Mobile Authentication

One Rack-App At A Time

DONAL ELLIS

twitter?
facebook?
donal@getperx.com

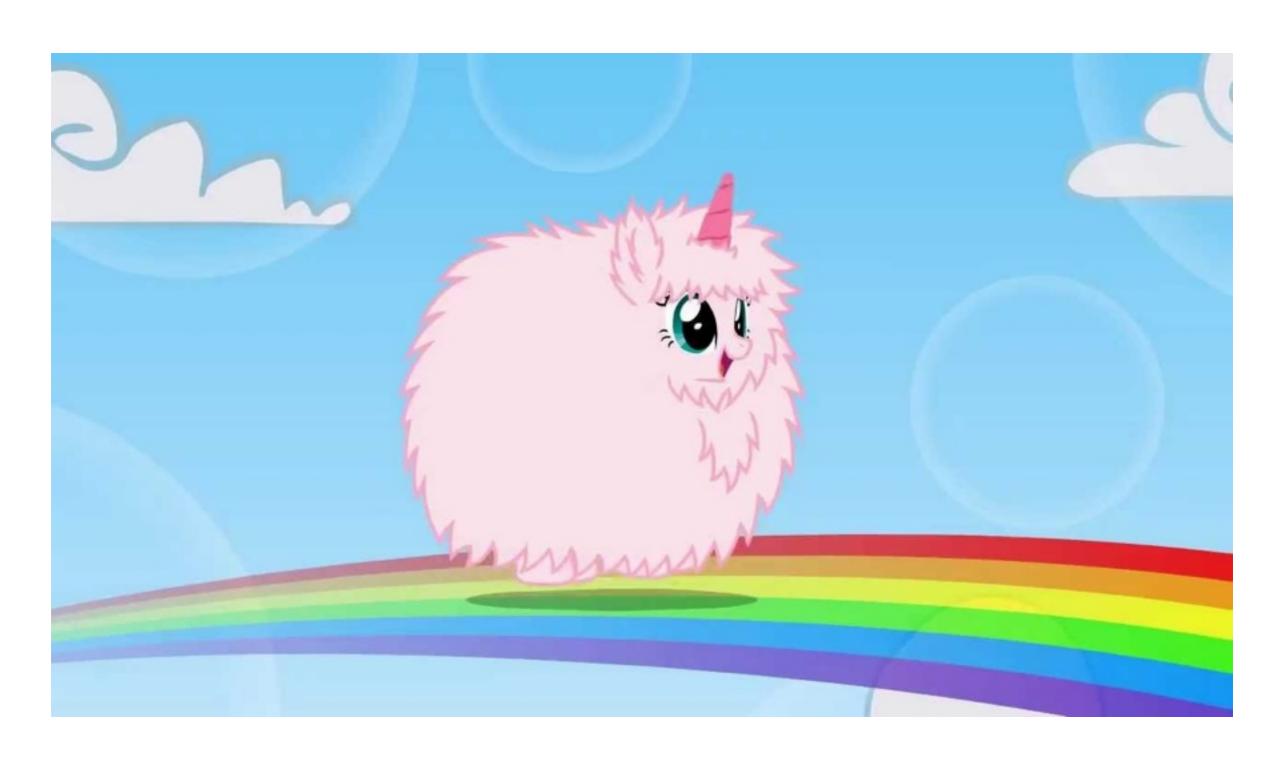
https://github.com/heptat/mobile-auth



From...



То...





NGINX

ubuntu®



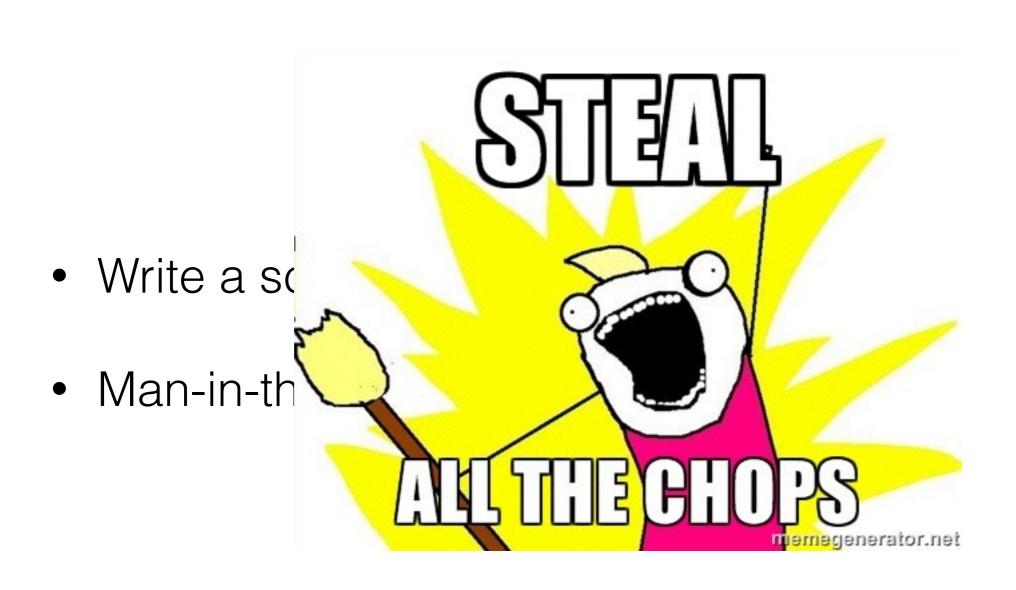
Web App for Mobiles

- API only (JSON)
- No Sessions
 - Every request must authenticate itself

Authentication

- Identify:
 - User (consumer_key)
 - Device (udid)

Secure Against?



OAuth 1.0a

(http://oauth.net/core/1.0a/)

- 1. Generate signature on client
- 2. Send nonce, timestamp, signature, consumer_key (but no udid in OAuth1.0a)
- 3. Generate signature on server and compare

2 Legged OAuth

- signature (hash) is generated using HMAC-SHA1
- secret_key is used as input
- secret_key ships with client application

Secure Against

- Write a script to mimic requests
- Man-in-the-middle attacks

Where's the OAuth?

- No token exchange
- Users can't grant/remove permission

OAuth1.0a vs. 2.0

- 1.0a
 - requires signatures
- 2.0
 - requires SSL
- Do both

API: login/register

- 1. Send your credentials (username, password)
- 2. Send auth data in Authorization header:
 - 1. nonce, timestamp, udid, signature
- 3. (You could send udid in User-Agent header)
- 4. Get back consumer_key

API: all other endpoints

- 1. Send auth data in Authorization header:
 - 1. nonce, timestamp, udid, signagure
 - 2. Plus send consumer_key
- 2. Get back (whatever you ask for)

Authorization Header

```
Authorization: OAuth realm="http://photos.example.net/",
   oauth_consumer_key="dpf43f3p2l4k3l03",
   oauth_token="nnch734d00sl2jdk",
   oauth_signature_method="HMAC-SHA1",
   oauth_signature="tR3%2BTy81IMeYAr%2FFid0kMTYa%2FWM%3D",
   oauth_timestamp="1191242096",
   oauth_nonce="kllo9940pd9333jh",
   oauth_version="1.0"
```

Client Side: Signature

- secret_key ships with phone app
- Build parameter string (see OAuth1.0a protocol) from outgoing data
 - Both Authorization header and request params
- Use secret_key as input to HMAC-SHA1 to encrypt parameter string
- Include signature in Authorization header

Server-Side: Signature

- Build parameter string (see OAuth1.0a protocol) from incoming data
 - Both Authorization header and request params
- Use secret_key (stored on server) as input to HMAC-SHA1 to encrypt parameter string
- Compare signatures

Server-Side: Timestamp

- "The timestamp value MUST be a positive integer and MUST be equal or greater than the timestamp used in previous requests."
- Use Redis GET/SET
 - request_timestamp:consumer_key:<123> =<timestamp>

Server-Side: Nonce

- "...a Nonce value that is unique for all requests with that timestamp"
- Use Redis SADD (a set with fixed duration)

Rack

- A protocol for an interface
 - Between Ruby web servers and applications
- https://github.com/rack/rack
 - There is code...but that's not important right now.

Rack.call(env)

- Is an object that responds to call(env)
- Must return array of:
 - Status
 - Hash of HTTP headers
 - Object that responds to #each the response body

Rack Middleware

- Response to call(env) like a Rack App
- Has an initialize(app) method that takes a reference to the downstream application

A Middleware For Every Auth

- 1. Check Authorization Token
- 2. Check timestamp
- 3. Check nonce
- 4. Check signature
- 5. Check user:
 - Check username/password (register and login) OR
 - Check consumer key

DEMO

- 1. Rack App
- 2. Rack Middleware
- 3. Auth in Middleware
- (No routing)

Warden

- Proxy injected into request
 - env['warden']
- Available to downstream middleware/apps
- Strategies
- Is it worth it?
 - Flexible, structure vs. early response

Conclusion

- Rack is widely used in Ruby
- Rack is an integral part of Rails (and Sinatra)
 - http://guides.rubyonrails.org/rails_on_rack.html
- Devise is built on Warden (built on Rack)
 - Warden is middleware
- Mobiles complicate web development
 - But also simplify!
- Know your technologies, know your protocols!