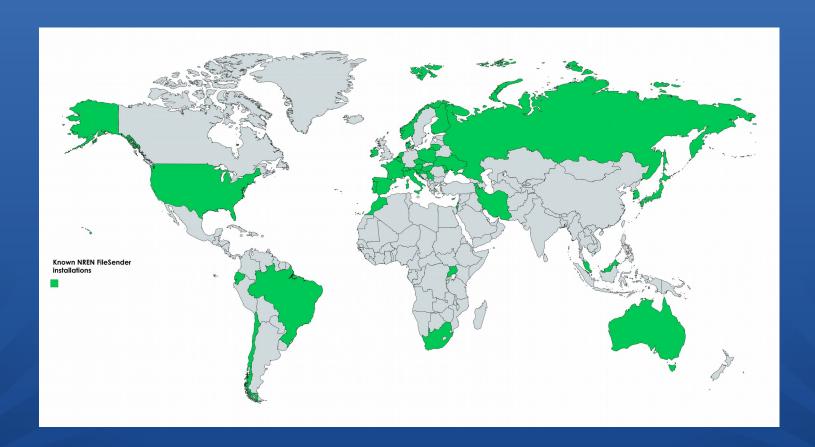
FileSender

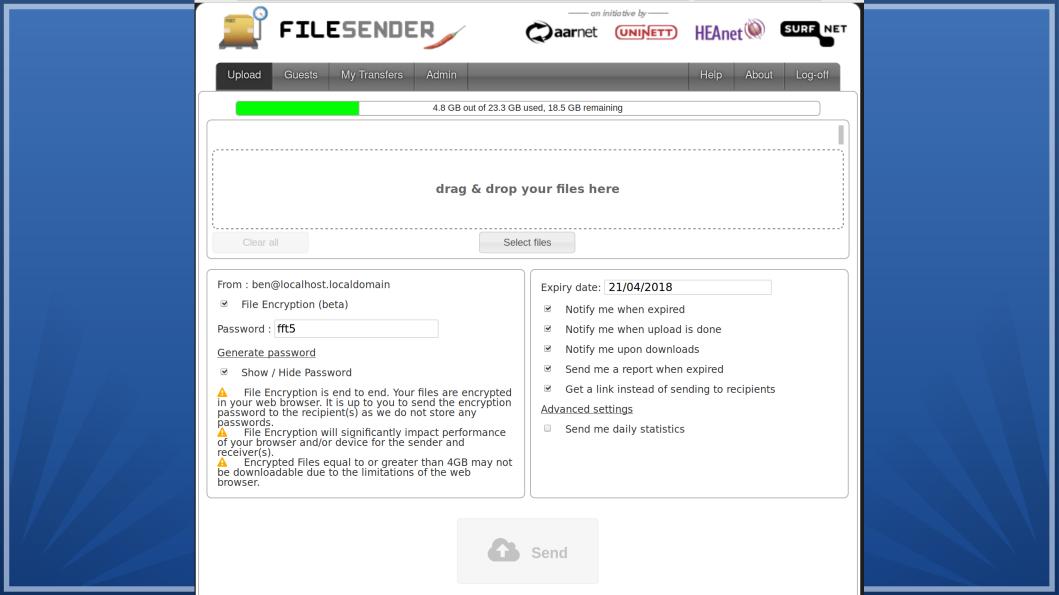
Browser based large file sharing

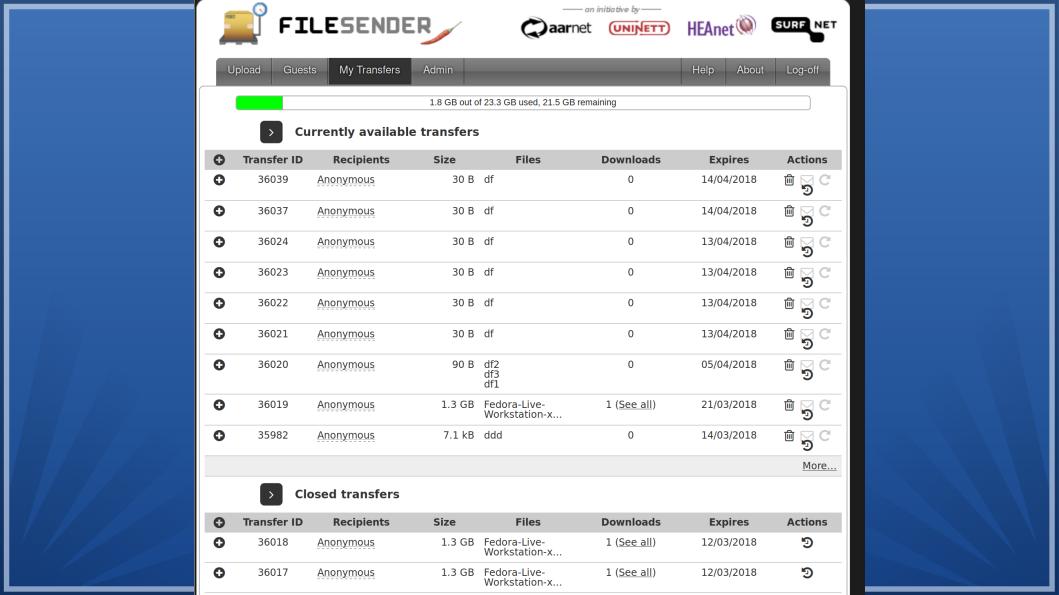
The Commons Conservancy

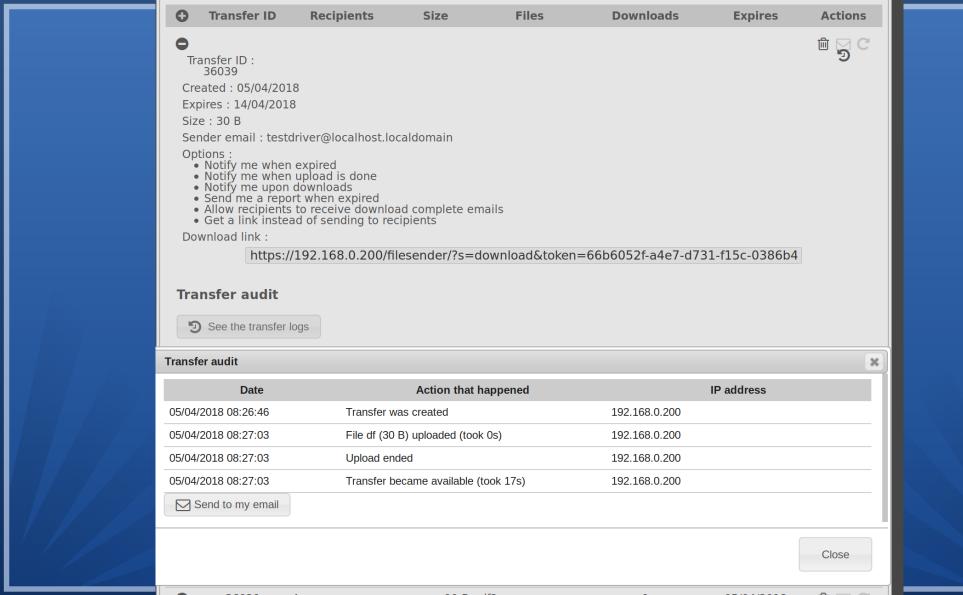
Dr Ben Martin

Used?









Why

- User only needs Web browser
- Full control of content
- Upload resume
- Notification on upload, download, etc
- Share with explicit groups
- Browser-to-Browser encryption of data AES-256
- SAML for auth scale
- Allow guests to upload to your server
- GDPR by default, auto generated "about privacy" page

How

- Server side is PHP
- Client side JS with light widgets
- Database is Maria 10.0+ or PostgreSQL \$recent
- Web server is Apache or NGINX
- Auth is SAML or hacked^TM

Chunks

- Minimal range of bytes to commit to server
- Size of memory needed in browser (blob.``slice")
- Somewhat hidden from users

Uploading

- TeraSender Web Workers
 - Number of chunks upload at once
- PUT chunks directly
- PUT wholefile
- REST client

Server Storage

- Contiguous file
- Chunked (5mb) files
 - Above either raw or in nesting path for NFS dispatch
- External script
- Ceph (aarnet)
- Cloud Azure, S3

Downloading

- One or more files listed per transfer
- On the fly zip64 archive creation
- Links for console download if desired

Dragons

- Robust, auto resume and fast upload cross browser is HARD
- eg, Disable wifi during upload
- Mixed browsers and racing
- Long uploads can exceed auth session times
- Web crypto support W3C

Database Design

```
userpreferences
        <----- transfer -> files
                 recipients
 guests
shredfiles,
auditlogs (subj, pred, action),
trackingevents (pred, action)
statlogs, new aggregatestats
translatableemails (class, id => data)
```

Database fun

- Moved to synth id for userprefereneces pk
- Moved saml authid to another table
- Added RI about the place
- Added secondary indexes
- Added views to ease query maintenance
- TODO

- Userpreferences/guest cleanup
- PHP ↔ SQL worlds

ORM

```
protected static $dataMap = array(
  'id' => array(
     'type' => 'uint',
     'size' => 'medium',
     'primary' => true,
     'autoinc' => true
  'email' => array(
     'type' => 'string',
     'size' => 255
```

ORM

All(), delete(), FromDB(), toDB() save()

client class override __get() and 'reach out' db migration based on metadata

DBI Smoothing

```
public static function datediff( $f1, $f2 ) {
  if(self::isPostgress()) {
     return "extract(day from " . $f1 . "-" . $f2 . " )";
  if(self::isMySQL()) {
     return "DATEDIFF(" . $f1 . "," . $f2 . ")";
  throw new DBIBackendExplicitHandlerUnimplementedException(
     'SQLUNIMP datediff() called on unsupported backend');
```

Uploading

TeraSender overview

<u>TeraSender</u>

- Create a number of Web Workers
- Logically files split into 'chunks' each of which is given to a worker
- Each chunk is committed server side on success so we can resume the whole upload* at any time
- Workers can work on a single file at once or across multiple files
- Encryption is done before upload

TeraSender upload

- upload_page.js onClick() → ui.startUpload()
 - → Transfer.start()
- One of these depending on config & browser
 - → terasender.start()
 - → uploadByChunks()
 - → uploadWholeFile()

Teraender.start(transfer)

- Start()
 - → createWorker() many times
- createWorker()

```
w = new Worker()
```

w.OnMessage();

w.sendCommand('start')

Worker.start

- OnMessage (start) → requestJob()
- RequestJob() → sendCommand('requestJob');

Ts.giveJob

```
var job = this.allocateJob(worker);
if( job ) {
  this.sendCommand(workerinterface, 'executeJob', job);
} else {
  workerinterface.status = 'done';
  workerinterface.terminate();
```

ts.allocateJob

- If worker already has a file keep it on that file
- Otherwise find a file with data to upload
- The 'job' is the next 'chunk' of the file to upload

Worker.executeJob

Data = Slice the file.blob to get the chunk

Xhr = createXhr()

Data = Encrypt Data if desired

Xhr.send(Data)

xhr.onreadystatechange()

 \rightarrow ok, auth token, error (sendmsg), restart?, 200 \rightarrow s(jobExecuted)

xhr.onprogress() → sendmsg(progress)

ts.jobExecuted

- Call evalProgress() to update UI progress data
- Fall through to requestJob() in case statement

Security

- Mixed auth model
- Email a group or "get a link"
- E2E crypto

Crypto

Currently each blob crypted with aes-256-cbc using a random iv

→ GCM + AEAD

crypto_app().encryptBlob(array, password, cb)

Encrypt (current)

```
generateKey(password, function (key, iv) {
 crypto.subtle.encrypt({name: $this.crypto crypt name, iv: iv}, key,
value).then(
   function (result) {
      var joinedData = ....crypto common()
           .joinIvAndData(iv, new Uint8Array(result));
       var btoaData = btoa(abtoString(joinedData));
       callback(btoaData);
```

Password \rightarrow { key, iv }

```
generateKey: function (password, callback) {
 var iv = getRandomValues(new Uint8Array(16));
 crypto.subtle.digest({name: this.crypto hash name},
                      stoabv(password)).then(function (h) {
 crypto.subtle.importKey("raw", h,
   {name: $this.crypto crypt name, iv: iv},
   false, ["encrypt", "decrypt"]).then(function (key) {
            callback(key, iv);
```

Genergate Key 2018

```
crypto.subtle.importKey( 'raw', UI.get( password),
     {name: 'PBKDF2'}, false, ['deriveBits', 'deriveKey']).then(function(dkey) {
       crypto.subtle.deriveKey(
         { "name": 'PBKDF2', "hash": 'SHA-256', "salt" saltBuffer },
         dkey,
         { "name": 'AES-CBC', "length": 256, iv: iv },
        false, // key is not extractable
         [ "encrypt", "decrypt" ] // features desired
            ).then(function (key) {
              callback(key, iv);
```

Decryption

```
generateKey(password, function (key) { ...
var value = ...separatelvFromData(encryptedData[i]);
crypto.subtle.decrypt(
  {name: $this.crypto_crypt_name,
   iv: value.iv},
   key,
   value.data).then( ... );
```

Encryption future

- Current: PBKDF2 (Password-Based Key Derivation Function 2)
- Future: password AND
 - doc id
 - Transfer id
 - sender email
 - other immutable data
- User passphrase: Selectable hashing level
- Using generated key from entropy with user readable display format (recommended)

Encryption GCM (future)

Key + IV <u>must</u> always be unique counter <u>must</u> never wrap hardware support for GCM

128 bit IV = 96 bits random() and 32 counter. 5mb chunks gives 20tb files

GCM operation

```
key = hased_password_or_secure_generated();
rand96 = randbits(96);
counter = 0
iv = combine( \{ rand96, counter \} )
otp = Encr( key, iv, plaintext=counter )
ct = pt XOR otp
counter++
```

I18ns

```
language/en_au/lang.php
$lang['about'] = 'About';
$lang['about page'] = 'About';
language/en au/guest cancelled.mail.php
A voucher from {guest.user_email} has been cancelled.
```

I18ns

import-translation-for-language.php en_au poeditor.export.php

export-all-terms.php terms.txt convert-one-per-line-terms-to-json.php terms.txt terms.json send terms.json to poeditor

Future Directions

- UI refresh
- Maybe mobile app, "send to" filesender
- Integration to endpoints (auto youtube etc)
- E2E encryption from CBC to CGM (NIST SP 800-38D)
- Docker images for easy setup and go. Migration docs for database upgrades in existing docker images
- More SAML info
- Maybe run the whole thing in Cloud as possibility
- Session clone?