Are you sure that you still need passwords?

Daniele Vantaggiato
CTO
SingleID Inc.
“The situation got out of hand”

He just wanted to solve two problems:

1. Avoid people nosing around in everybody's files

2. Four hours usage limit
Yesterday

No remote access
Storage was expensive

Today

Remote access (WWW)
Storage has never been cheaper
Any login follows this scheme

```php
<?php
$data = $pdo->prepare('SELECT * FROM users_table WHERE id = :id');

// ...

if (password_verify( $_POST[password], $data[hashed_pwd]) ) {
    do_something(); // Logged
} else {
    die('Sorry. Wrong Password');
}
?>
```
A change as small as you can imagine
How do we do that?

in three simple steps
Step 1: Front-end

1. Install the app and input your data
Step 1: Back-end

- Data is stored encrypted on the device only
- Decryption key is on our servers
Step 2: Front-end

2. Whoever wants to read data directly from the user device must install a button "Login With SingleID"
Step 2: Back-end

```php
<?php
$fields = array(
    'SingleID' => $_POST['single_id'], // the value typed in the button
    'UTID' => $_SESSION['SingleID']['hash'], // 32 hex char string
    'url_waiting_data' => $protocol[$ssl] . '://'. $SERVER['HTTP_HOST']
);

$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, SINGLEID_SERVER_URL);
curl_setopt($ch, CURLOPT_POSTFIELDS, count($fields));
curl_setopt($ch, CURLOPT_HTTPHEADER, $headers);

$result = curl_exec($ch);
```

then the iframe refreshes itself looking for a file called like the UTID
Step 3: Front-end

3. A push notification will wake-up the app with a YES/NO question.
Step 3: Back-end

The data sent from the user’s device will be stored as a JSON array in a file named with the identical value of the UTID.

Like that the other session that refreshes itself can now read the data sent from another browser/device/session.
Tell me one more time!

Recipient System

SingleID Server
- Creates data request and session token
- Checks and then forward to

User Device
- Receives data, creates push notification and delivers to
  - If User accepts...
    - ...the session token and the data request are returned directly from User Device to Recipient System
Advantages

**Decentralized**
Your identity (as set of attributes related to an entity) is not under the control of any organization.

**Granularity**
I can give you permission to read my birthday or my phone number but not my credit card details and you can have this data only for this transaction.

**Password-less**
Nothing to remember. Not even a single master password.

**No more data-breaches**
Because websites can now delete sensitive data of their customers at the end of the session.
Consequently

Our servers do not know or own, not even in an encrypted way, user data.

Our solution is as close as possible to the 25 features that a perfect authentication method should have according to “The quest to replace passwords” – University of Cambridge.

With our solution a user must indeed never create, memorize, see or type a password again.
The password-less trend (1)
The password-less trend (2)

Passwords are Obsolete
And they make Heartbleed a thousand times worse
The password-less trend (3)

Per Thorsheim retweeted

Colin Percival @cperciva

Either I haven't been paying attention, or this is new: Log in to your AWS account via phone number instead of email.

Sign In or Create an AWS Account

What is your e-mail or mobile number?

E-mail or mobile number:
The trick?

Decentralize data storage and resend the entire identification data set to the recipient system every time.
IDENTIFICATION AND/OR AUTHENTICATION METHOD

Applicant: Daniele Vantaggiato, Mestre-Venezia (IT)

Inventor: Daniele Vantaggiato, Mestre-Venezia (IT)

U.S. CL.
CPC ........................................... H04L 9/3213 (2013.01)

ABSTRACT
A computer-implemented authentication method allows a user having a personal electronic device (PED) to login to a recipient system. The user establishes an account on an authentication server, provides personal information to the PED, and uniquely identifies the user’s PED. The authentication method establishes a unique token ID for the user. Upon interacting with the recipient system, the user is prompted for his token ID. The recipient system communicates with the authentication server to request the user’s information. The authentication server sends an authentication request to the user’s PED, which prompts the user for a decision to proceed or not. The user, if deciding to proceed with authentication, selects a subset of the user’s personal information that is then sent to the recipient system by the PED, the recipient system authenticating the user thereby.
The good news

The app is already done for 99% of the platforms*

Simple, open and documented API

Login button on Github
What do you think?
www.singleid.com

daniel@singleid.com

White-paper and Playground on https://www.singleid.ovh