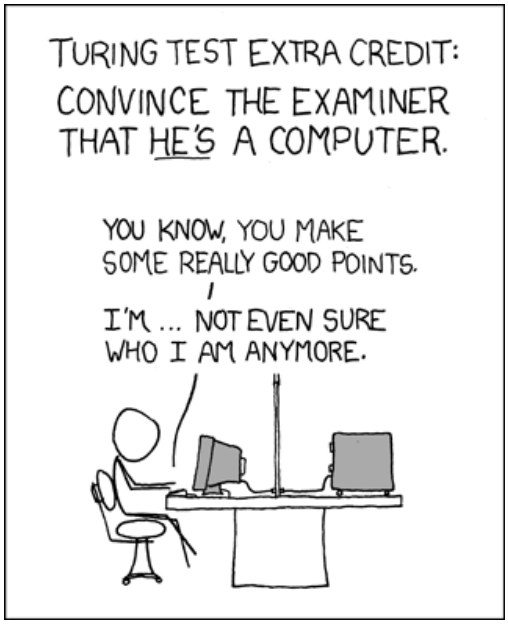
Name: _____ UNI: _____

Midterm Exam: February 2023
COMS W4182: Computer Security II

Rules

- Remember to write your name and UNI on the exam/scoresheet paper
- Also write your name and UNI on your blue book
- You must turn in *both* the exam sheet and the blue book
- Books and notes are allowed during this examination.
- Consultation or collaboration with anyone, in the class or not, is **strictly prohibited**.
- This is a timed test. All papers must be turned in 75 minutes after the beginning of the test.
- Most questions can be answered in just a paragraph or two; if you think you need to write several pages, you're writing too much and may be on the wrong track entirely. If a question is worth only a very few points, that's a pretty good clue that the answer is pretty simple.
- The total points add up to 55.
- Good luck, and may the Force be with you.

Question	Points	Score
1	15	
2	15	
3	10	
4	15	
Total:	55	



https://xkcd.com/329/

1. (15 points) Excluding the database lookup to find the proper user's hashed password entry, the code for password authentication is pretty simple: hash the supplied password and compare it with the database entry. By comparison, other forms of authentication are considerably more complex, though TOTP is close in simplicity. (Recall that for TOTP, the site calculates $H_k(T)$, where T is the time of day and H_k is a keyed hash function using a per-user key k. It then compares that value to what the user has entered.)

   Assertion: for lower-value sites, the code complexity of, e.g., FIDO2 creates more security problems than passwords create. Consequently, these schemes should not be used. True or false? Explain.

   Answer:

   Either true or false is acceptable, depending on the reasoning. Note that the question is not about how good FIDO2 is or about how much it costs. Rather, it's about code complexity.

   True As noted, complexity is the enemy of security. While you don't want your site to be penetrated, that isn't what's at risk from compromised passwords; rather, it's user accounts.

   Also: most people don't care about 2FA. Twitter said that $< 3\%$ of its users adopted such a scheme.

   False This is probably a better answer.

   First: people reuse passwords. By not having passwords on your site, you help the net as a whole.

   Second: with the advent of PassKey, logging in without a password will be simpler for users.

   Third: even for low value sites, people get annoyed when they can't log in because their password has been stolen and perhaps changed by the attacker.

   It isn't valid to say that you won't have to implement a password recovery mechansim: you will have to implement credential recovery for some other scheme.

2. (15 points) IoT devices generally have operating systems to handle things like file systems, input/output, process scheduling, etc. Things may use a specialized real-time operating system like VxWorks, or a variant of a general-purpose OS such as Windows or Linux. It is therefore conceivable that there could be an IoT worm, spreading from your thermostat to your garage door opener to your car, and your car can in turn spread it to other folks' door locks or what have you.

   What should vendors do to prevent such problems?

   Answer:

   There are thus two ways IoT worms are likely to spread: on-LAN and via the Vendor. The same reasons why it's hard for an authorized user to talk directly to an IoT device mean that a worm won't be able to spread easily between households except via Vendors.

   Trying to put antivirus software on IoT devices is probably futile: there's no eonomic model, there are no keyboards or displays, etc. However, that does not apply to the Vendor. The Vendor could use some sort of intrusion detection to notice odd things coming from the Things it manages.

   Hubs can do some of this, but what is their update model? If the Vendor system goes away, the device is useless, so we can assume that the Vendor system will always be there.

Blocking attempts by Things to communicate on-LAN except to the Hub is a good idea. Vendors cannot enforce network separation; that's an installation issue.

Passwords often have little to do with the question, as the owner has no way to set one.

Bonus: all of this implies that it's hard for an infected car to directly attack someone else's door locks. That may not be true if the car owner has the WiFi password to that person, e.g., perhaps a close relative or friend. However... A lot of Apple gear talks peer-to-peer over WiFi at layer 2, rather than using the standard access point protocol. If the car and the door lock do the same, an attack might be feasible. There are no good defenses here, except again, at the Vendor level.

3. (10 points) One problem with people using certificates to authenticate themselves is the difficulty of securely moving the corresponding private keys around to many different devices. Someone suggests converting an iris scan to a private key. You'll thus always have it with you and don't have to worry about copying it. Is this a good idea? Why or why not?

Answer:

This is a bad idea. First and foremost (and as discussed in class), it isn't easy to convert a biometric into a key. Second, irises are exposed to public view; a high-resolution photograph of a face will likely capture it. Third, not all devices have suitable cameras. (As I type this, I'm on Zoom office hours. The camera on my monitor doesn't seem good enough to pick up details of my iris, which protects me from others, but that also means it can't read my iris to generate my private key when I want to use it.)

Also: you can't change an iris scan, and hence can revoke the certificate. Not all devices have scanners, so the solution isn't universal across devices. And some people have eye injuries, etc.

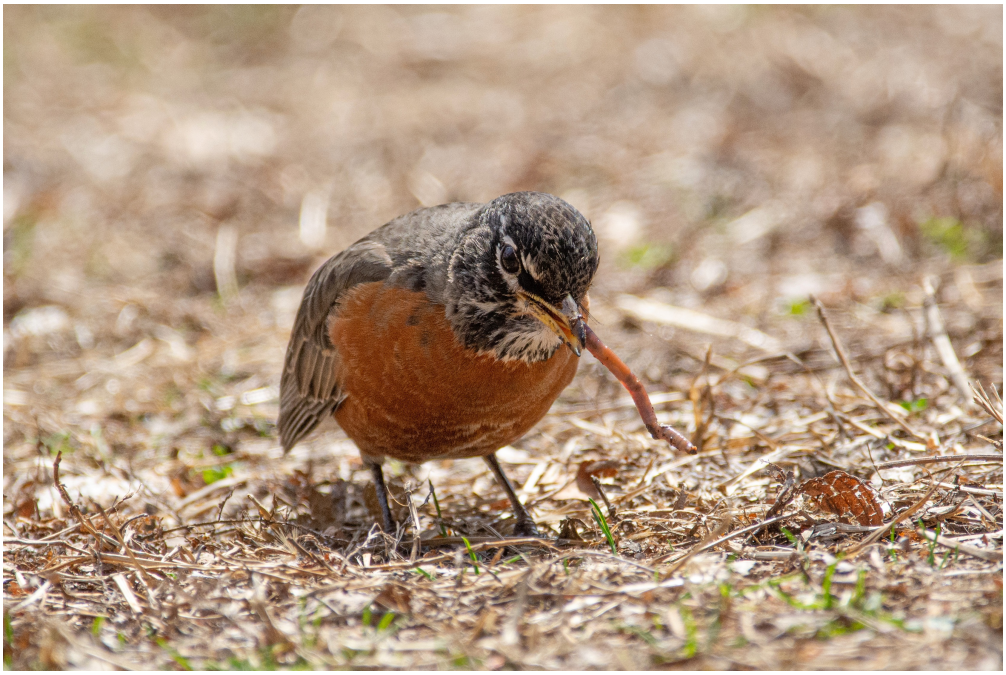
Bit replay isn't at issue here—we're not talking about sending the iris scan somewhere, we're talking about using it as a private key, which is never transmitted at all; rather, it's used for local computation. The same answer applies to remote biometrics—it's not one; it's local.

4. (15 points) Conceptually, FIDO2 can be implemented in several places. One, of course, is in an outboard security key; another is in a phone app. For these two options—and I'm specifically excluding other possibile implementations—explain some security advantages and disadvantages of each choice.

Answer:

Outboard key Outboard security keys have a strong separation from the general host environment. In theory, this should give them a lower attack surface, though there is great potential for weaknesses in the USB or network protocol interface. Also, they can be lost or stolen, or simply left home. Finally, by their nature they can't be backed up; users need to have more than one to continue access in event of loss or failure.

Phone app People more or less always have their phones handy and miss them quickly if they've left them someplace. There is no incremental hardware cost, and phones typically have more or less any kind of interface a host might have. However, the attack surface of a phone is fairly large. Furthermore, although many phones have secure storage and/or a built-in HSM equivalent, e.g., Apple's Secure Enclave, not all do. Apps don't always use these correctly.



Not a good anti-worm defense.