DataLock

SYSCTL AB



Contents

Introduction **3**

Definitions **3**

Client usage and SSH keys **4**

The DataLock's own key pair - used to send files onward . . . 4

Setting up a receiver . . . 4

The sender's authentication key - used to log in to DataLock . . . 5

Setting up a sender . . . 5

Generating a key pair on Windows . . . 5

Generating a key pair on Linux and macOS . . . 5

Uploading files . . . 6

Checking your uploads in the queue . . . 6

Scan reports **8**

Limitations **9**

File sizes . . . 9

Supported protocols . . . 9

Performance . . . 9

Troubleshooting **10**

Logging in as `datalock-status` gives `Permission denied (publickey)` . . . 10

Uploading gives `Permission denied (publickey)` . . . 10

`scp` fails while `sftp` works . . . 10

`PTY allocation request failed on channel 0` . . . 10

An upload is not in the queue view . . . 10

Note: the queue has not reported since 11

Introduction

DataLock is a member of the Impex product family, alongside USB Protect and Xtransfer. While USB Protect scans files carried on removable media at a physical kiosk, DataLock performs the same kind of scanning and control for files moving across a network boundary between two zones, for example between an office network and a protected network. It acts as an application-level firewall for files: it inspects the files themselves, not just the network packets carrying them, and only allows them through if they pass every configured check. It runs as a service on a scanning appliance, physical or virtual, rather than as dedicated kiosk hardware, and it is managed centrally from the ICC in the same way as any other Impex station.

Files are uploaded to DataLock over the network and checksummed (MD5, SHA1 and SHA256), scanned by multiple antivirus engines, and checked against any configured Yara rules and file filters. If all checks pass, the files are forwarded on to a configured destination together with a signed report. If a check fails, the report is sent onward instead of the file, and the file itself can be quarantined depending on how that specific DataLock is configured.

The complete path a file takes, from a source to its destination, is called a **flow**. A single DataLock can run several flows at once, each with its own destination and its own sender keys, see Client usage and SSH keys below.

This document describes DataLock from a client's point of view: how files are transferred to it, how the SSH keys involved in that process work, and the scan reports it produces.

Configuration of DataLock flows and destinations is done entirely from the ICC web interface, see the ICC manual for that. Hardware, network and firewall requirements are covered in the Prerequisites document, and the installation procedure is covered in the Installation manual.

Definitions

Word	Definition
Impex	The family name of USB Protect, Xtransfer, ICC, Repo and DataLock
Stations	The family name of USB Protect, Xtransfer and DataLock
ICC	The server which controls the Stations and DataLocks
USB Protect	The kiosk computer used to scan mass storage devices
Xtransfer	USB Protect with support for sending files to network shares
Repo	The server that has the updates and definitions
DataLock	Server used for network flows that will scan files before transferring them onwards
Network flows	The description of data being scanned and transferred through the DataLock to a remote destination. A DataLock can have multiple remote destinations configured
Flow	A single configured route on a DataLock: one or more sender SSH keys paired with one destination
Operators	The users who will use the administrative interface on the ICC server

Client usage and SSH keys

DataLock uses SFTP (SSH File Transfer Protocol) for file transfers. This runs over SSH and is authenticated with the SSH key pairs described below. It uses the same protocol in both directions: a sender uses SFTP to upload files to DataLock, and DataLock uses it to deliver files onward to a destination. SFTP is what most graphical clients such as WinSCP use by default, and it is also what the command line `sftp` tool speaks. The `scp` command can be used as well, but only on reasonably modern clients, see Supported protocols. Interactive shell login is not possible.

Two separate SSH key pairs are involved whenever a file passes through a DataLock, and it is easy to confuse them because both are, in a loose sense, "an SSH key used by DataLock." They serve opposite purposes.

The DataLock's own key pair - used to send files onward

Each DataLock station has its own SSH key pair, created when the station is set up. The private key never leaves the DataLock, and it is the key DataLock itself authenticates with when it forwards files onward to the destination server. The public half can be downloaded from the "more details" view on the station's card in the ICC.

That public key needs to be handed to whoever manages the destination server, so it can be added to the destination's list of authorized keys. This is a one-time step done when a flow is created: without the DataLock's public key being trusted by the destination, DataLock has no way to deliver files to it.

The scan reports are signed with a separate key, the same way USB Protect and Xtransfer sign theirs, see Scan reports.

Setting up a receiver

Before a flow can deliver files, the destination server must be configured to trust the DataLock that will be sending to it.

1. Download the DataLock's public key from the "more details" view on the station's card in the ICC.
2. On the destination server, add that public key to the `authorized_keys` file of the account the flow will upload as, normally located at `~/.ssh/authorized_keys` for that user.
3. Make sure the username and target directory entered in the flow's configuration in ICC match the account and path that was actually set up on the destination server.

Once the key has been added, DataLock can authenticate to that account and deliver files to it. If a destination has multiple DataLocks delivering to it, for example because several DataLock instances share the same remote server, each DataLock's public key must be added to the destination separately.

The sender's authentication key - used to log in to DataLock

Every user or system allowed to upload files to a DataLock authenticates with its own SSH key pair. An administrator uploads the *public* half of that key pair into the ICC and attaches it to a specific flow. From that point on, whoever holds the matching *private* key can upload files through that flow, and the flow determines which destination host and directory the files are ultimately delivered to.

In short:

- The **DataLock's own key** authenticates the delivery to the destination.
- The **sender's key** authenticates the upload from the sender to DataLock.

A single DataLock only ever has the one key pair of its own, but it can have many different sender keys configured across many different flows.

Setting up a sender

1. The sender generates an SSH key pair on their own machine.
2. The sender gives the *public* key to an ICC administrator, who uploads it and links it to the correct flow.
3. The sender can now authenticate to the DataLock and upload files through that flow.

DataLock does not identify senders by username, every sender connects as the same account, `datalock`. Authentication is by SSH key only: what distinguishes one sender from another, and therefore which flow an upload belongs to, is exclusively which private key was used to connect. Two different senders, each with their own key but both connecting as `datalock`, are still handled as two separate senders and routed through their own separate flows, purely on the basis of which key they presented.

Generating a key pair on Windows

To generate a new SSH key pair on a recent Windows machine, open a terminal and run `ssh-keygen`, then choose a name for the key.

When prompted for a passphrase, consider whether the key will be used by an unattended script rather than a person (a machine-to-machine scenario). If so, a passphrase may not be practical to use, though there are ways to store private keys securely in such scenarios beyond a plain passphrase.

Generating a key pair on Linux and macOS

Both Linux and macOS ship with an OpenSSH client by default, no additional software needs to be installed. Open a terminal and run:

```
$ ssh-keygen -t ed25519
```

Accept the suggested default location (`~/.ssh/id_ed25519`) or choose a different name, and decide on a passphrase using the same considerations as for Windows above. The command produces two files: the private key, and a matching file with a `.pub` extension, the public key, which is the one to hand to the ICC administrator.

Uploading files

Files are uploaded to DataLock over SFTP. The command line `sftp` tool speaks it directly, and the `scp` command can also be used provided the client is new enough, see Supported protocols. Always connect using the username **`datalock`**, as shown in the examples below, the sender's own identity is established by their private key, not by the username.

Command-line SFTP example:

```
$ sftp -i privkey.key datalock@datalock-ip
Connected to datalock.
sftp> put /home/user/Downloads/somefile.exe
Uploading /home/user/Downloads/somefile.exe to /somefile.exe
somefile.exe                                  100% 5144KB 165.7MB/s   00:00
sftp> quit
$
```

Command-line `scp` example, which requires OpenSSH 9.0 or newer on the sending client:

```
$ scp -i privkey.key -r somefiles datalock@datalock-ip:
```

On Windows, the default `sftp` client shipped with the OS is quite old, we recommend installing a newer OpenSSH client, for example with:

```
c:\> winget install Microsoft.OpenSSH.Beta
```

A GUI alternative is WinSCP[1], where the private key is selected the same way the `-i` option is used on the command line.

Checking your uploads in the queue

A DataLock scans one upload at a time, so after a large upload, or when several senders upload at once, an upload can wait in a queue before it is scanned and sent onward. A sender can see where their own uploads are in that queue by logging in over SSH as **`datalock-status`**, using the same private key they upload with.

This queue view is **off by default**. It has to be turned on by an ICC administrator with the setting *Uploader Queue View (DataLock)* on the configuration card the DataLock uses, see the ICC manual. While it is off, the `datalock-status` login is refused with `Permission denied (publickey)`, even for keys that can upload.

[1] GUI variant of SFTP tool for windows available here https://winscp.net

```
$ ssh -i privkey.key datalock-status@datalock-ip
Your uploads on this DataLock at 16:31:30

  UPLOAD            STATE
  1 file, 806.9 kB  scanning
  1 file, 806.9 kB  waiting since 16:31
  1 file, 806.9 kB  waiting since 16:31

2 other uploads are in the queue.
```

Each of your uploads is listed with its number of files, its size, and whether it is being scanned right now or since when it has been waiting. Uploads made with other keys are only counted, nothing else about them is shown, and the same goes for your uploads when someone else looks. Which uploads are yours is decided by the key you logged in with, so an upload only shows up when you log in with the key it was uploaded with. Times are shown in the DataLock's own time zone.

An upload leaves the view once it has been scanned. Delivery to the destination happens after that and is not shown here; the scan report that travels with the files tells the outcome.

A word after the address chooses what the view does, for example:

```
$ ssh -i privkey.key datalock-status@datalock-ip watch
```

Word	What it does
(none)	Prints the queue once and logs out, the same as `once`
`once`	Prints the queue once and logs out
`watch`	Prints the queue again on every change, until none of your uploads are left
`follow`	Prints the queue again on every change, until you disconnect
`help`	Shows these choices

`watch` and `follow` end with Ctrl-C, and on their own after one hour; log in again to keep watching. If the line *Note: the queue has not reported since ...* appears, the DataLock's queue has stopped updating the view and may not be running, contact the DataLock's administrator.

The `datalock-status` login only ever prints the queue. It gives no shell, accepts no input and cannot read or change any files. It does not allocate a terminal either, so a client that asks for one, as `ssh` does when no word is given, may print `PTY allocation request failed on channel 0` before the queue. That message is harmless and can be avoided with `ssh -T`.

Scan reports

Every transfer through a DataLock produces scan reports, which are delivered to the destination in a `.impex` folder beside the uploaded files. If the scan did not pass, the files are not delivered and the `.impex` folder is all that reaches the destination, so the outcome is known there even in setups where the destination sits behind a data diode and would otherwise have no way of confirming that a blocked file was ever sent.

The reports are the same as the ones USB Protect and Xtransfer save on USB, and are chosen with the same "Save Report to Destination" report types in the configuration in the ICC: PDF, CSV and JSON. A DataLock always sends at least the PDF report, also when no report type is chosen. The reports list the files in the upload with their checksums (MD5, SHA1 and SHA256), what the antivirus engines found, which key uploaded it and where it was sent, and for each engine whether it completed its scan and how long it took. An engine that failed is marked as such, so a scan that passed with one engine missing can be told from one where every engine looked at the files.

The PDF report lists at most 1000 files, with the files that have malware hits listed first, and says so when the upload holds more. The CSV and JSON reports have no such limit and always list every file in the upload. Since a DataLock sends only the PDF when no report type is chosen, choose CSV or JSON as well if the destination needs the complete file listing.

The JSON report contains all the information about a scan, this is, among other things:

- `uuid`: the ID of the operation
- `result`: `passed` or `not passed`
- `files_count` and `malware_count`: the number of files scanned and malware detected
- `start_time` and `end_time`
- `upload_ip`, `upload_username` and `upload_fingerprint`: where the upload came from and who uploaded it
- `engines`: each antivirus engine with its `name` and a `status` which is `done` for an engine that finished successfully
- `files`: each file that was scanned with its `file_name`, `file_size`, `md5`, `sha1`, `sha256` and `malware_name`, which is empty when nothing was found
- `config`: the Datalock's configuration during the scan

```
fil1.txt
fil2.txt
.impex/
.impex/key.pub
.impex/key.pub.icc-sig
.impex/scan_report.pdf
.impex/scan_report.pdf.sig
```

If the upload itself already contains a `.impex` folder, it is delivered untouched and the reports are put in `.impex1` instead (or `.impex2`, and so on).

A DataLock always signs its reports. Each report has a signature beside it, made with the DataLock's report signing key, `key.pub`, and that key is in turn signed by the ICC, `key.pub.icc-sig`. Together with the checksums in the reports, this proves that the files passed through the DataLock

and were not altered afterwards. The reports are verified the same way as those from USB Protect and Xtransfer, see “Sign Saved Reports” in the ICC manual. In short, with the ICC’s public key downloaded from the “Server Settings” view in the ICC:

```
$ signify -C -p icc_signify_pub -x key.pub.icc-sig key.pub
$ echo -n "datalock $(<key.pub)" > allowed_signers
$ ssh-keygen -Y verify -f $PWD/allowed_signers -n report-sign@sysctl.se \
-I datalock -s scan_report.pdf.sig < scan_report.pdf
Good "report-sign@sysctl.se" signature for datalock with [ ...]
```

Limitations

File sizes

Problems transferring files of a certain size can usually be traced back to the size of the disk partition Impex uses to intermediary-store files before forwarding them to the destination. Make sure capacity planning for the disk accounts for the expected file sizes and volumes as part of setting up the DataLock.

Supported protocols

DataLock only supports file transfers over SFTP, carried over SSH. DataLock does not allow interactive SSH login. The only other login is the optional queue view, see Checking your uploads in the queue, which prints the queue and accepts no input.

The `scp` command is not a separate protocol on the wire, it is a client that picks one. As of OpenSSH 9.0 it uses SFTP by default, so it works against DataLock unchanged. Older releases instead use the legacy SCP protocol, which DataLock does not accept, so `scp` on those clients will fail. Passing the `-O` flag forces that legacy protocol even on a new client and will fail for the same reason. There are two workarounds:

1. Upgrade to an OpenSSH 9.0 or newer client, where `scp` uses SFTP by default.
2. Use the `sftp` tool instead of `scp` on that client.

Performance

DataLock performance depends on a mix of factors:

- which AV scanners are used
- how many AV scanners are used
- the size of files in a session
- the number of files in an upload session
- the specification of the hardware hosting the DataLock
- the configuration of the virtual environment, if set up as a virtual appliance

Understanding your specific setup and use case is important for optimizing the actual performance of a flow. Consult with sysctl if you have questions or issues about optimizing your setup.

Troubleshooting

Logging in as `datalock-status` gives `Permission denied (publickey)`

The queue view is turned off. It is off by default, and only an ICC administrator can turn it on, with the setting *Uploader Queue View (DataLock)* on the configuration card the DataLock uses. While it is off, every key is refused, also keys that can upload.

To tell this apart from a problem with your key, connect with the same key to upload:

```
$ sftp -i privkey.key datalock@datalock-ip
```

If that connects, your key is fine and the queue view is off: ask the DataLock's administrator to turn it on. If it is refused as well, see the next section.

Once the setting is saved in the ICC, the DataLock picks it up with its next configuration update, normally within seconds.

Uploading gives `Permission denied (publickey)`

The DataLock does not accept the key you connect with. Check that

- you connect as `datalock`, the only account for uploads,
- you use the private key whose public half you gave the ICC administrator, for example with `-i privkey.key`,
- the administrator has uploaded that public key into the ICC.

A key uploaded into the ICC reaches the DataLock with its next configuration update.

`scp` fails while `sftp` works

Your `scp` uses the legacy SCP protocol, which DataLock does not accept, see Supported protocols.

`PTY allocation request failed on channel 0`

`ssh` prints this before the queue when you log in as `datalock-status` without a word after the address. It is harmless, the login does not give a terminal, and `ssh -T` avoids it.

An upload is not in the queue view

The view only lists the uploads made with the key you logged in with, and an upload leaves it once it has been scanned. Log in with the key the upload was made with; if the upload is still not listed, it has been scanned, and the receipt that travels with the files tells the outcome.

Note: the queue has not reported since ...

The DataLock's queue has stopped updating the view and may not be running. Contact the DataLock's administrator.