# Secure your app with 2FA

Rob Allen ~ September 2015

# Your users' passwords will be leaked

(It might not even be your fault)

# Passwords have leaked from

| | | | |
|---|---|---|---|
| Sony | Zappos | JP Morgan | Gap |
| Facebook | Twitter | eBay | AT&T |
| Adobe | Target | Blizzard | Drupal |
| Evernote | Yahoo! | Steam | Ubisoft |
| Kickstarter | Home Depot | TK Maxx | Vodafone |
| HP | AOL | Citigroup | WorldPay |
| Gmail | last.fm | Apple | SnapChat |
| Ubuntu | D&B | Formspring | Betfair |

It will take 14 minutes* to crack one of your users' passwords

(English word, stored using bcrypt)

# Two-factor authentication protects your users

# What is 2FA?

# Implementation on a website

# How do we send the code?

# Email

- Used by Steam
- Wide adoption (everyone has an email address!)
- Likely failures: delivery problems, blocking, spam etc
- Usually slow!
- Same system as recover password…

# SMS

- Used by Twitter & LinkedIn
- Wide adoption
- But, SMS can be delayed & could cost to receive

# Physical device

- Used by banks, YubiKey, Blizzard, etc
- Small, long battery life
- But, expensive

# App

- Easy to use
- No Internet or cellular connection required
- App is free and trusted

# One Time Password algorithms

# HOTP

- HMAC-based One-Time Password algorithm
- Computed from shared secret and counter
- New code each time you press the button
- RFC 4226

# HOTP algorithm: step 1
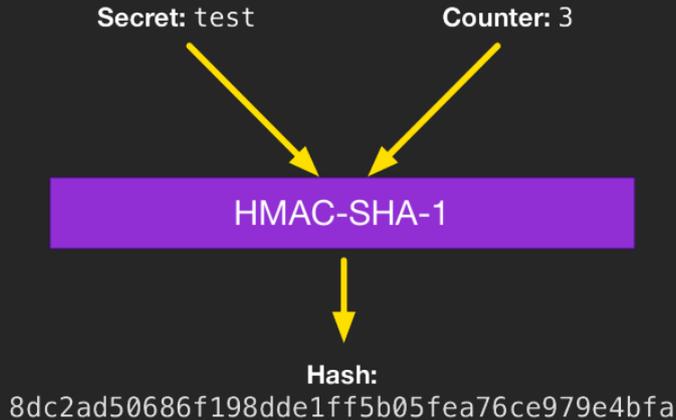
# HOTP algorithm: step 2

**Hash:**
8dc2ad50686f198dde1ff5b05fea76ce979e4bfa



Find lower 4 bits of last byte

8dc2ad50686f198dde1ff5b05fea76ce979e4bf**a**

**offset:**
0x0A (hex) => 10 (decimal)

# HOTP algorithm: step 3

**Hash:**
8dc2ad50686f198dde1ff5b05fea76ce979e4bfa

**offset:**
10

Extract 4 bytes starting at offset (10)

8dc2ad50686f198dde1f**f5b05fea**76ce979e4bfa

Mask top bit & extract last N digits

75b05fea (hex) => 1974**493162** (dec)
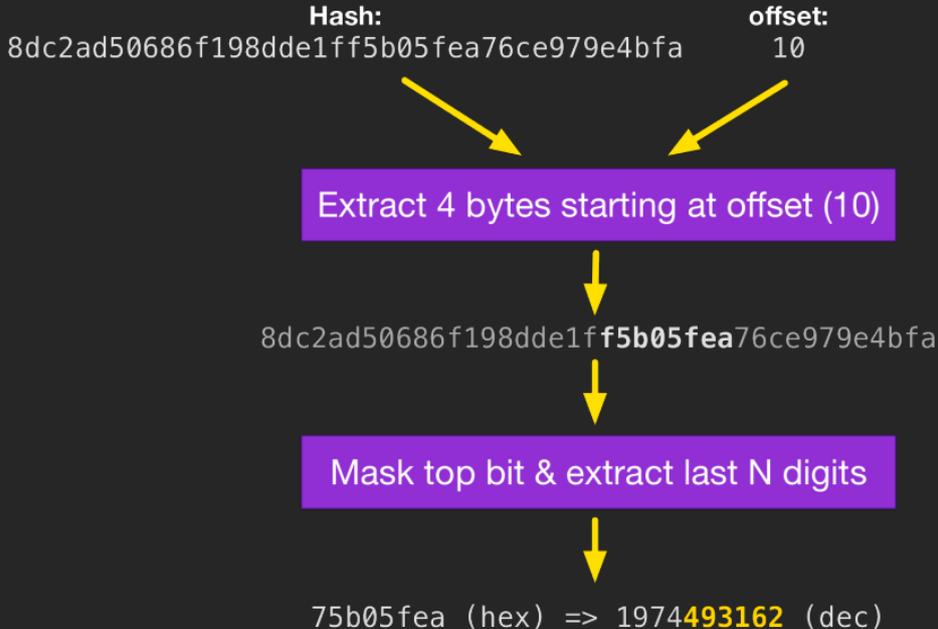
# HOTP in PHP

```php
function hotp($secret, $counter)
{
  $bin_counter = pack('J*', $counter);
  $hash = hash_hmac('sha1', $bin_counter, $secret, true);

  $offset = ord($hash[19]) & 0xf;

  $bin_code =
      ((ord($hash[$offset+0]) & 0x7f) << 24 ) |
      ((ord($hash[$offset+1]) & 0xff) << 16 ) |
      ((ord($hash[$offset+2]) & 0xff) <<  8 ) |
      (ord($hash[$offset+3]) & 0xff);

  return $bin_code % pow(10, 6);
}
```

# TOTP

- Time-based One-Time Password algorithm
- Computed from shared secret and current time
- Increases in 30 second intervals
- RFC 6238

# TOTP Algorithm

**time:** 1441048014        **period:** 30 seconds

Round time down to period boundary

**T:** 48034933        **Secret:** test

HOTP algorithm
(repeat for T−1 & T+1)
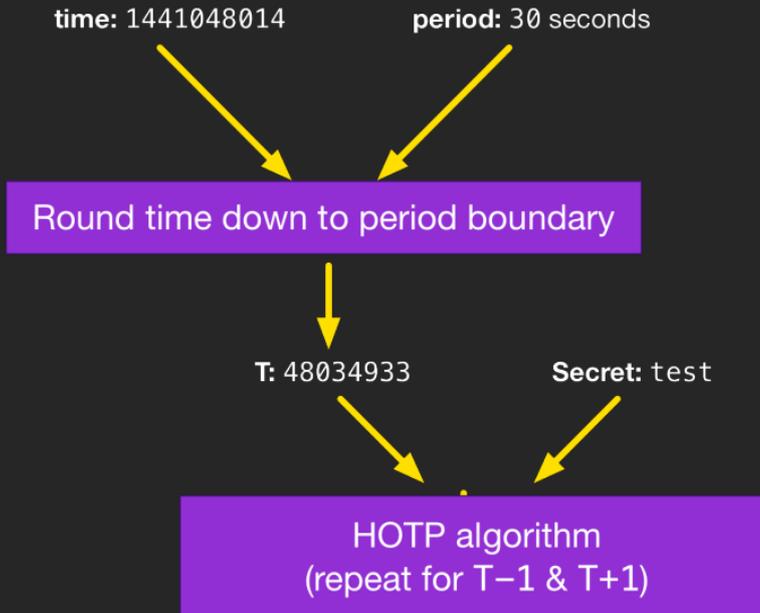
# TOTP in PHP

```php
1 function totp($secret)
2 {
3     $counter = floor(time() / 30);
4
5     return hotp($secret, $counter);
6 }
```

# Implementing 2FA
## in your application

# Things to do

1. Registration of Authenticator
2. Check 2FA TOTP code during login

# Registration

User enables 2FA on their account

# Registration

User enables 2FA on their account

↓

Site generates secret key and then displays QR code & confirmation field

# Registration

User enables 2FA on their account

↓

Site generates secret key and then displays QR code & confirmation field

↓

User adds to Authenticator

# Registration

# Registration



User enables 2FA on their account

↓

Site generates secret key and then displays QR code & confirmation field

↓

User adds to Authenticator

↓

User enters Authenticator 2FA code into site to complete registration

↓

Site confirms code & stores secret to database

# Logging in

After login form, prompt for 2FA code

# Logging in

After login form, prompt for 2FA code

↓

User enters 2FA code

# Logging in

# Logging in

# Logging in

After login form, prompt for 2FA code

↓

User enters 2FA code

↓

Site generates own 2FA code based on user's secret and current time

↓

Site compares own 2FA code with user's supplied one

↓

User allowed into website

# Coding it

```
$composer require sonata-project/google-authenticator
```

Usage:

```php
$g = new \Google\Authenticator\GoogleAuthenticator();

// create new secret and QR code
$secret = $g->generateSecret();
$qrCode = $g->getURL('rob', 'akrabat.com', $secret);

// validation of code
$g->checkCode($secret, $_POST['code']);
```

Example project: https://github.com/akrabat/slim-2fa

# Registration

# Login

# Round out solution

- Prevent brute force attacks
- Consider adding a "remember this browser" feature
- Need a solution for a lost/new phone

# Hardware OTP: YubiKey

# Operation

1. Insert YubiKey into USB slot
2. Select input field on form
3. Press button to fill in OTP field
4. Server validates OTP with YubiCloud service

# Yubikey's OTP

cdccccetfjnfigkgkgudlkcbjnjnfrkhkbelbhfvnhcj
cdccccetfjnfuuhtrebhjekcciljdbifgfrlccbnjhkf
cdccccetfjnfljrhetkvijcgddkenbtuiknktejgnvgb



**Public id**

**Yubico OTP**

6 byte private identity field
Counter
Timer field
Random number
CRC16 checksum

# Coding it

```
$composer require enygma/yubikey
```

Usage:

```php
$v = new \Yubikey\Validate($apiKey, $clientId);
$response = $v->check($_POST['yubikey_code']);

if ($response->success() === true) {
    // allow into website
}
```

# Pre-built plugins

# Pre-built plugins

Drupal:

- Two-factor Authentication
- Yubikey

WordPress:

- Google Authenticator
- yubikey-plugin

Joomla:

- Built-in!

# Drupal

# Set up

# Set up

# Log in

# WordPress

# Set up

# Log in

# Joomla

# Log in

# Questions?

Rob Allen - http://akrabat.com - @akrabat

# Thank you!

Rob Allen - http://akrabat.com - @akrabat