BASICS OF COMPUTER SECURITY

RULE #1 OF CYBERSECURITY

IF YOU ARE NOT AN EXPERT, DO NOT IMPLEMENT YOUR OWN SECURITY PROGRAMS (EXCEPT FOR FUN OR LEARNING).

These things are incredibly hard to get right. People who have been writing them for 30+ years still occasionally make subtle mistakes.

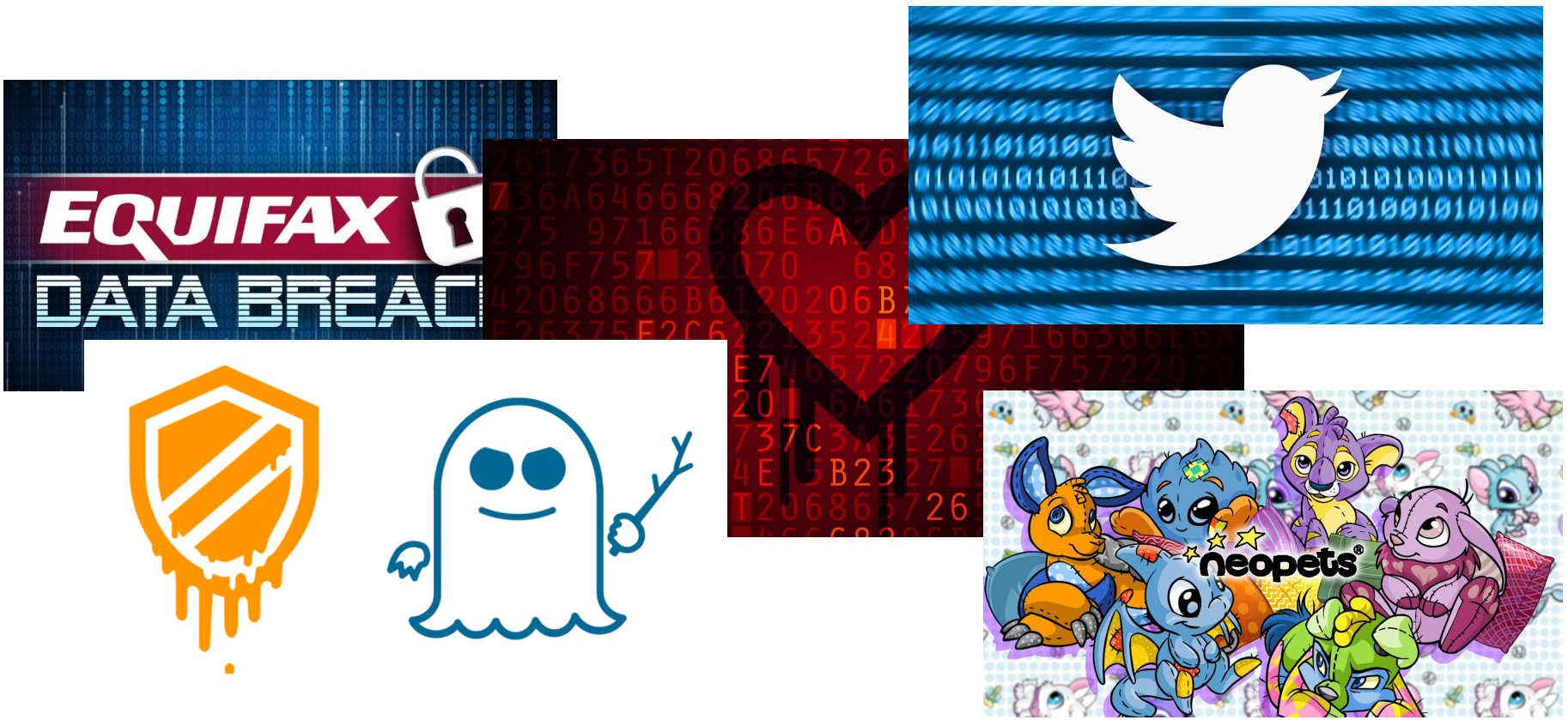
IN THIS CLASS, WE'VE BEEN ASSUMING THAT OUR USERS MIGHT BE CONFUSED OR MAKE MISTAKES, BUT NOT THAT THEY'RE ACTIVELY TRYING TO BREAK OUR PROGRAMS

In the real world this might not be accurate!

If we're writing software for a bank, someone might be able to break open people's accounts and siphon off millions of dollars if the software is wrong.

THAT'S A BIG INCENTIVE!

So how do we keep our software and users secure while everyone out there is trying to break down our doors?



WE DON'T REALLY DO A GOOD JOB OF IT.

There's simply too much in this field for me to cover. Instead, I'll cover a few common problems and the solutions we've come up with for them.

WHAT'S THE SECRET WORD?

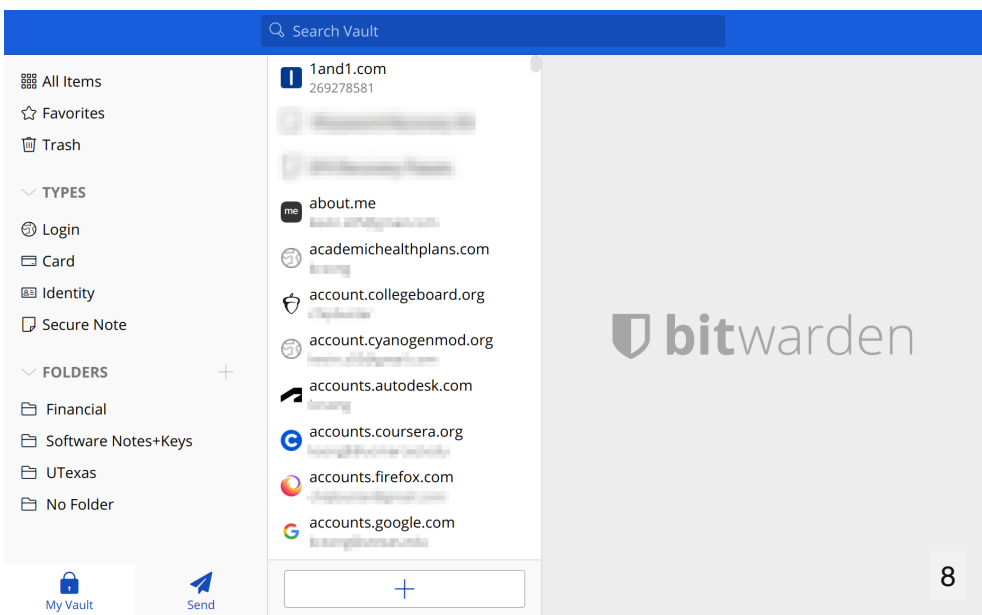
Passwords have been around in computers for as long as we needed to keep computers secure.

In the 1980s, you usually had a single computer for everyone at the company. If their password was lost, someone would still have to get into the building (using stolen keys, etc.) to use it.

And what value were they going to get out of it anyways?

Today, most people have hundreds of logins, all accessible over the internet.

The rules of the password game have changed!



INTERNET ACCESSIBLE PASSWORDS

The service provider having access to the actual password of a user is a *terrible* idea.

> *Someone broke into my bank account and transferred all my money into an offshore account. The only other website that I used that password on was your website, therefore, one of your employees must be responsible.*

eek

More generally, if someone breaks into our server, we don't want them to have the passwords for all our users!

But we need to check the user's password when they log in.

SO....WE CAN'T STORE THE SECRET. BUT WE NEED TO CHECK THAT THE USER KNOWS THE SECRET!

SOLUTION: PASSWORD HASHES

Specially designed one-way functions: if you know the password, it's easy to get the hash, but if you only know the hash, it's very, very hard to figure out the password.

Input	**Hash Value**
thisismypassword	584428bdac29487f8a8066f5f162e19cb7ae268f44f7c240043b75b39d83c7e7
echidna	55901cd27840ca340a22bea64ca389e1434ff537e124896fe0199fddf8936ebe
kangaro	d5433b61c55907787740f91d43c18235fde94ffbe37ca048a6008eff45f201bc
kangaroo	01dd2561496389a4532598373c3c51191c1dabdede5335be055682c1259bd457

When a user creates an account with us, we take their password and hash it. We then store the hash, but do not store the password.

When the user tries to log in, we once again take their password and hash it, and compare. If the hashes match, the user must have the right password.

If someone breaks into our server, all they get are the hashes, which are impossible to invert!

STOPWATCHES AND SECRET STEALING

HOW DO I WRITE A PROGRAM TO CHECK THAT TWO STRINGS ARE EQUAL?

```
1 def string_equals(str1, str2):
2   if len(str1) != len(str2):
3     return False
4   for i in range(len(str1)):
5     if str1[i] != str2[i]:
6       return False
7   return True
```

Congrats, you've created a security vulnerability!

```
1 def string_equals(str1, str2):
2   if len(str1) != len(str2):
3     return False
4   for i in range(len(str1)):
5     if str1[i] != str2[i]:
6       return False
7   return True
```

Suppose I pass str1 = "aaaaaaaaaaaaaaaaa" and str2 = "aaaaaaaaaaaaaaaab". How many times does the loop run?

Suppose I pass str1 = "aaaaaaaaaaaaaaaaa" and str2 = "bbbbbbbbbbbbbbb". How many times does the loop run?

TIMING ATTACK!

By timing how long the comparison function takes to complete, we can tell the first non-matching character.

With a good enough timer, we can figure out how many of the first characters we have correct!

VULNERABLE

```
1 def string_equals(str1, str2):
2   if len(str1) != len(str2):
3     return False
4   for i in range(len(str1)):
5     if str1[i] != str2[i]:
6       return False
7   return True
```

```
1 def string_equals(str1, str2):
2   is_equal = True
3   for i in range(len(str1)):
4     if str1[i] != str2[i]:
5       is_equal = False
6   return is_equal
```

SECURE (?)

TALKING IN THE CLEAR

Once upon a time, the websites of the internet lived in harmony...

JUST KIDDING, IT WAS FIRESHEEP.

Website Authentication Flow:

- User connects to website
- User enters password, website takes hash
- Website compares hash to known password hash
- If hashes match, website gives user a timed-life *cookie* which can be presented to prove that this is the logged-in user.
- Think of it like tickets and wristbands.

UP UNTIL 2010, THESE COOKIES WERE USUALLY TRANSMITTED IN CLEARTEXT!

Firesheep was an extension that read these authentication cookies out of the browser and used them to "hijack" your logins.

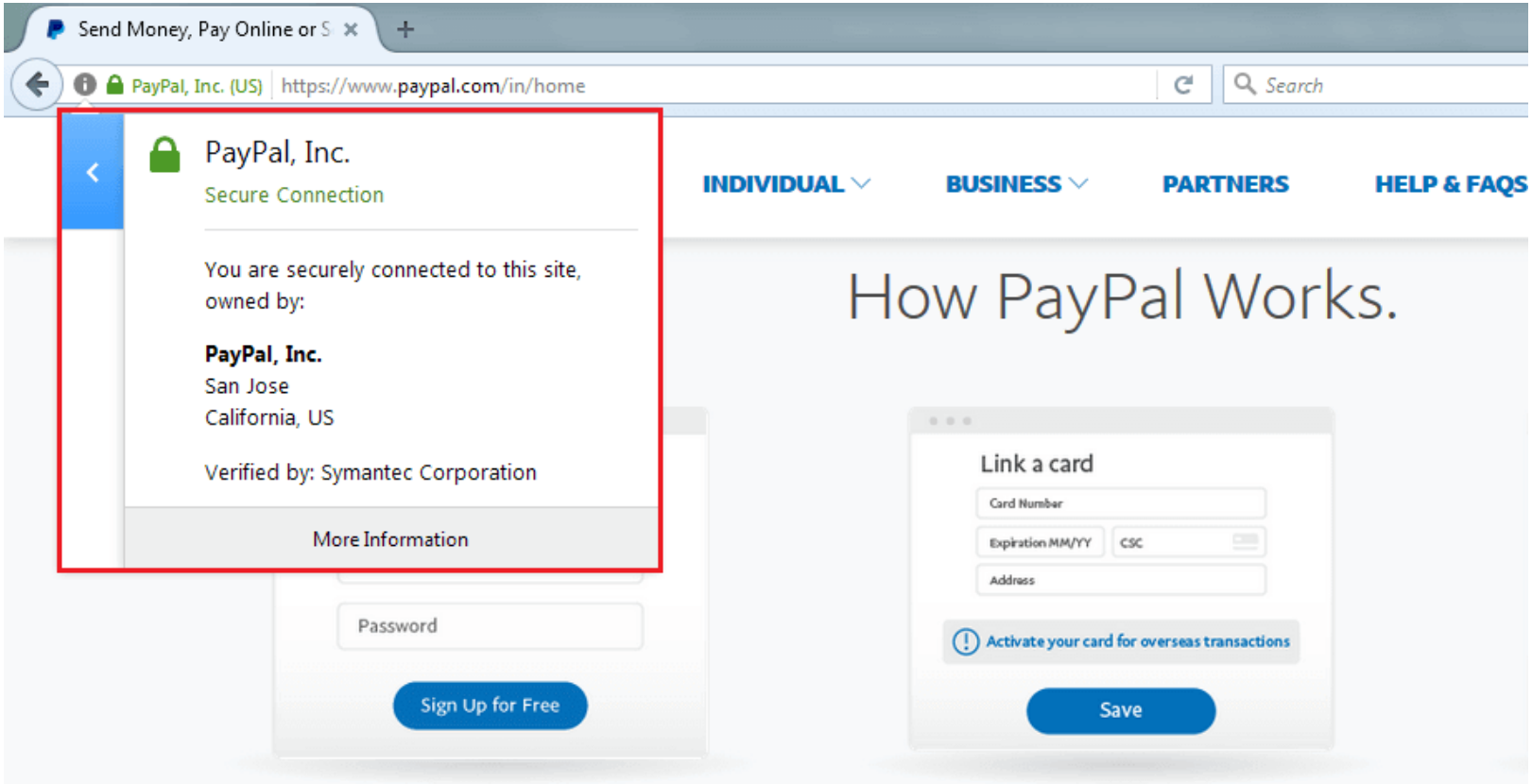
About Issues Our Work Take Action Tools Donate

The Message of Firesheep: "Baaaad Websites, Implement Sitewide HTTPS Now!"

COMMENTARY BY SETH SCHOEN | OCTOBER 29, 2010

Co-authored by Richard Esguerra

Pushed most major websites into using encrypted connections.



Firesheep is the main reason you see these at all today!

WHAT DO WE TAKE AWAY FROM THIS?

The internet was designed on all users trusting each other.

In today's world, that is no longer the case. New protocols need to be designed with security in mind.

OTHER QUESTIONS