member will ignore the public key in the database in favor of the one in the directory. If the public key is not in the directory or is different, then the message can not get decrypted. This way the attacker cannot connect to other systems.

Miscellaneous

Stop Message Propagation to iframe parent window

FastX integrators often wrap the FastX browser client in an iframe to package FastX with their own software. By default, protocol events are propagated up to the iframe so the integrators can handle the event. Security conscious users may wish to disable this feature in System > Customization > Client Settings