3

IDENTIFICATION AND AUTHENTICATION

When you're developing security measures, whether they're specific mechanisms or entire infrastructures, identification and authentication are key concepts. In short, *identification* makes a claim about who or what someone or something is, and *authentication* establishes whether this claim is true. You can see these processes taking place daily in a wide variety of ways.

One common example of an identification and authentication transaction is the use of payment cards that require a personal identification number (PIN). When you swipe the magnetic strip on the card, you are stating that you are the person indicated on that card. At this point, you've given your identification but nothing more. When you're asked to enter the PIN associated with the card, you complete the authentication portion of the transaction, proving that you are the legitimate cardholder.

Some of the identification and authentication methods that we use daily are particularly fragile, meaning they depend largely on the honesty and diligence of those involved in the transaction. If you show your ID card to buy alcohol, for example, you're asking people to trust that your ID is genuine

and accurate; they can't authenticate it unless they have access to the system that maintains the ID in question. We also depend on the competence of the person or system performing the authentication; they must be capable not only of performing the act of authentication but also of detecting false or fraudulent activity.

You can use several methods for identification and authentication, from requiring simple usernames and passwords to implementing purpose-built hardware tokens that serve to establish your identity in multiple ways. In this chapter, I'll discuss several of these methods and explore their uses.

Identification

Identification, as you just learned, is simply an assertion of who we are. This may include who we claim to be as people, who a system claims to be over the network, or who the originating party of an email claims to be. Let's look at some methods for determining identity and examine their trustworthiness.

Who We Claim to Be

Who we claim to be is a tenuous concept in the best case. We can identify ourselves by our full names, shortened versions of our names, nicknames, account numbers, usernames, ID cards, fingerprints, or DNA samples. Unfortunately, with a few exceptions, such methods of identification are not unique, and even some of the supposedly unique methods of identification, such as fingerprints, can be duplicated.

Who we claim to be can, in many cases, be subject to change. For example, many women change their last names when they get married. In addition, we can generally easily change logical forms of identification, such as account numbers or usernames. Even physical identifiers, such as height, weight, skin color, and eye color, can change. One of the most crucial factors to realize is that a claim of identity alone is not enough.

Identity Verification

Identity verification is a step beyond identification, but it's still a step short of authentication, which I'll discuss in the next section. When you are asked to show a driver's license, Social Security card, birth certificate, or other similar form of identification, this is generally for verification of identity, not authentication. This process is the rough equivalent of someone claiming the identity John Smith, and you asking whether the person is indeed John Smith and being satisfied with an answer of "Sure, I am" from the person (plus a little paperwork).

We can take the example a bit further and validate the form of identification (say, a passport) against a database holding an additional copy of the information that it contains, matching the photograph and physical specifications with the person standing in front of us. This may get us a bit closer

to ensuring that we have correctly identified the person, but it still does not qualify as authentication. We may have validated the status of the ID itself, and we know that the person meets the general specifications of the person to whom it was originally issued, but we have taken no steps to prove that the person is really the right one. The more we trend toward verification and away from authentication, the weaker our controls are.

Computer systems also use identity verification. When you send an email, the identity you provide is considered true; the system rarely takes any additional steps to authenticate you. Such security gaps contribute to the huge amount of spam traffic, which Cisco's Talos Intelligence Group estimates to have accounted for more than 90 percent of all email sent in 2024.[13]

Identity Falsification

As I've discussed, methods of identification are subject to change. As such, they are also subject to falsification. Minors often use fake IDs to get into bars or nightclubs, while criminals and terrorists might use them for more-nefarious tasks. You could use some methods of identification, such as birth certificates, to obtain additional forms of identification, such as Social Security cards or driver's licenses, thus strengthening a false identity.

Identity theft based on falsified information is a major concern today; nearly 300,000 identity thefts per quarter were reported by US consumers in 2024.[14] Unfortunately, this type of attack is common and easy to execute. Given a minimal amount of information (usually, a name, address, and Social Security number are sufficient), it is possible to impersonate someone just enough to be able to conduct a variety of transactions in their name, such as opening a line of credit. Such crimes occur because many activities lack authentication requirements. Although most people think that identity verification is sufficient, verification is easy to bypass by using falsified forms of identification.

Many of the same difficulties exist in computer systems and environments. For example, sending an email from a falsified email address is entirely possible. Spammers use this tactic on a regular basis. I'll address such issues at greater length in Chapter 16.

Authentication

In cybersecurity, authentication is the set of methods used to establish whether a claim of identity is true. Note that authentication does not decide what the party being authenticated is permitted to do; this is a separate task, known as authorization. I'll discuss authorization in Chapter 4.

Factors

Authentication is based on several *factors*: something you know, something you are, something you have, something you do, and where you are. When

you're attempting to authenticate a claim of identity, you'll want to use as many factors as possible. The more factors you use, the more positive your results will be.

Something you know, a common authentication factor, includes passwords or PINs. However, this factor is somewhat weak because if the information on which the factor depends is exposed, your authentication method may no longer be unique.

Something you are is a factor based on the relatively unique physical characteristics of an individual, often referred to as *biometrics*. Although biometrics can include simple attributes such as height, weight, hair color, or eye color, these aren't usually distinctive enough to make very secure identifiers. Complex identifiers such as fingerprints, patterns in the iris or retina, or facial characteristics are more common. These are a bit stronger than, say, a password, because forging or stealing a copy of a physical identifier is somewhat more difficult, although not impossible. There is some question as to whether biometrics truly count as an authentication factor or whether they constitute only verification. I'll discuss this in greater depth in "Biometrics" on page 35.

Something you have is a factor generally based on physical possession, although it can extend to some logical concepts. Common examples are automatic teller machine (ATM) cards, state or federally issued identity cards, or software-based security tokens, as shown in Figure 3-1. Some institutions, such as banks, have begun to use access to logical devices, such as cell phones or email accounts, as authentication methods as well.

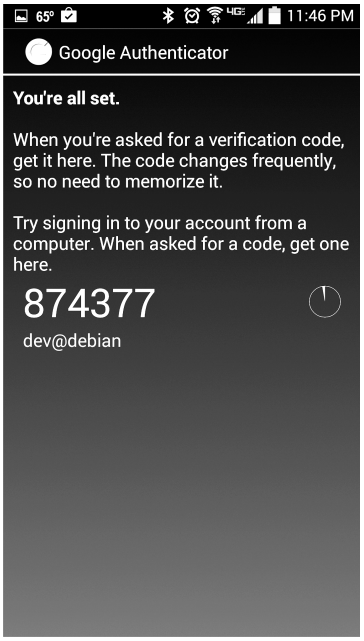

Figure 3-1: Sending a security token to a mobile phone is a common authentication method.

This factor can vary in strength depending on the implementation. If you wanted to use a security token sent to a device that doesn't belong to you, you'd need to steal the device to falsify the authentication method or otherwise compromise the mechanism being used. On the other hand, a security token sent to an email address would be much easier to intercept, and you'd have a measure of considerably less strength.

Something you do, sometimes considered a variation of something you are, is a factor based on the actions or behaviors of an individual. This may include an analysis of the individual's gait or handwriting or the time delay between keystrokes while typing a passphrase. These factors present a strong authentication method and are difficult to falsify. They do, however, have the potential to incorrectly reject legitimate users at a higher rate than some of the other factors.

Where you are is a geographically based authentication factor. This factor operates differently than the others, as it requires a person to be present at a specific location. For example, when changing an ATM PIN, some banks require you to go to a branch, where you will then be required to present your ID and account number. If the bank allows the PIN to be reset online or over the phone, an attacker could change your PIN remotely and proceed to clean out your account. Although potentially less useful than some of the other factors, this factor is difficult to counter without completely subverting the system performing the authentication.

Multifactor Authentication

Multifactor authentication (MFA) uses one or more of the factors discussed in the preceding section. When you're using only two factors, this practice is also sometimes called *two-factor authentication (2FA)*.

Let's return to the ATM example because it illustrates MFA well. In this case, you use something you know (your PIN) and something you have (your ATM card). Your ATM card serves as both an authentication factor and a form of identification. Another example of MFA is writing checks. In this case, you're using something you have (the checks themselves) and something you do (signing them). Here, the two factors involved in writing a check are rather weak, so sometimes you will see a third factor, a fingerprint, used with them.

Depending on the factors selected, you can build stronger or weaker MFA schemes particular to each situation. In some cases, certain methods may be more difficult to defeat but impractical to implement. For example, DNA makes for a strong authentication method but is not practical in most situations. In Chapter 1, I said that your security should be proportional to what you're protecting. You could certainly install iris scanners on every credit card terminal, but this would be expensive, impractical, and potentially upsetting to customers.

Mutual Authentication

In *mutual authentication*, both parties in a transaction authenticate each other. These parties are typically software based. In the standard one-way authentication process, the client authenticates to the server. In mutual authentication, not only does the client authenticate to the server, but the server authenticates to the client as well. Mutual authentication often relies on digital certificates, which I'll discuss in Chapter 6. Briefly, both the client and the server would have a certificate to authenticate the other.

When you don't perform mutual authentication, you leave yourself open to impersonation attacks, often referred to as *man-in-the-middle attacks*. In this attack, the attacker, located between the client and the server, impersonates the server to the client, and the client to the server, as shown in Figure 3-2. This is accomplished by intercepting and forwarding the traffic that would usually flow directly between the client and the server.

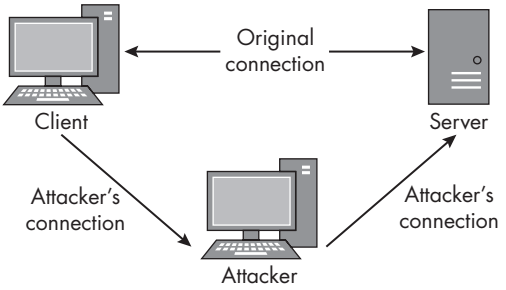

Figure 3-2: A man-in-the-middle attack

This circumvention is typically possible because the attacker needs to subvert or falsify authentication only from the client to the server. If you implement mutual authentication, this attack becomes considerably more difficult because the attacker would have to falsify two authentications.

You can also combine mutual authentication with MFA, although the latter generally takes place only on the client side. MFA from the server back to the client would be not only technically challenging but also impractical in most environments because it would require some technical heavy lifting on the client side, potentially on the part of the user. You'd likely lose a significant amount of productivity.

Common Identification and Authentication Methods

I'll conclude this discussion by exploring three common identification and authentication methods in detail: passwords, biometrics, and hardware tokens.

Passwords

Passwords are familiar to most of us who use computers regularly. When combined with a username, a password will generally allow you access to a computer system, an application, a phone, or a similar device. Although they're only a single factor of authentication, passwords can represent a relatively high level of security when constructed and implemented properly.

People often describe certain passwords as being *strong*, but a better descriptive term might be *complex*. If you construct a password that uses lowercase letters only and is eight characters long, you can use a password-cracking utility to quickly reveal it, as discussed in Chapter 1. Adding character sets to the password makes it increasingly difficult to figure out. If you use uppercase letters, lowercase letters, numbers, and symbols, you'll end up with a password that is potentially more difficult to remember, such as *$sU&qw!3*, but much harder to crack.

In addition to constructing strong passwords, you need to practice good password hygiene. Don't reuse or share your password. Don't write it down and post it on your monitor; doing so completely defeats the purpose of having a password in the first place. Applications called *password managers* can help us manage all our logins and passwords for multiple accounts, some as locally installed software, and others as web or mobile device applications. Many arguments can be made for and against such tools; some people think keeping all your passwords in one place is risky, but when used carefully, these managers can help you maintain good password hygiene.

Another common problem is the manual synchronization of passwords—in short, using the same password everywhere. If you use the same password for your email, for your login at work, and for your online knitting discussion forum, you are putting the security of all accounts in the hands of those system owners. If any one of them is compromised, all your accounts become vulnerable; all an attacker needs to do to access the others is look up your account name on the internet to find your other accounts and log in using your default password. By the time the attacker gets into your email account, the game is over because an attacker can generally use it to reset account credentials for any other accounts you have.

Biometrics

Although some biometric identifiers may be more difficult to falsify than others, this is only because of the limitations of today's technology. At some point in the future, we'll need to develop more-robust biometric characteristics to measure or stop using biometrics as an authentication mechanism.

Use of Biometrics

Biometrics-equipped devices are becoming increasingly common and inexpensive. You can find a wide selection for less than $20. It pays to research such devices carefully before you depend on them for security, as some of the cheaper versions are easy to bypass.

You can use biometric systems in two ways. You can use them to verify the identity claim someone has made, as discussed earlier, or you can reverse the process and use biometrics as a method of identification. This process is commonly used by law enforcement agencies to identify the owner of fingerprints left on various objects. This can be a time-consuming effort considering the sheer size of the fingerprint libraries these organizations have.

To use a biometric system in either manner, you need to put the user through some sort of enrollment process. Enrollment records the user's chosen biometric characteristic (for instance, making a copy of a fingerprint) and saves it in a system. Processing the characteristic may also include noting elements that appear in certain parts of the image, known as *minutiae* (Figure 3-3).

Figure 3-3: Biometric minutiae

You can use the minutiae later to match the characteristic with the user.

Characteristics of Biometric Factors

Biometric factors are defined by seven characteristics: universality, uniqueness, permanence, collectability, performance, acceptability, and circumvention.[15]

Universality means you should be able to find your chosen biometric characteristic in the majority of people expected to enroll in the system. For instance, although you might be able to use a scar as an identifier, you can't guarantee that everyone will have a scar. Even if you choose a common characteristic such as a fingerprint, you should take into account that some people may not have an index finger on their right hand and be prepared to compensate for this.

Uniqueness is a measure of the distinctiveness of a characteristic among individuals. For example, if you choose to use height or weight

as a biometric identifier, you'd stand a good chance of finding several people in any given group who have the same height or weight. You should try to select characteristics with a high degree of uniqueness, such as DNA or iris patterns, but even these could be duplicated, whether intentionally or otherwise. For example, identical twins have the same DNA, and an attacker could replicate a fingerprint.

Permanence tests how well a characteristic resists change over time and with advancing age. If you choose a factor that can easily vary, such as height, weight, or hand geometry, you'll eventually find yourself unable to authenticate a legitimate user. It's better to use factors such as fingerprints, which are unlikely to change without deliberate action.

Collectability measures how easy it is to acquire a characteristic. Most commonly used biometrics, such as fingerprints, are relatively easy to acquire, which is one reason they are common. On the other hand, a DNA sample is more difficult to acquire because the user must provide a genetic sample to enroll and to authenticate again later.

Performance measures how well a given system functions based on factors such as speed, accuracy, and error rate. I'll discuss the performance of biometric systems at greater length later in this section.

Acceptability is a measure of the level of acceptance of the characteristic by the users of the system. In general, systems that are slow, difficult to use, or awkward to use are less likely to be acceptable to the user.[16] Systems that require users to remove their clothing, touch devices that have been repeatedly used by others, or provide tissue or bodily fluids are unlikely to have a high degree of acceptability.

Circumvention describes how easy it is to manipulate a system by using a falsified biometric identifier. The classic example of a circumvention attack against the fingerprint as a biometric identifier is the "gummy finger." In this type of attack, a fingerprint is lifted from a surface and used to create a mold that the attacker uses to cast a positive image of the fingerprint in gelatin. Some biometric systems have secondary features specifically designed to defeat such attacks by measuring skin temperature, pulse, or pupillary response.

Performance Measurement

The performance of a biometric system can be measured in many ways, but a few primary metrics are particularly important. The *false acceptance rate (FAR)* and the *false rejection rate (FRR)* are two of these.[17] The FAR measures the frequency of accepting a user who should be rejected. This is also called a *false positive*. The FRR measures how often we reject a legitimate user and is sometimes called a *false negative*.

You want to avoid both of these situations in excess. You should aim for a balance between the two error types, known as an *equal error rate (EER)*. If you plot both the FAR and the FRR on a graph, as I've done in Figure 3-4, the EER marks the point where the two lines intersect.

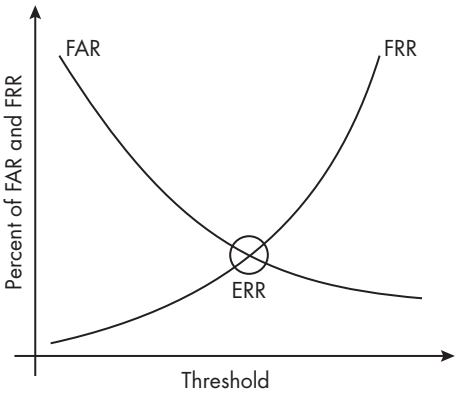

Figure 3-4: The equal error rate is the intersection of the false acceptance rate and false rejection rate.

We sometimes use EER to measure the accuracy of biometric systems.

Flaws in Biometric Systems

Biometric systems are prone to several common issues. As I mentioned when discussing circumvention, it's easy to forge some biometric identifiers. Moreover, once they're forged, it's hard to re-enroll a user in the system. For example, if you enroll a user with both index fingers and those fingerprints are compromised, you could remove these from the system and enroll two of the user's other fingers. However, if you've already enrolled all their fingers in the system, you'd have no means of re-enrolling them using fingers at all. Depending on the system in question, you may be able to select a different set of minutiae for the same identifier, but this avoids the point of the discussion, which is that biometric identifiers are finite.

This issue became tangible in 2015, when an attacker hacked the United States Office of Personnel Management and stole the fingerprint records of 5.6 million federal employees holding security clearances.[18] Repercussions are still being felt a decade later, with legal proceedings still not concluded.[19]

You also face possible privacy issues with the use of biometrics. When you're enrolled in a biometric system, you're essentially giving away a copy of the identifier, whether it's a fingerprint, iris pattern, or DNA sample. After that item has been entered into a computer system, you have little, if any, control over what happens to it. We hope that once you are no longer associated with the institution in question, the institution would destroy such materials, but you have no way to guarantee this. In particular, in the case of DNA sampling, the repercussions of surrendering genetic material could affect you for the rest of your life.

Hardware Tokens

A standard *hardware token* (Figure 3-5) is a small device, typically in the general form factor (size and shape) of a credit card or keychain fob. The simplest hardware tokens look identical to Universal Serial Bus (USB) flash drives and contain a certificate or unique identifier. They're often called *dongles*. More-complex hardware tokens incorporate liquid-crystal displays (LCDs), keypads for entering passwords, biometric readers, wireless devices, and additional features to enhance security.

Figure 3-5: Hardware tokens

Many hardware tokens contain an internal clock that generates a code based on the device's unique identifier, an input PIN or password, and other potential factors. Usually, the code is output to a display on the token and changes on a regular basis, often every 30 seconds. The infrastructure used to keep track of these tokens can predict what the proper output will be at any given time in order to authenticate the user.

The simplest kind of hardware token represents only the something-you-have factor and is thus susceptible to theft and potential use by a knowledgeable criminal. Although these devices provide an increased level of security for the user's accounts and aren't generally useful without the associated account credentials, you do need to remember to safeguard them.

More-sophisticated hardware tokens could represent the something-you-know or something-you-are factors as well. They might require a PIN or fingerprint, which enhances the security of the device considerably; in addition to getting the hardware token, an attacker would need to either subvert the infrastructure that uses the device or extract the something-you-know or something-you-are factor from the legitimate owner of the device.

Summary

Identification is an assertion of the identity of a party, whether it is a person, process, system, or other entity. Identification is only a claim of identity; it doesn't say anything about any privileges that might be associated with the identity.

Authentication is the process used to validate whether the identity claim is correct. This process is different from verification, which is a much weaker way of testing someone's identity.

When you perform authentication, you can use several factors. The main factors are something you know, something you are, something you have, something you do, and where you are. An authentication mechanism that includes more than one factor is known as MFA. Using multiple factors gives you a much stronger authentication mechanism than you might otherwise have.

The common set of tools used for authentication includes passwords, tokens, and biometric identifiers. Each has its own set of unique challenges that you will need to deal with when you implement it as part of your set of security controls.

In the next chapter, I'll discuss the steps that take place after identification and authentication: authorization and access control.

Review Questions

1. What authentication factors could a hardware token potentially represent?
2. What is the benefit of using a password manager to store passwords? How might a user circumvent these benefits?
3. What biometric characteristics might be affected as a person ages?
4. What methods of personal identification might be easily falsified? What approaches are used to harden these methods against falsification?
5. Why are the various forms of common identification generally insufficient for authentication purposes?
6. Using mutual authentication protects against what type of attack?
7. What methods comprise the set of authentication factors? Give one example for each.
8. What purposes does MFA serve?
9. What common forms of identification do we use as individuals in our daily lives? List several.
10. What are some cases where we might see identification substituted for authentication?

Project #3: Hashing Passwords

One of the authentication methods I discussed in this chapter is the use of passwords. How are passwords stored safely on your system so that they can be checked against the password you enter when you log in? What keeps an attacker from finding them and recovering them?

Well, when you set a password on a system, the system doesn't store the password itself, but a hash of the password. A *hash* function is a mathematical operation that produces a value that can act like a fingerprint of the thing being hashed. I'll come back to hashes in more detail in Chapter 6.

When you log into the system, the password you type gets hashed in the same way as the stored version, and the two hashes are compared to see whether they match. If so, you've typed the correct password and are allowed to log in. In this way, the system never has to store your actual password, where it could be vulnerable to attackers.

To practice hashing passwords, you'll use the CyberChef tool discussed in Chapter 1. You'll also need a list of passwords to test. We'll use the same ones from Project #1 in Chapter 1:

- *Tr0ub4dor&3*
- *correcthorsebatterystaple*
- *password*
- *Jtx2NAg6GctawDF_diM_*

Fire up your browser and visit *https://gchq.github.io/CyberChef/*. In the upper-left corner, you'll see a search field under Operations. Enter **fork** in this box, and you'll see the Fork operation show up at the top of the list.

Next, enter **sha2** and drag the SHA2 operation over, right below Fork. In the SHA2 operation, click the **Size** drop-down and select **256**. SHA-256 is a hashing algorithm frequently used for passwords.

In the Input field on the right side, enter the value **password**. In the Output field, you should see the SHA-256 hash of the word *password*, which should look similar to the value in Figure 3-6.

When given the same input, a hash function always produces the same output. To see this fact in action, click somewhere in the Output box and press ENTER to get a new line. Enter **password** again. You should now see a second line in the output box with an identical hash.

Note one issue with this method of handling passwords: If two users have the same password, the system will produce the same stored hash. This would allow an attacker who had access to the stored hashes and knew one user's password to realize that the second user had an identical password, making the stored password much less secure.

Figure 3-6: CyberChef showing hashed passwords

We can fix this situation, to some extent, by using a *salt*, an additional value added to the password before it's hashed to ensure that the hash produced is unique, even when the password is the same. For example, we could add some variation of the username as a salt.

In the output box, add the value **user1** to the front of the first line and **user2** to the front of the second line. You should now see two hashes, as the input values are now different as a result of adding the salt. Play around a bit more with the different example passwords, and different types and lengths of salts, to see how they change the output. This is a somewhat simplified example; modern operating systems use much stronger hashing algorithms and salt mechanisms than those we looked at here.

To go even deeper, use CyberChef to answer the following questions:

- What happens if you perform another SHA-256 operation on the output of the first? Is two rounds of hashing more secure?
- If an attacker could discover both the hashing algorithm and the salt used, is there any way they could discover the password? (Hint: Research rainbow tables.) How might you mitigate such an issue?
- Using a salt makes this storage mechanism more secure. What does a pepper do, and how does it impact the security of the stored password?