

Forum: [Hack Forums](#)

Thread: [files.elmnts.co - Encrypted File Storage](#)

money.team - March 26, 2016, 5:26 pm

files.elmnts.co

Easy, drag and drop, no bullshit file sharing

- Files are stored encrypted on the server, keys are in the download URL
- Optional self destruct on first download mode
- Serve any type of file
- UNIX upload script (more info in spoiler)

Spoiler [\(Click to View\)](#)

Install Dependencies

The UNIX upload script uses cURL and wget, these packages are typically installed by default. If they are not, install them:

Debian/Ubuntu

Code apt-get install curl wgetCentOS/Fedora

Code yum install curl wget

Install the Script

Download and setup the script by executing the command below.

Code wget files.elmnts.co/upload_script -O /bin/upload && chmod 755 /bin/upload

Upload your First File

Uploads are as simple as can be using this script. Give it a try!

Code upload <filename>Note: add the -d option if you would like to enable self destruct mode.

Code upload -d <filename>

Demo GIF: [i.elmnts.co/i/s3dP.gif](#)

Lukeidiot - March 26, 2016, 6:04 pm

If these files are encrypted, how come you can directly link to an image?

<http://files.elmnts.co/dl/oei4xmeu3di/amfico.png>

I see no decryption when the image is loaded. If you are storing that key in your database then there is no point to encrypt them. This is pointless if you store the key lol.

money.team - March 26, 2016, 6:09 pm

(Mar 26, 2016 06:04 PM)Lukeidiot Wrote: [\[CITE\]](#) If these files are encrypted, how come you can directly link to an image? [files.elmnts.co/dl/oei4xmeu3di/amfico.png](#)

I see no decryption when the image is loaded. If you are storing that key in your database then there is no point to encrypt them. This is pointless if you store the key lol.

Did you even read the post? "keys are in the download URL", so in your case oei4xmeu3di is the key, the key is used by the server to decrypt the file then send it to the client, then it is discarded.

Sikes - March 28, 2016, 5:28 pm

Vouch for files.elmnts.co, use it for my private and h*****}cking related files ?

resent - March 28, 2016, 8:27 pm

(Mar 26, 2016 06:09 PM)money.team Wrote: [\[CITE\]](#) Did you even read the post? "keys are in the download URL", so in your case oei4xmeu3di is the key, the key is used by the server to decrypt the file then send it to the client, then it is discarded.

Which means all files can be decrypted by you or anyone else that gains access to the server. good job, you ruined the point of encryption.

money.team - March 30, 2016, 4:46 pm

(Mar 28, 2016 08:27 PM)?eko Wrote: [\[CITE\]](#) Which means all files can be decrypted by you or anyone else that gains access to the server. good job, you ruined the point of encryption.

The autism is off the charts. The point of encryption in this case is to prevent somebody from reading the files directly off of the server. Its pretty obvious that there isn't end to end encryption, that would require clientside decryption, which is not possible with direct linking. This is the best we can do in this situation.

resent - March 30, 2016, 7:54 pm

(Mar 30, 2016 04:46 PM)money.team Wrote: [\[CITE\]](#) The autism is off the charts. The point of encryption in this case is to prevent somebody from reading the files directly off of the server. Its pretty obvious that there isn't end to end encryption, that would require clientside decryption, which is not possible with direct linking. This is the best we can do in this situation.

Then what's the point? If someone has access to your server to view all of the files, they have access to all of the private keys.

It's like having a safe but having the combination written on the door in big clearly readable letters

pastenib - March 31, 2016, 6:09 am

(Mar 30, 2016 07:54 PM)?eko Wrote: [\[CITE\]](#) Then what's the point? If someone has access to your server to view all of the files, they have access to all of the private keys.

It's like having a safe but having the combination written on the door in big clearly readable letters
Maybe I'm missing something, but I don't think the keys are saved on the server at all. It sounds like the only copy of the key was given to the uploader in the URL so they could share it with people, that's assuming that the server owner isn't secretly logging all the keys. However, because SSL isn't in use, the key is basically useless due to the fact that anyone monitoring the network can see it.

So, to use your analogy, it's like having a safe but whenever someone puts the combination in, they also shout it out loud at the same time.

Unless the web server has full access logs (thus logging the keys), I think this service would be quite secure from someone breaching the server and stealing the files[SEE MY EDIT], but not particularly secure for the people actually downloading/uploading stuff.

This site relies on trust that the server owner isn't logging the keys, but that's the case for a lot of services that don't have end-to-end encryption.

The site does actually have a working SSL certificate so you just need to put https at the start of the URL, but for some reason it doesn't force people to use it or even suggest people use it.

Update: I noticed the code is on Github so I've been looking through some of it. I've only looked at two files so far and have already found a couple of things that concern me:

Code `var key = Math.random().toString(36).substring(6);` That is the code that generates the key. `Math.random()` is NOT secure for anything related to encryption, it's not very random, in fact, it's pretty predictable.

Secondly, when the file gets decrypted, it's stored on the disk in plain text, gets sent to the client, and then the plain text version is deleted:

```
Code encryption.decryptFile(req.params.filename, req.params.key, false, function(tempfile) {  
.....
```

```
encryption.removeFile(tempfile);
```

There's a few reasons why this is a serious problem:

A) What if the server loses power before the tempfile is deleted, or if the code throws an exception/crashes? Then it will stay in plain text permanently on the server.

B) Computers don't actually delete files immediately so there'd be recoverable plain text content all over the disk, just run a recovery program and a bunch of files will be readable without even decrypting.

C) If a malicious user somehow gained access to the server, they could just make a script to copy all the tempfiles to somewhere permanent when they get created.

Basically, the encryption is useless. Sorry, didn't mean to shit all over your site, but it really needs some more work doing to make it properly secure.

money.team - March 31, 2016, 5:01 pm

(Mar 31, 2016 06:09 AM)pastenib Wrote: [\[CITE\]](#) Maybe I'm missing something, but I don't think the keys are saved on the server at all. It sounds like the only copy of the key was given to the uploader in the URL so they could share it with people, that's assuming that the server owner isn't secretly logging all the keys. However, because SSL isn't in use, the key is basically useless due to the fact that anyone monitoring the network can see it.

So, to use your analogy, it's like having a safe but whenever someone puts the combination in, they also shout it out loud at the same time.

Unless the web server has full access logs (thus logging the keys), I think this service would be quite secure from someone breaching the server and stealing the files[SEE MY EDIT], but not particularly secure for the people actually downloading/uploading stuff.

This site relies on trust that the server owner isn't logging the keys, but that's the case for a lot of services that don't have end-to-end encryption.

The site does actually have a working SSL certificate so you just need to put https at the start of the URL, but for some reason it doesn't force people to use it or even suggest people use it.

Update: I noticed the code is on Github so I've been looking through some of it. I've only looked at two files so far and have already found a couple of things that concern me:

Code `var key = Math.random().toString(36).substring(6);` That is the code that generates the key. `Math.random()` is NOT secure for anything related to encryption, it's not very random, in fact, it's pretty predictable.

Secondly, when the file gets decrypted, it's stored on the disk in plain text, gets sent to the client, and then the plain text version is deleted:

```
Code encryption.decryptFile(req.params.filename, req.params.key, false, function(tempfile) {  
.....
```

encryption.removeFile(tempfile);There's a few reasons why this is a serious problem:

A) What if the server loses power before the tempfile is deleted, or if the code throws an exception/crashes? Then it will stay in plain text permanently on the server.

B) Computers don't actually delete files immediately so there'd be recoverable plain text content all over the disk, just run a recovery program and a bunch of files will be readable without even decrypting.

C) If a malicious user somehow gained access to the server, they could just make a script to copy all the tempfiles to somewhere permanent when they get created.

Basically, the encryption is useless. Sorry, didn't mean to shit all over your site, but it really needs some more work doing to make it properly secure.

Thanks for the constructive post. My intention of encryption here is to prevent ALL of the files from being dumped if it were ever hacked. SSL was not forced because I'm using LetsEncrypt and they are not trusted in every browser yet.

I will push changes to git for the RNG and decrypting files to memory only.

pastenib - April 1, 2016, 7:42 am

(Mar 31, 2016 05:01 PM)money.team Wrote: [\[CITE\]](#) Thanks for the constructive post. My intention of encryption here is to prevent ALL of the files from being dumped if it were ever hacked...

Yeah, I understand, making it more secure wouldn't hurt though :)

They were pretty small mistakes, but could definitely cause huge problems if the server was to get breached.