# Website Penetration Testing Lab

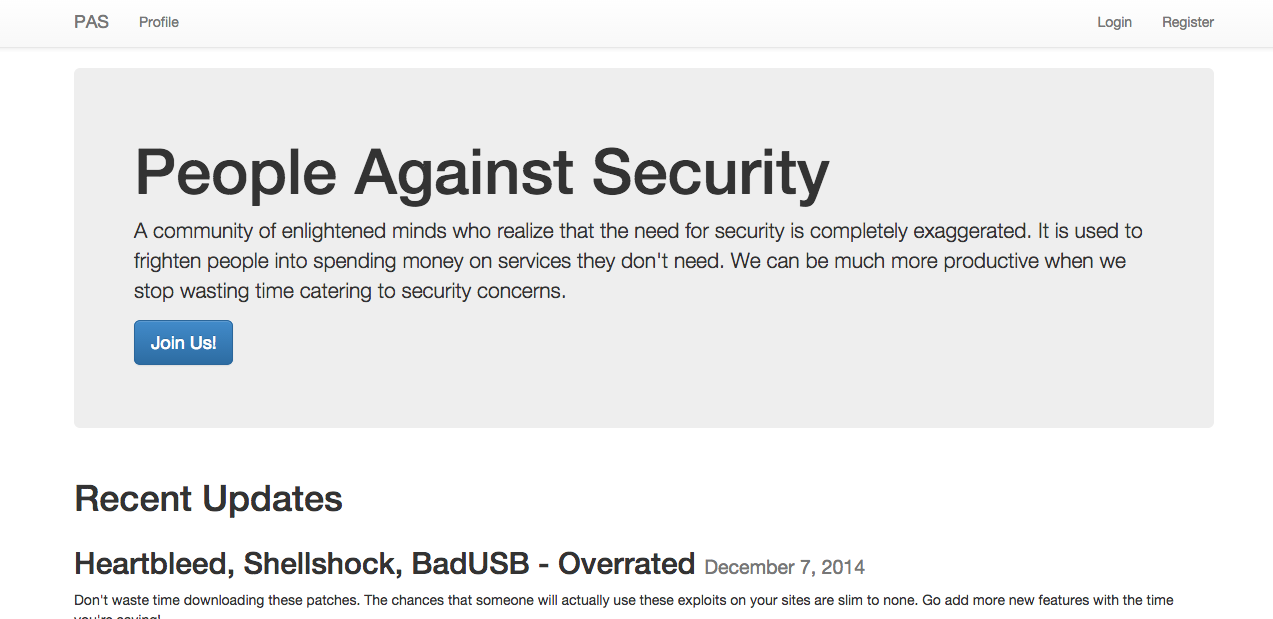
Ryan Robeson

## Objective

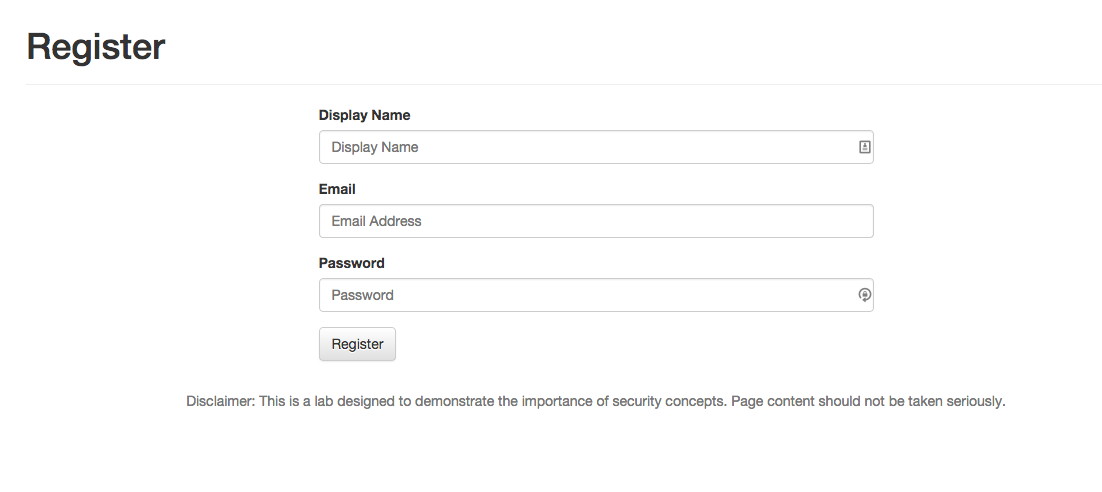
Gain an understanding of how multiple small exploits can lead to huge problems for web developers.

## Results

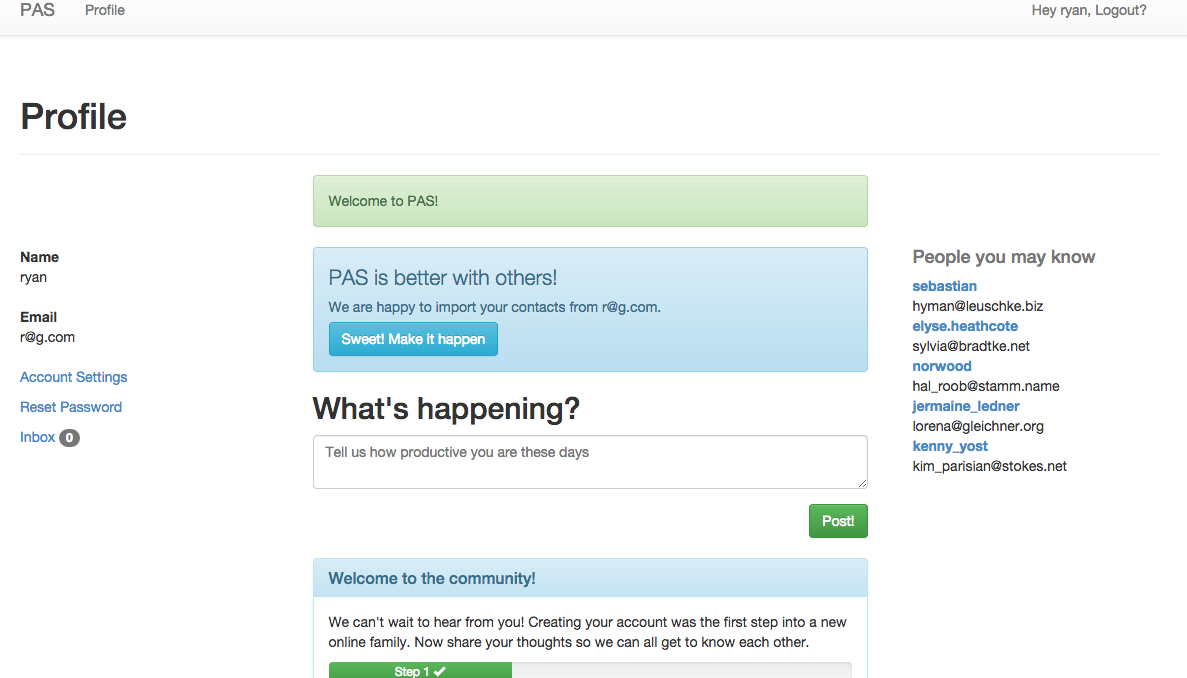
### Got PAS running



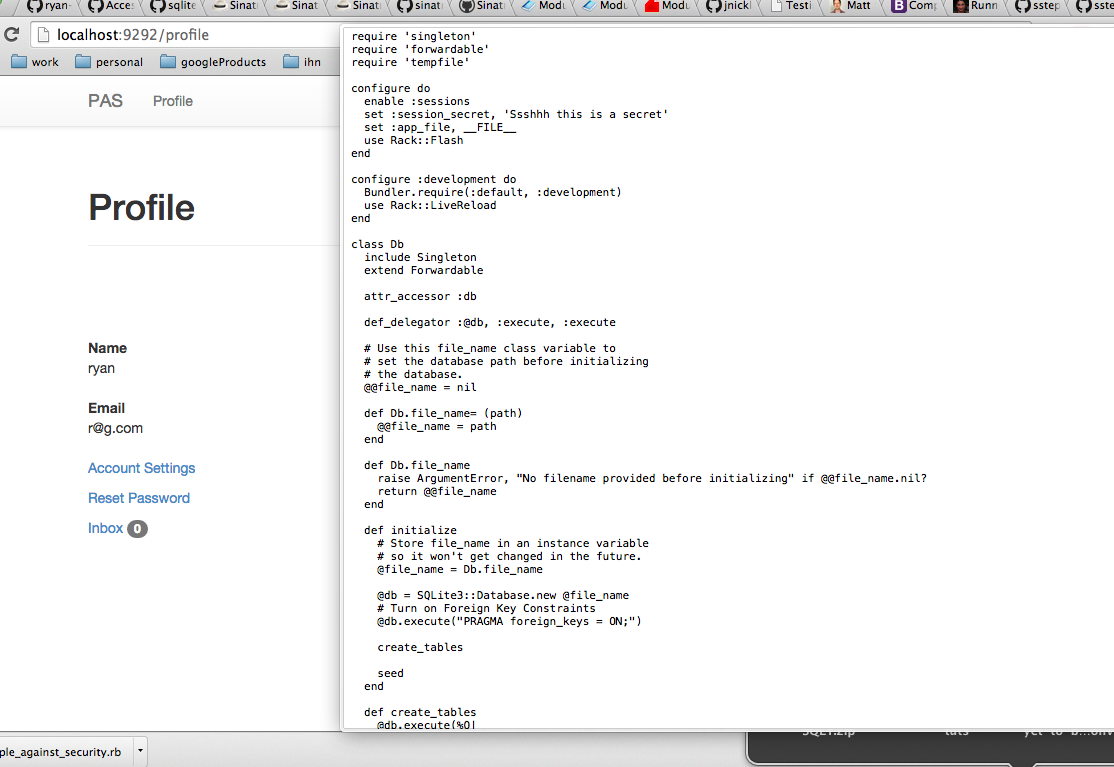
### Registering



### Viewing Profile

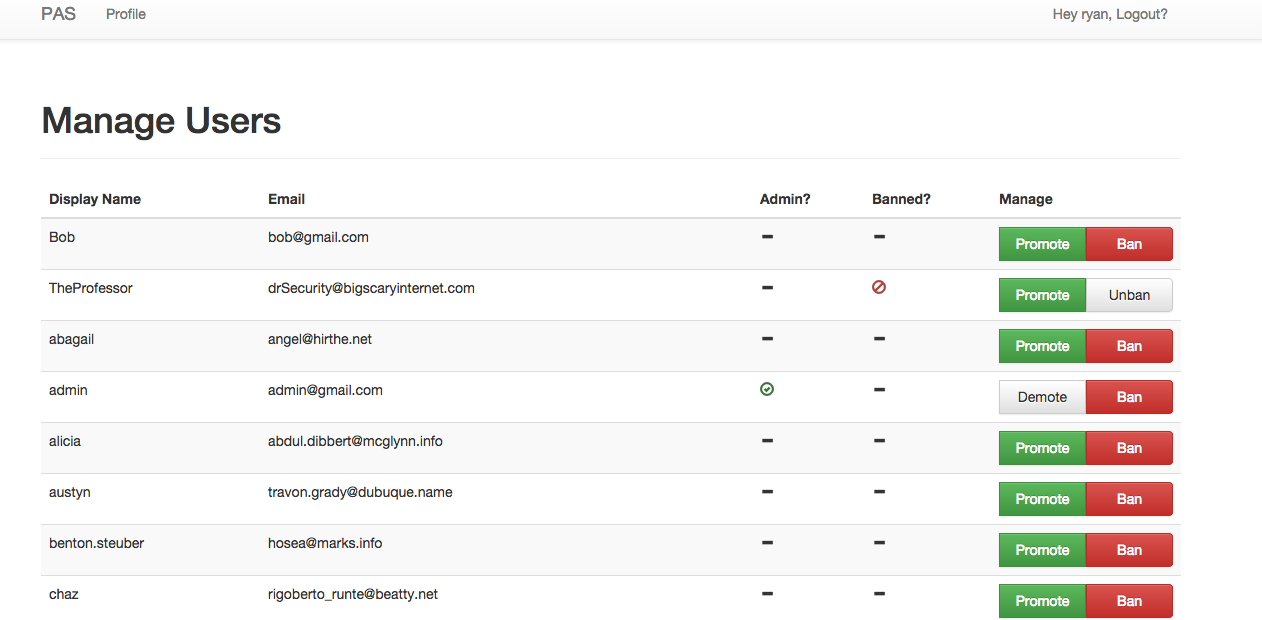


### Found the Source



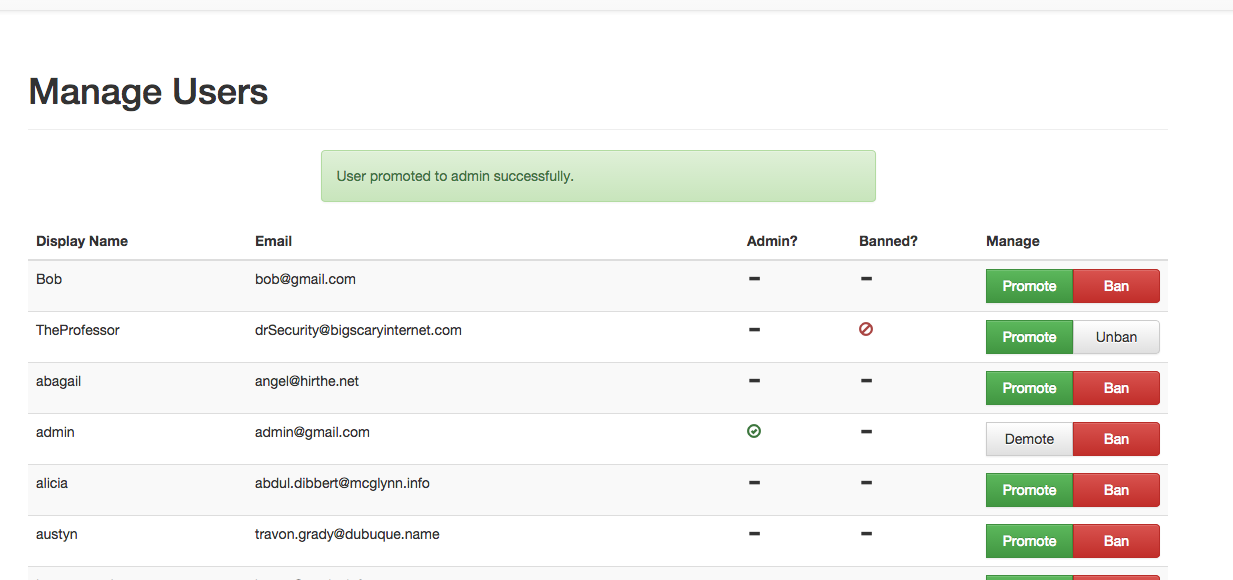
I found out that everything after /download gets mapped to the file system with no validation. When I tried /download/index.php, I was able to obtain the source of the website.

### Manage Users



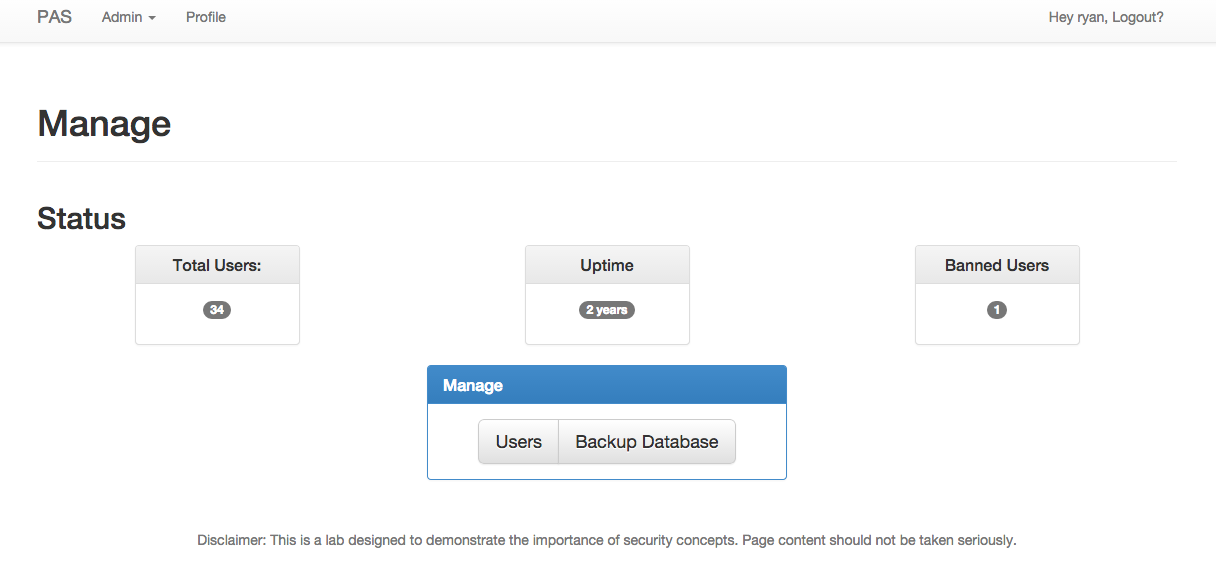
Looking through the source, I discovered that /admin/manage/users is only authenticated, not checked for authorization. Even though /admin/manage was correctly protected, I was able to reach the manage users page as a normal user.

### Escalated Privileges



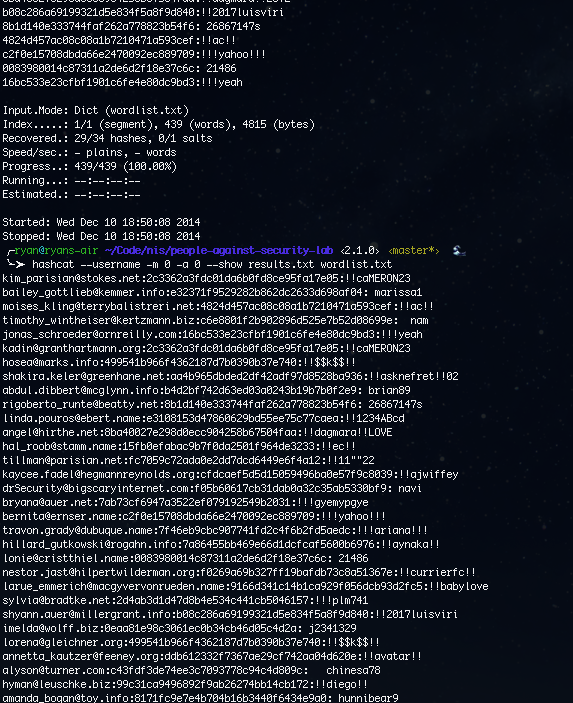
There was no further checking to stop me from promoting my account to an administrator.

### Obtained the Database



The /admin/manage page has a convenient "Backup Database" link on it. As I was now an administrator, I had access to this page and there were no further checks to prevent me from downloading the entire database.

### Cracked the Hashes



The database had no access controls to speak of. It was completely unencrypted. I used the sqlite3 command to interact with it. There was only one table, users. This table contained email addresses and password hashes for every user on the site. To determine the type of hash being used, I simply looked back at the source code and saw that they were using standard MD5 without a salt. I used sqlite3 again to retrieve the emails and passwords into a text file that I could give to hashcat. I then ran hashcat with this hash file and the provided word list and was able to recover all but a few hashes.

## Reflection

It's amazing how something as small as a badly written download script can lead to a stolen database. Without that script, I never would have found the "Manage Users" URL. I couldn't have promoted myself to an administrator and I couldn't have obtained the database. Even without the download vulnerability, they should have tested that authorization was being applied on the controller rather than simply administrative links. They also should have had checks in place to prevent a user from promoting their own account. However, that could have been circumvented by simply creating another account to promote had they not locked down their user management page.

The database backup should not have been accessible to every admin. Having all of that customer data available so easily is inexcusable. If they were going to offer this functionality, it should have been limited to specific accounts and encrypted with a password or key not related to the website.

Finally, using MD5 for password hashing is barely any better than storing passwords in plaintext with today's hardware. They should have migrated to salted password hashing with one of the SHA variants years ago, and should now be moving to bcrypt or PBKDF2. These algorithms are much more secure against brute force. Web applications being so publically accessible have a huge attack surface and security must be taken seriously.