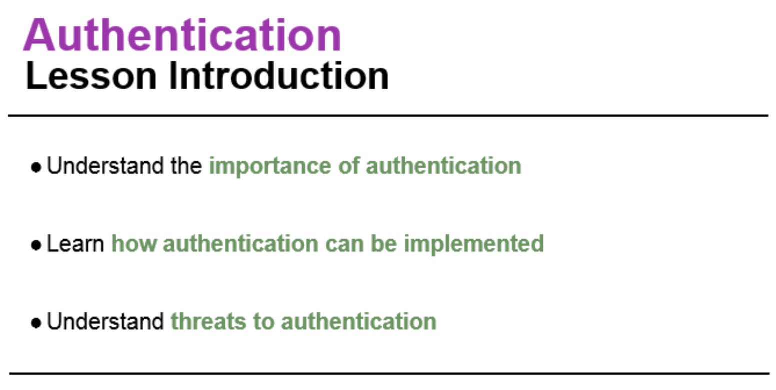
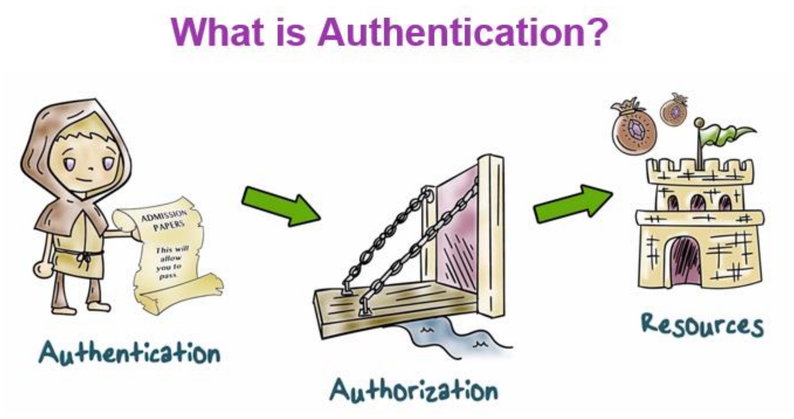


Authentication
Lesson Introduction

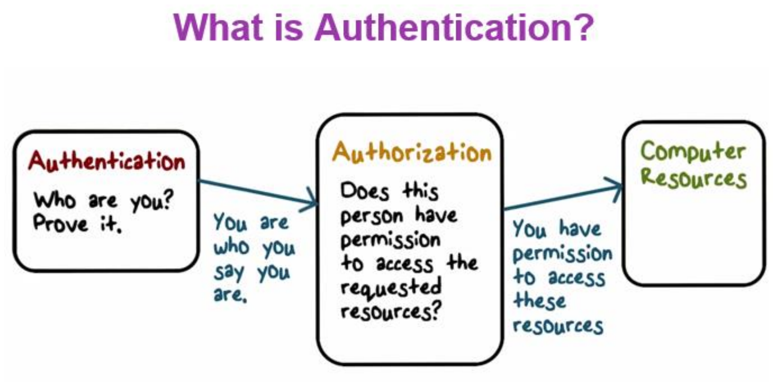
- Understand the importance of authentication
- Learn how authentication can be implemented
- Understand threats to authentication

In the last lesson, we saw that an operating system is really a reference monitor. Every time a request is made for a particular resource, it must say, what is the source of this request? Authentication helps us answer this question. In particular, it tells us on whose behalf a process is running that is making a request for the resource. We will explore multiple authentication methods and how they can be implemented in a computer system.



We can understand the importance of authentication by going back to the discussion we had about what a trusted computing base does.

So we have resources that need to be protected and we can do that by having a trusted computing base that is a reference monitor. Every request that comes for a resource has to be monitored. And when you monitor the request, the question you have is, should this request be able to access the resource for which that is the target of this request? To answer that question we really have to establish who is the request coming from. The source of the request is, we're going to be able to identify that because we have authentication in the system.

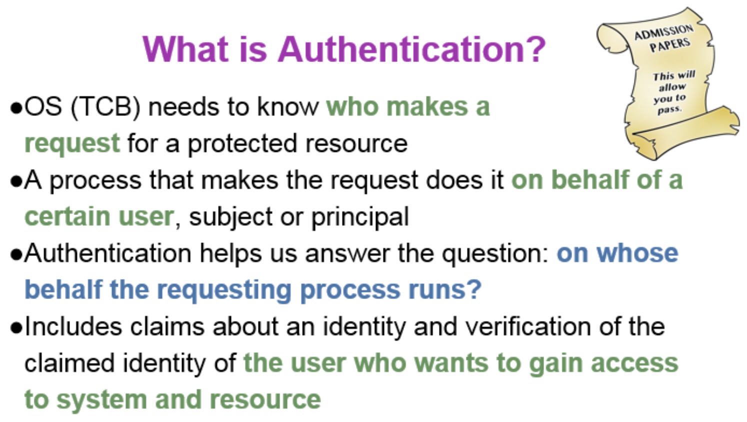

What is Authentication?
Authentication
Who are you? Prove it.
You are who you say you are.
Authorization
Does this person have permission to access the requested resources?
You have permission to access these resources
Computer Resources

So let's look a little bit more closely at what authentication really is.

Authentication, we said we have to establish the source of the request. So we have to ask the question, who are you, if you happen to be the source? And of course, you can't just claim to be anybody. When you're going to claim that you are Alice, you have to provide some evidence to us that is going to convince us that you really are Alice.

So authentication is knowing the identity of the source and establishing that it indeed is that person who is making that claim about the identity. So once we establish the source of a request, convince ourselves that the user is who he or she claims to be, the next part is authorization.

So in authorization we're really establishing whether the source of the request does have the permissions necessary for the resource that they want to access. So this permission check is the authorization process and once authentication and authorization is done, of course then we allow access to the resource when permitted and the source of the request is able to make use of the resources.


ADMISSION PAPERS
This will allow you to pass.

What is Authentication?

- OS (TCB) needs to know **who makes a request** for a protected resource
- A process that makes the request does it **on behalf of a certain user**, subject or principal
- Authentication helps us answer the question: **on whose behalf the requesting process runs?**
- Includes claims about an identity and verification of the claimed identity of **the user who wants to gain access to system and resource**

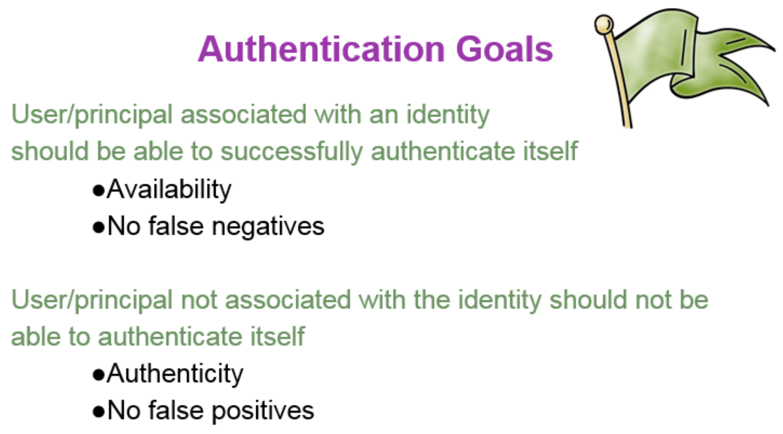
So let's dig a little bit deeper into this question about what is authentication?

We just discussed that the operating system or the trusted computing base, OS is that plus more perhaps. So the OS of the trusted computing base needs to know who is making a request for the resource that is protected. So access to that resource has to be secured.

So we know that in a computer system a request actually comes from a process. This process may be running one of your services, your browser, a mail client, whatever it is. So processes making the request, but we know that processes are run on behalf of users. So a given process that is making a request must be running on behalf on a certain user. We call the user also a subject or a principal. These are the active entities that actually initiate request or cause actions. So these terms are used interchangeably, but we'll stick with user. So process runs on behalf of a user.

Authentication, which is our topic, essentially is going to help us answer this question that we have, is that if a process is making a request to know who's making the request, what user is making the request, we must answer the question on whose behalf is the process running? So the requesting process makes the request by making a system call. It comes through the operating system, but on whose behalf is it making that call?

To establish the user on whose behalf the process is running, of course we have to start with authentication. In the beginning, when a user comes to the system, the user is going to authenticate himself or herself. So that's how we, for example, start a login session. You login to the system, or start the session that's going to launch one or more processes that you need to run, and then in the end the session is going to be over. So when you start the login session, prior to launching the process that is making the request, of course you're going to make a claim about your identity. You're going to tell the system, I am Alice, and then the system is going to ask you, well give me some evidence so I can believe that you really are Alice. And when that evidence is provided we need to verify it. The verification actually tells us that the user is actually who he or she claims to be, and then that user launch an application or start a process. And that process is then going to have, sometimes we say credentials of this user, because this user was authenticated. In other words, those processes are going to run on behalf of that user that just authenticated, and whatever resources this user is able access, those processes will be able to successfully gain access to those resources. So really, requests when we talk about, do come from processes, but processes run on behalf of users. And the system knows what user a process is running on behalf of because of authentication, because that comes before the process gets launched and makes the request.



Authentication Goals

User/principal associated with an identity should be able to successfully authenticate itself

- Availability
- No false negatives

User/principal not associated with the identity should not be able to authenticate itself

- Authenticity
- No false positives

What are some of the goals of the authentication process itself?

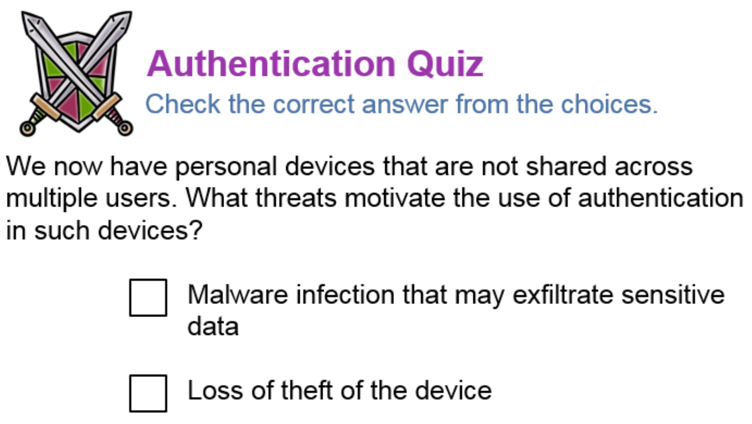
So remember, authentication in a nutshell is a user convincing a system who he or she is, so using some technique or some method. So obviously we're talking about the user associated with a certain identity. Username for example, wanting to authenticate herself to the system. Whatever authentication method we use, what do we desire from it, what should be its goal?

First of all, when a legitimate user tries to authenticate herself, the system can demand some evidence but when the right evidence is provided, the system should allow the login to complete successfully. Well, that is called availability, okay? The system is available to the user who's able to provide the right evidence to support the claim, for example, the user is Alice. So the system should be available to the legitimate user who is able to provide the correct evidence.

In other words, we don't want to have any false negatives. So let's just spend a little bit of time on what do we mean by this phrase false negative that I have here. So negative refers to the outcome. So if the authentication process is successfully complete, that is a positive outcome. A negative outcome is when we deny the authentication request or the login request. So if, for some reason, believe the evidence is not right or whatever it is, so we're going to say login unsuccessful, for example, or authentication unsuccessful. If it's the right user providing the right evidence and the system still has a negative outcome, well that is incorrect. In fact, that outcome we see is not a correct outcome, that result is false. The user was the right one with the right evidence. Whatever method we're using, perhaps, has a problem and because of that it's not able to allow the user in. So if the outcome is negative and that's done incorrectly that is called a false negative. So false negative basically means when we incorrectly deny access or incorrectly decline an authentication request, we don't want to have that. We have an authentication method, we don't want it to deny authentication requests for right users who provide the right evidence.

Well the other possibility is the bad person, Eve, is trying to impersonate her. So Eve walks up to the system and says I am Alice. So it's the user principal that is associated with identity, which is Alice. It's really not her and it's Eve, so the user or principal who is not associated with Alice identity, we don't want that person, or Eve, to be able to login. Well, if you had a guarantee, then we have authenticity in the authentication process. If we authenticate someone, we know for sure that it is that person, so that's the authenticity part.

And another way to sort of think about this is that we don't want to have any false positives. So positive here refers to the outcome of the authentication process. Positive means we allow someone to login or authenticate. And false means we actually make a mistake. We do that incorrectly. If Eve is able to login as Alice, that is not right. That is incorrect. So that's a false positive. So we don't want to have any false positives. The second goal that we have for authentication is that the right user with the right identity and evidence should be allowed in. That means we have no false negatives, and somebody who is going to impersonate a user of the system should not be allowed in which means we should not have any false positives. So these are the goals of a good authentication method that you would want to implement in your system.



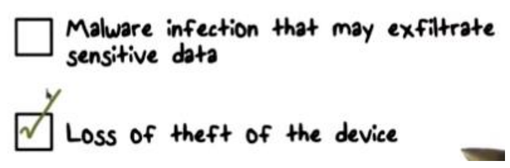
Authentication Quiz

Check the correct answer from the choices.

We now have personal devices that are not shared across multiple users. What threats motivate the use of authentication in such devices?

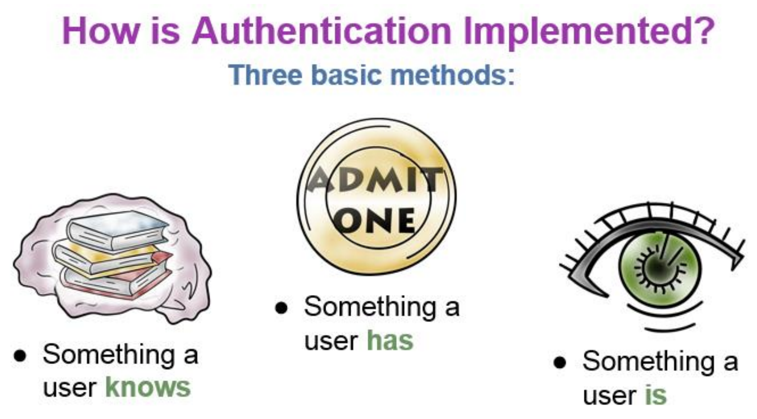
- ☐ Malware infection that may exfiltrate sensitive data
- ☐ Loss of theft of the device

Let's look at our authentication quiz. You should check the answer from the choices that we have here. And this question really is talking about a lot of devices that we have. Those are our personal devices. So the device, it's your smartphone or your laptop. And they're not shared between many different users. Even in this case, we do use authentication. So in a smartphone you may have a PIN or a pattern to unlock it. On your laptop you may have to type a password to gain access to it and things like that. So if it's not shared and we don't have multiple users, why do we have authentication here?



- ☐ Malware infection that may exfiltrate sensitive data
- ☑ Loss of theft of the device

The fact that you authenticate malware infection, malware finds a way to get onto your device and it's running there. So who authenticated, it really doesn't matter. So that first option is not the correct option here. The correct option is actually the second one. One threat with these personal devices is that they can be stolen, misplaced and things like that. And if the device is in the possession of somebody else, the only protection you have is that they authenticate themselves as you. So your PIN or pattern that you have for unlocking your phone, sort of protects in case the device gets into the hands in the wrong person. So authentication that really is addressing the thread that the device may be in the hands of a person who's not authorized to access it.

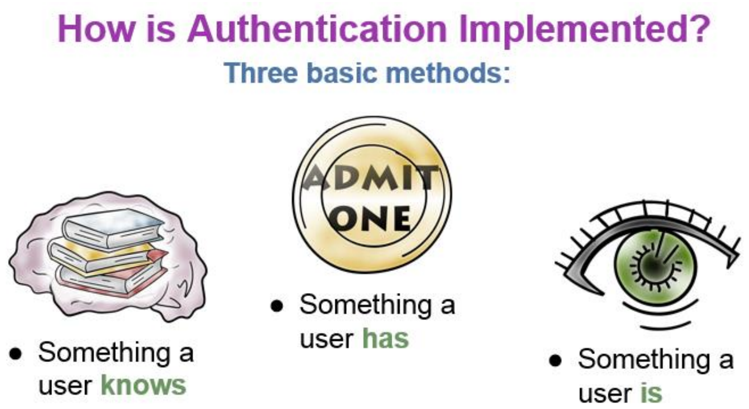


How is Authentication Implemented?

Three basic methods:

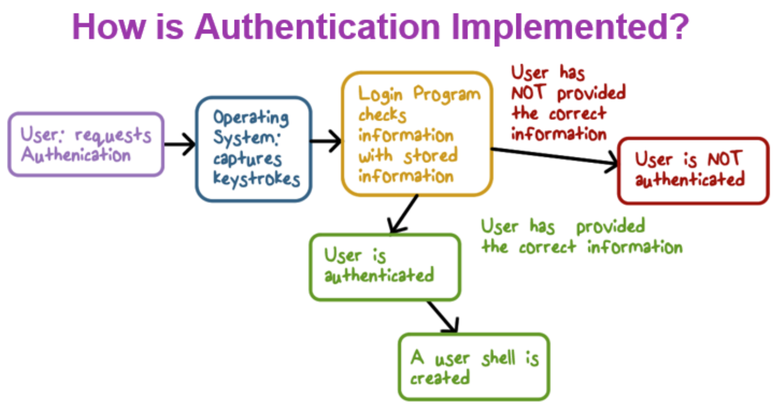
- Something a user knows
- Something a user has
- Something a user is

So now let's talk about how authentication can be implemented. It could be something that the user knows. So this has to be a secret, a secret that is shared between the user and the system. The fact that you are able to produce that secret that's associated with the user, that means you are that person. That is the evidence that you are who you claim to be. A password for example is a secret, isn't it? We say ask people to pick a password that others can not guess or easily guess. Well, that's your secret. So one way to do authentication is that users have a secret that they can produce to the system and the system is able to check that the secret really is the one that is associated with that user.



The other possibility is something the user has. So think of this some sort of a token or a smart card or something that you use and the fact that you have it means you have possession of it. So, possession of that token, we have to talk about how this could be used for authentication. The fact that it's with you, smart card or token is with you, a fob, for example, people use that. That's with you, that means it must be the user who you're claiming to be or the user who should be possessing this particular thing. The first thing is something you would know. The second is something you have on you, or with you.

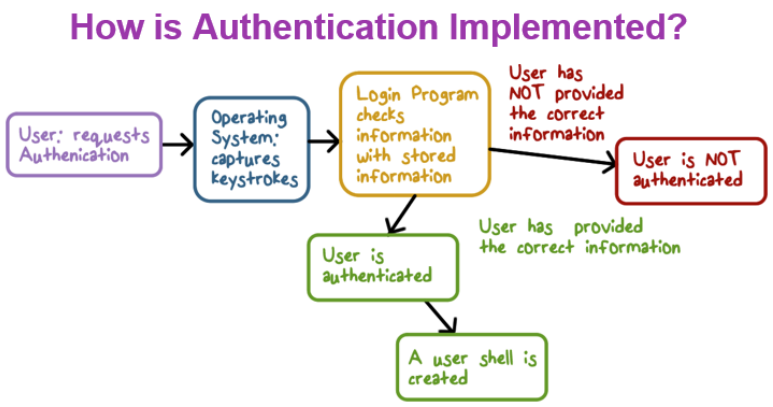
The third thing we can think of is something the user is or something you are. So this could be for example your finger print. This could be your voice, for example, people talk about voice biometric, or their smartphone devices that you can talk to, for example. So this would be a biometric, something that is unique, hopefully, to the user, because we don't want somebody else to become you, so this biometric has to be something that's specific to you. And if you're able to produce that, the system you're able to give that to the system and the system is able to check that actually it is your fingerprint then the authentication is based on something the user is. Something you know, something you have, or something you are. These are sort of the three basic methods that are used in almost all the authentication techniques that we use in computer systems.


How is Authentication Implemented?
User: requests Authenication
Operating System: captures keystrokes
Login Program checks information with stored information
User has NOT provided the correct information
User is NOT authenticated
User has provided the correct information
User is authenticated
A user shell is created

So we are going to again look a little bit more deeply into how authentication is implemented, using one of these methods that we talked about.

A secret you share with the system, or something you have, or something you are. So the basic sort of sequence of steps that we're going to have is that the user is actually going to come to the system and is going to request access to it. And we know that is going to start with request for authentication, you first have to authenticate yourself.

The system, the operating system here is going to take whatever claim about identity you make, so for example you may be saying this is my log in name. So it's going to read those keystrokes and that's who you are. And if you're providing, for example, a password, then the operating system at this point is going to run this login program and the login program is going to check that information that you provide. For example, the password you provide is really your password. Is the password we have associated with this user? If there's a match with what the system knows and what you provide, that is a true positive. If it's the right user then that's a true positive, the outcome is positive. Here, so the user is

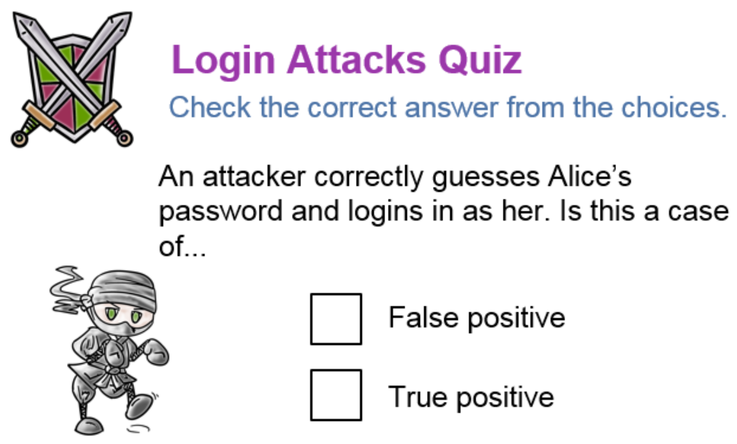

How is Authentication Implemented?
User: requests Authenication
Operating System: captures keystrokes
Login Program checks information with stored information
User has NOT provided the correct information
User is NOT authenticated
User has provided the correct information
User is authenticated
A user shell is created

authenticated and when the user is authenticated, this is how you can now initiate any further actions you want to perform.

So this is sort of the case where the right user provides the right kind of evidence. The log in program is able to check and successfully log in the user.

Well, the check could fail and if it fails then the system believes that the user has not provided the correct information. And if the user has not provided the correct information, it's probably not the user who is being claimed. The identity that is being claimed probably doesn't belong to the user who is asking for authentication. So this could be impersonation for example.

So in this case, the system doesn't have the right evidence, doesn't have this match, so it's not going to authenticate the user. So in this case, authentication fails. Maybe you retype your password. If you're not the right user, then that's a good thing. So then, that if you are not the right user, even you are trying to login as Alice and the system says no, that's good. That is a true negative actually because outcome is negative and it's the right thing. Okay so Eve tries to login as Alice and we end up here. That's a true negative. When Alice tries to login as Alice. We come here, that is a true positive. So this is how the authentication process is implemented. Couple of things to know here is that we have to read the evidence or capture the evidence. We have to compare it and then decide whether there's a match or not and go either this path or the other path.



Login Attacks Quiz

Check the correct answer from the choices.

An attacker correctly guesses Alice's password and logins in as her. Is this a case of...

- ☐ False positive
- ☐ True positive

An attacker correctly guesses Alice's password. So the method here is something you know in particular you know, a password. So he was able to guess Alice's password and is able to login to the system. So is this a false positive or a true negative is the option that you have to pick?

[✓] False positive
☐ True negative

So we're going to pick the first option here. It's positive because A is able to log in. So login process is successful or the outcome of the login process is positive. Okay, that's why we're going to pick positive.

Why is it false? It's false because it's an incorrect outcome because Eve was able to log in as Alice, stolen or guest password, whatever it is, we actually have sort of a security problem here isn't it. Alice's account is being compromised. We allowed authentication, that's the positive. And it's an undesirable thing that we just did, so that reference to the outcome not being desirable, which makes it false. So this is a false positive. It will authenticate even if it's authenticating as Alice, which is undesirable and the positive is actually false here. It's not a true negative. Negative would have meant that the authentication process doesn't succeed. Request is denied, which is not the case.

Implementation Quiz

Check the correct answer from the choices.

A number of online banking systems send a limited lifetime PIN to your smartphone for you to be able to authenticate yourself to the bank. Is this an example of...

☐ Something you have
☐ Something you are

So here again you choose the correct answer. The question is saying many online banking systems including the one I use. Actually they send you a pin when you login as a SMS to your smartphone. And they ask you to type that pin. So you had to provide that pin in addition to the password that you might have. So is this an example of authentication with something you have or something you are? So which case it is.

[✓] Something you have
☐ Something you are

Well it's not something you are, there's nothing biometric about it. It's a device that you have that receives a code. Since there's a device that allows you to authenticate yourself, and device is something you have, this option would be the right one. Okay, so the smartphones are essentially being used as something you have part of the authentication method that we're talking about.

Threat Modeling of the Password Method

- **Guessing the password** for a given user allows impersonation
- **Impersonating** a real login program
- **Keylogging** to steal a password

Somebody is trying to compromise the security of your system, somebody who is not authorized is trying to access your sensitive data.

How can they attack, if what's they're asked to attack or what they're allowed to attack is the authentication part of whatever security you have in your system. So authentication is how you start. So the target of the attacker here is going to be how you authenticate. In particular, what they're going to try to do is defeat the authentication system. So Eve is going to try to be successful in authenticating herself as Alice. In that case, you would have successfully compromised the authentication method. So we're going to do this threat modeling saying well what can they do. What can the bad guy or that hacker do.

Well, let's just sort of talk about the password method. So remember the password method is something you know. And we all know what passwords are. So let's say passwords are being used by users to login to a system. What kind of threats do we need to be concerned about?

- **Guessing the password** for a given user allows impersonation

Well, it's a secret that's shared between you and the system, well, someone can try to guess that secret. So one threat is the not having good passwords that can be easily guessed by attackers. That's one threat we have to be concerned about, authentication that is based on passwords.

- **Impersonating** a real login program

So this is kind of interesting. It really is not talking about, how can you be a user, guess their password, or steal their token. But it's talking about impersonating a real login program. So think about you come in, and we're talking with passwords, you provide your password to the system. How do you know you're really talking to the system? How do you know it's not a program that is impersonating the system? All it has to do depending on which system you have, it has to display login, call in, or something like that for you to provide your login ID. Or login name and then it has to display password colon so you can type your password, or whatever the interface that you have. This a Trojan horse, this a program that is impersonating the system at the other end. Then what this program is able to do is, as you type your password, it'll be able to steal it.

- **Keylogging** to steal a password

So what's the last one? We're talking about someone trying to steal your password, isn't it? If your computer is infected, in particular if it is infected with malicious software called keyloggers. They are able to capture the keys that you press, you are going to provide your password by pressing a bunch of keys. And so keylogger grabs your password.

Okay, so if you are doing a password method, bad guy has to get hold of your password, so they become you. They can guess it, they can steal it, either using a piece of software that fools you into believing that you're giving your password to the operating system. Or it could run as the real login program is running and steal these keys that you press or the characters that you press that make up your password.

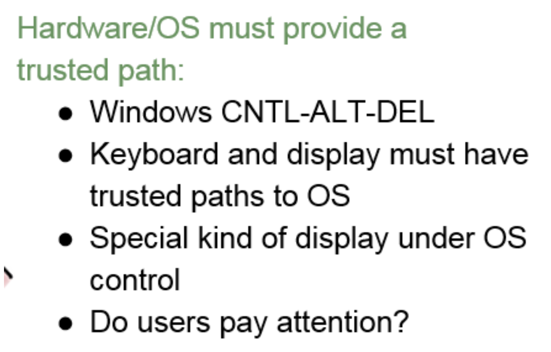
Importance of a Trusted Path

Hardware/OS must provide a trusted path:

- Windows CNTL-ALT-DEL
- Keyboard and display must have trusted paths to OS
- Special kind of display under OS control
- Do users pay attention?

So trusted path means that you really talking to the trusted computing base or the operating system, okay. So path that connects you typing the keys and where these keystrokes are being captured. That path is really connecting you and the trusted computing base and there's no one else in between. If that is the case, you have a trusted path.

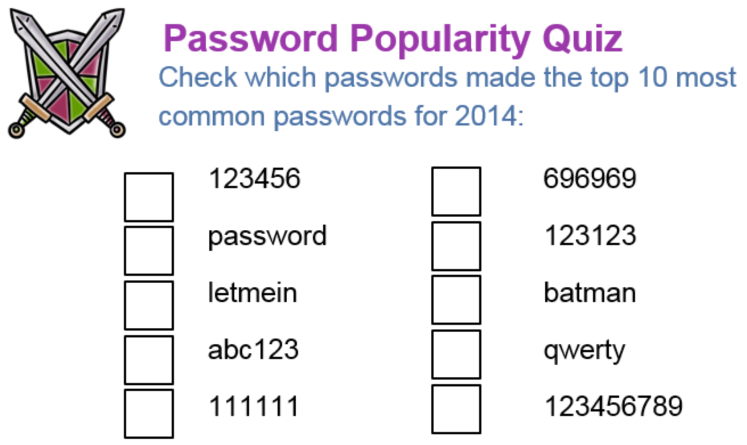
How do you make sure you have a trusted path? We're going to talk about a couple different ways. This trusted path really has to be provided by the operating system, maybe some combination of hardware as you look at combination of the operating system and the hardware as we're going to see. So for example, if you login to Windows, you have to press this key sequence, Ctrl+Alt+Del. The idea is that you press this key sequence. This sequence cannot be trapped by any other program. Great, this is going to take you to the operating system. So the display and the keyboard has to be connected to the CPU on which the operating system is running in a way that there is no one in between. Display is not under control of somebody else. And somebody else is not able to capture the key strokes and perhaps even alter them and things like that. So generally these are sort of physically connected, isn't it and they're in close proximity, they're right next to your CPU, both the display and the keyboard. So we assume there's no one else in between.



Hardware/OS must provide a trusted path:

- Windows CNTL-ALT-DEL
- Keyboard and display must have trusted paths to OS
- Special kind of display under OS control
- Do users pay attention?

But if you're paranoid we can do sort of couple of different things to make sure that really we're not talking to somebody else but the operating system. People had to propose a bunch of different ideas to make sure that when you're authenticating yourself you have a trusted path. They said there's a special part of the display that only the operating system can write. So for example you could indicate on the display when you're talking to the operating system and when you're not talking to the operating system. Or actually have a light on the keyboard. Okay, that will light up when you're talking to the operating system and it will be off when you're not. So this way the user would know that when it's providing, I don't know, his or her password, it is actually going over a trusted path to the operating system. The other entity, or the entity at the other end actually, is the trusted computing base. So one challenge with these kind of things is, do users pay attention to something you display, or a light on the keyboard or something like that. If it's a red, flashing light maybe they do, or something like that. But the idea here is I think the concept of a trusted computing path is something we want to understand.

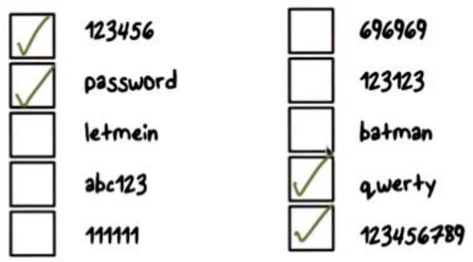


Password Popularity Quiz

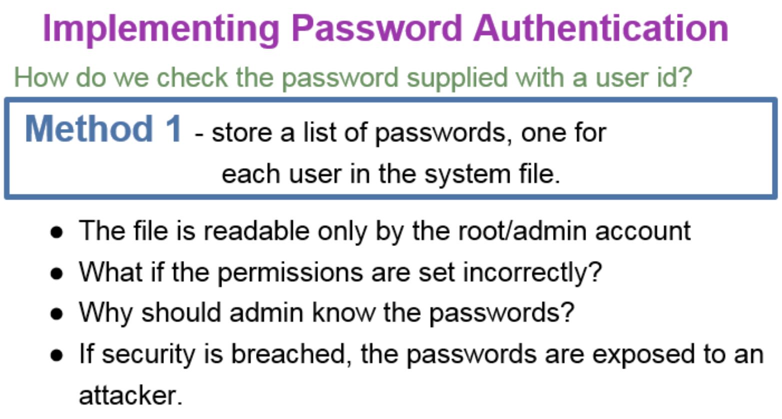
Check which passwords made the top 10 most common passwords for 2014:

☐	123456	☐	696969
☐	password	☐	123123
☐	letmein	☐	batman
☐	abc123	☐	qwerty
☐	111111	☐	123456789

Here are sort of let's guess about, and since you are in a security class, you may have somewhat of a different mindset. But let's try to think what kind of passwords a lot of people use. Okay. What are common passwords? So, check everyone you think made the top ten list.



Actually, password is a pretty popular password people use. It did make the top 10 list. 123456 actually has also made the top 10 list. Again, maybe it's the ease remembering it that's why people use it. Here someone is picking a longer password. It's not weird if it's a stronger password or better password, but it's commonly used. qwerty is faster to type, well actually that made the list also. Some of the other ones that we have here actually are on the longer list but not on the top ten. What else could be on top ten? A lot of sports fans, baseball, football, made the list. Actually 12345, 123456, All these are on that longer list. So, I think one takeaway here is that people put their ability to remember their password, perhaps ahead of how good those passwords are. And they continue to pick passwords that are pretty weak. And we're going to talk about why common passwords are a bad idea.



Implementing Password Authentication

How do we check the password supplied with a user id?

Method 1 - store a list of passwords, one for each user in the system file.

- The file is readable only by the root/admin account
- What if the permissions are set incorrectly?
- Why should admin know the passwords?
- If security is breached, the passwords are exposed to an attacker.

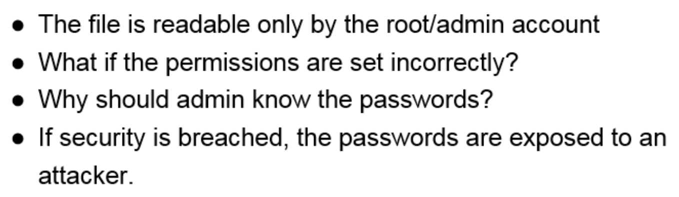
So now let's talk about how password-based authentication can be implemented.

So we know the evidence that's going to be provided is a password user is going to supply, and then they're going to type or touch the screen or something like that. And that's going to follow, you're telling the system who they are, which typically they provide the user ID. So they tied their user ID and password, let's say. And the system now has this task of, well, how do I know that this is the correct password for this user, and if it is the correct password, authentication will be successful. So how can we perform this check?

So you would say well an obvious solution to this problem is that we ask users to share their passwords with the system. So the first time this is called registration or enrollment or some out of band method by which. Of course the system has to know something about your secret. If it doesn't, it can't check it. So let's say the secret is password so we have a set of users with passwords. And the system stores those passwords. It's trusted system we, let's say, give it our password. And if we do that, it's not a good idea and actually we won't have to do that. But if you do that than how can we do that check that we're talking about. I will see it's pretty straight forward. All we have to do is impair the type password with the one that's stored.

So where do we store it? We have to store it in a file. People come back, you turn the computer off and then come back and login so this information has to persist. Your computer multiple sessions, so persistent information is stored in files. One thing we should be smart enough to do is that only the trusted root or admin be able to read this file. The login program is going to run on behalf of them and they are the ones that are going to check and somebody's trying to authenticate themselves. So this file should be made readable only by the admin or the root. The most user we have in the system.

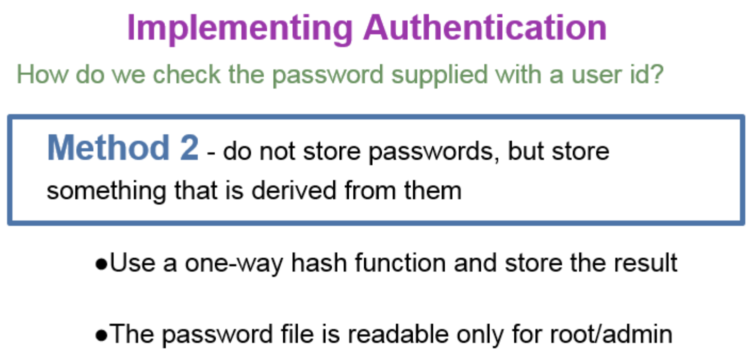
But what if we make a mistake? What if permissions are set incorrectly? The permissions are set incorrectly somebody else may be able to read that file. And if they're able to read that file, they're able to learn everyone's password, which is not a good thing.



- The file is readable only by the root/admin account
- What if the permissions are set incorrectly?
- Why should admin know the passwords?
- If security is breached, the passwords are exposed to an attacker.

Even if the permissioners are set right, why should an admin be able to know all the passwords? Okay, every user's password, if they know it, there's no reason for them to have access to passwords of all users in the system.

And the one reason we don't want to have that is that in case you have security breach, the passwords are going to be exposed, everyone's password is going to be exposed to that hacker. So it's clear that something about the secret has to be shared with the system, the password has to be shared with the system. But storing those secrets in a file, even if you do access control, is not a good idea. So can we do better?



Implementing Authentication

How do we check the password supplied with a user id?

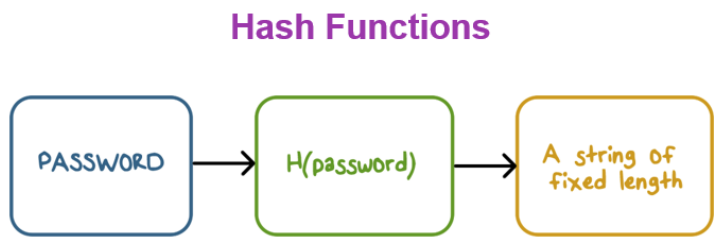
Method 2 - do not store passwords, but store something that is derived from them

- Use a one-way hash function and store the result
- The password file is readable only for root/admin

So indeed, we can actually do better, and the way to think about that is that we don't store the passwords themselves, but store something that is derived from it. It's not the actual password, but something that is possible to get hold of only if you had access to the password at some point. Or at least in the beginning or the enrollment time or when the password was changed and then you're not storing the full password but just something derived from it.

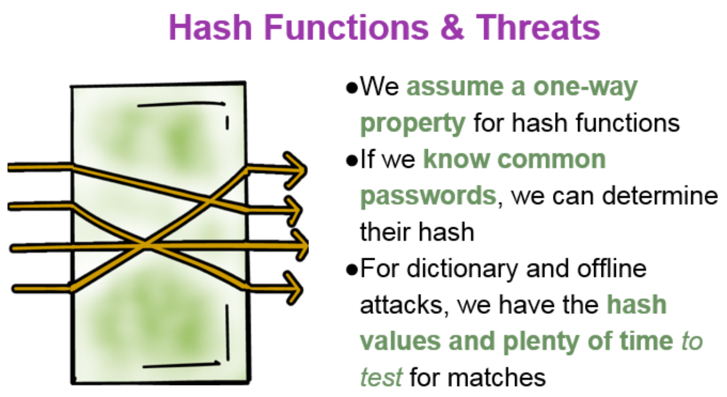
The question is if you're storing something derived from what is it and how do you use it. The result that we derived from password. One way to do that is to use what is called a one-way function. This called are called hash functions. Hash functions are widely used to take an arbitrary length of input and produce a fixed size output that is fairly unique to the input. And one-way basically says that it's easy to compute given the input of the hash value is easy to compute, but it's very hard to compute the input if you just had the hash value. So such functions exist, we're going to talk about some examples of those.

So thing to remember here is that given a password it produced a hash value of that password. And going from password to hash value is straightforward going in the other direction which is from hash value to the password is really hard. So what you're going to do is you're going to use a function like that so rather than storing the password we're going to store something derived from it. In particular, we're going to store a hash value, and that's what we're going to have, and even the hash values that we have, those have to be stored in a file as earlier we were talking we're storing the passwords in a file. Even this file should be readable only for the root user or the admin user that we're talking about on whose behalf the log in program is going to run.



A hash function basically as we said, the input is a password, this box is the hash function, whatever computation goes on here is going to happen here. And the output, this password could be any length, could be a past phrase that combines a bunch of words, letters, digits, whatever it is. But it produces the hash value is of a fixed length.

In this direction we said, it's one way so it's easy to go. It's easy to compute this function given this input and for us to produce this output. If it just had this and wanted to invert it going the other direction that's going to be really hard, which is the basis of which we're talking about here.



Hash Functions & Threats

- We **assume a one-way property** for hash functions
- If we **know common passwords**, we can determine their hash
- For dictionary and offline attacks, we have the **hash values and plenty of time** *to test* for matches

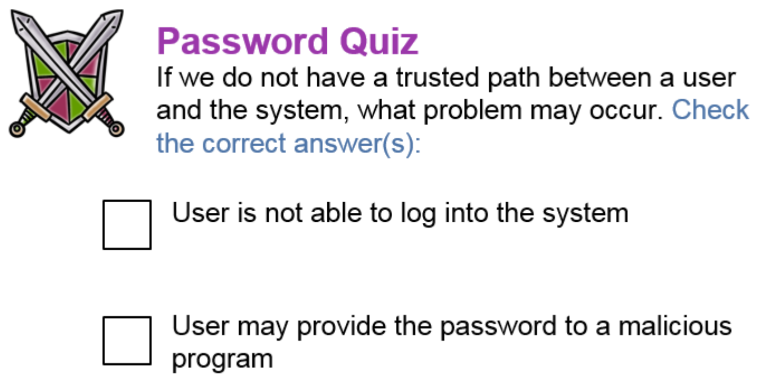
Given that we're going to derive something from a password using a hash function and store that, let's talk about a couple of threats. Because of the security mindset we have, anytime we sort of make a decision, we want to think about its security implications. So what are some threats?

We are going to assume that the one-way property of the hash functions holds. You make some assumptions so we are going to assume that inverting or going the other direction from hash value to a password, we're not going to worry about that.

Let's say we're talking about common passwords. You know that password is a common password, or one, two, three, four, five is a common password, but this hash function is known. This one-way hash function we're talking about, we're not doing security, web security, so it's known. We can compute the hash values off the common passwords. And we're saying these hash values are stored by the system, those are the derived values. If you can get hold of the hash values the system has and if you can find the hash values for the common passwords among those stored values, it's not hard to see how we can find out passwords of certain users in the system. So if the hash value of 1, 2, 3, 4, 5 matches with the stored value, we know that user's password is 1, 2, 3, 4, 5.

The other kind of attacks we can mount, one is called a dictionary attack. So it's like a dictionary has lots of words and think about a long list of passwords, a dictionary of possible passwords. Words and maybe they're mutated in a certain way, you add a few digits or something like that, but you have this called space of passwords or the dictionary of passwords, and you can try to brute force each one of them. That's called a dictionary attack.

An offline attack is one where we're not interacting with the system. We're not trying to log in each time, we try a different password. If you do that, then you are, that's called an online attempt we are making. We are interacting with the system so when you do an online, sort of guessing attack, the system can stop you after a certain number of tries. It locks you out. So, to avoid that problem we can sort of offline in our free time, we can take the hash values, take a dictionary and sort of compute for each dictionary word the hash value and search for that in the set of hash values that we have. When we find a match we know our password. And we can do this as I said without interacting the system. We interact with the system only when we discover a correct password. So those are called offline attacks and dictionary attacks, sort of brute force attacks. So one thing even when we store derived values, in particular hash values here, these kind of attacks are always possible. Common passwords, the hash value, dictionary or offline attacks. The assumption here is that the attacker does have access to the hash values. And the hash values don't immediately allow him or her to find out the password. They can try different, or try to guess the passwords that match and when they run into a match for the hash value then they know for sure that this is the password. So these are the kind of attacks that are possible even when you only store hash values.

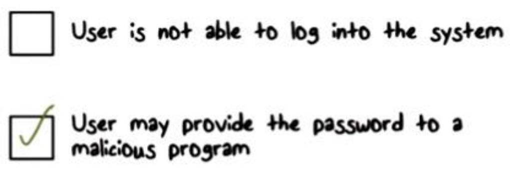


Password Quiz

If we do not have a trusted path between a user and the system, what problem may occur. Check the correct answer(s):

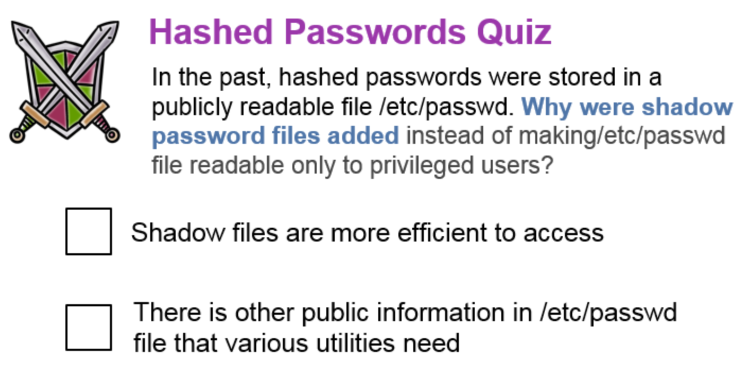
- ☐ User is not able to log into the system
- ☐ User may provide the password to a malicious program

So we talked about a trusted path, remember? So, between the user and the trusted computing base he or she wants to log in. They want to provide the password to the trusted computing base. Okay? So they need to have a path to the trusted computing base, and that's what we talked about when we were talking about a trusted path. So this question's saying what problem would arise if we don't have a trusted path?



- ☐ User is not able to log into the system
- ✓ User may provide the password to a malicious program

Well if you don't have a trusted path, what is the problem? So not the case that the user, so let's say what the problem could be that we may have a Trojan, okay, that creates an interface that looks like what the operating system to be do we normally would have. So, for example, would ask you for user id and then when you type that it may you know next ask you for your password. Trojans may sort of do exactly that. You just don't know you're talking to the Trojan so if it does this that steals your password and logs you in anyway. Then it's not the case that you are not able to log in. What's really happening is that you're giving your password to a Trojan who is going to steal it so you're giving your password to a malicious program because if you don't know for sure that you're talking with a trusting computing base that means you don't have trusted path. So possibly, potentially you're giving your password to some malicious code.



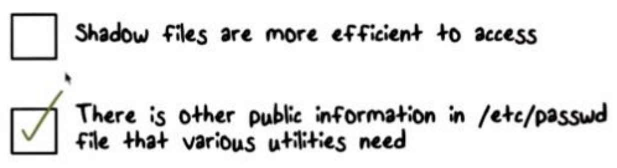
Hashed Passwords Quiz

In the past, hashed passwords were stored in a publicly readable file /etc/passwd. Why were shadow password files added instead of making/etc/passwd file readable only to privileged users?

☐ Shadow files are more efficient to access

☐ There is other public information in /etc/passwd file that various utilities need

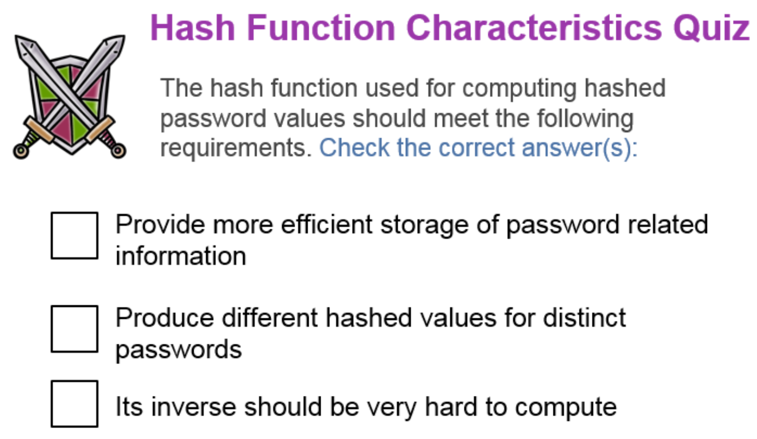
The question here is why did we add the shadow password files instead of protecting the earlier publicly readable file, which was /etc/passwd? Why couldn't we just make that readable to sysadmin or admin or root user, rather than separating the hash values into a shadow file?



☐ Shadow files are more efficient to access

[✓ There is other public information in /etc/passwd file that various utilities need]

So the reason we have to separate and we couldn't just make the /etc/password file readable only for root is that actually it included public information that other programs or utilities used. So if you turned off information for that file, in particular the /etc/password file then certain utilities will not function properly. So this actually was the reason for shadow files.



Hash Function Characteristics Quiz

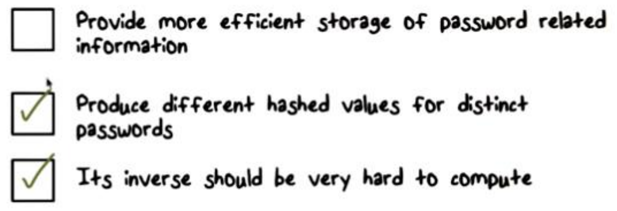
The hash function used for computing hashed password values should meet the following requirements. Check the correct answer(s):

☐ Provide more efficient storage of password related information

☐ Produce different hashed values for distinct passwords

☐ Its inverse should be very hard to compute

So hash function used for computing hash values. If it's a good one, and serves the purpose, then what properties or characteristics it should have? Or what requirements it should meet? Should it be efficient storage of the hash values? Unique hash values for different passwords? Or should going back in the reverse directions be hard? So these are the options. Chose one or more that you think make sense.

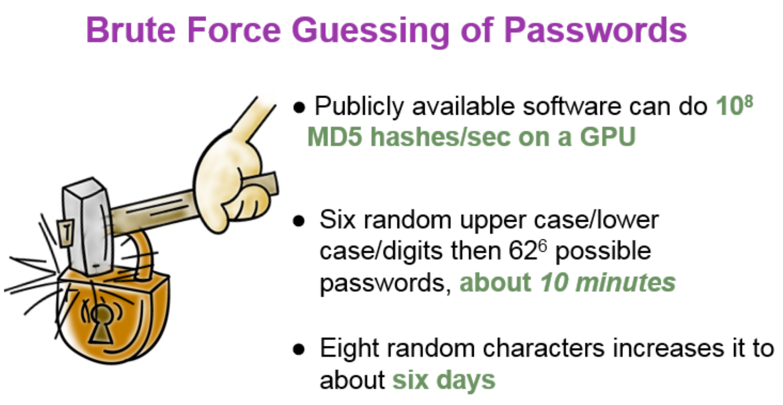


☐ Provide more efficient storage of password related information

[✓ Produce different hashed values for distinct passwords]

[✓ Its inverse should be very hard to compute]

We already talked about the one way requirement of these kind of functions. So given a hash value, we can't go back to the password itself. So this is obviously a requirement. The inverse should be very hard to compute. Actually, we do want different hash values for different passwords, because what would happen otherwise? If user one and user two had the same hash values, then user ine can actually log into user two's account, using user one's password, because user two's hash value is what we compare. So, when user 1 claims to be user 2, types his password, the right hash value is produced, right for user two. So user one is able to get into user two's account if we don't have the different hash values for distinct passwords. So second option is also a requirement. Provide more efficient storage for, that's not really a big deal. We said passwords produce a fixed size output. Could be a couple of hundred bits, 128 or 256 or something like that, but storage itself is not a big deal.



Brute Force Guessing of Passwords

- Publicly available software can do **10^8 MD5 hashes/sec on a GPU**
- Six random upper case/lower case/digits then 62^6 possible passwords, **about *10 minutes***
- Eight random characters increases it to about **six days**

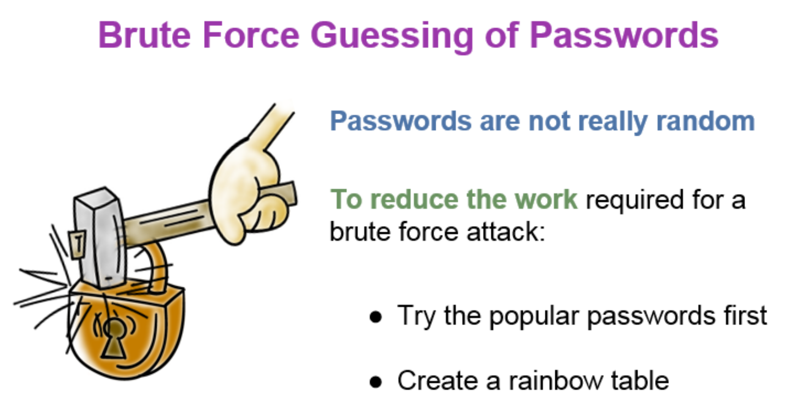
So what can that hacker still do? So this brute force guessing, they'll have to get hold of the shadow password files. They'll learn the hash values.

So let's say there is a compromise of the system and that file is exfiltrated or something like that. In fact in the past year or two, there have been several examples where LinkedIn and Adobe and others had these password, derived values from password files, those were stolen by hackers and actually made widely available. So if you have these hash values, let's say if you get a hold of them, you're doing brute force guessing of passwords. How hard is this task?

One thing we can do is we have some hardware software and whatever hash function here, we're talking MD5. That's one of the hash functions that is used. Publicly available software that we have, and certain kind of hardware that is common and quite popular for graphics. GPUs are graphics processing unit, and they're particularly well suited for doing password kind of calculations or brute force kind of attacks we're talking about. So a GPU with available software can do 10^8. So that's 100 million hashes it can compute and check per second.

Okay, so if the password is only, let's say six random upper or lowercase, okay, 26 lowercase, 26 uppercase, all digits, that will give us 62 total number of characters. If you have 62 possibilities, and it's 6 characters long, the first character, there's 62 choices. The second character, there's 62 choices, the 6th character, there's 62 choices. So total number of possible passwords we can have is 62 times 62, 6 times, or 62^6. If you actually do your calculation, you see how much that is. Divide that number by 10^8, because that's how many we can do per second. If you try to find that out, you'll see that this calculation, or this brute force attack, guessing attack we're talking about, can be carried out in about 10 minutes.

One thing we do is we increase the size of the password. We ask people to pick longer passwords. So from six if we go to eight, of course, the effort for that hacker is going to go up. It's going to go up from minutes to days. But if you take eight random characters of the same kind, you would actually get 62^8, because there are 62 choices for each. If you add special characters, the calculation would change a little bit, but if you calculate this number, again divide this by 10^8, it's going to give you 6 days. You don't have to limit yourself if you're the attacker to one GPU. If you add GPU, this kind of computation can be done in parallel and you can reduce the time if it's too long. But the idea here is that brute force guessing of passwords, even when they're reasonable respectable length, is certainly doable for an attacker.



Brute Force Guessing of Passwords

Passwords are not really random

To reduce the work required for a brute force attack:

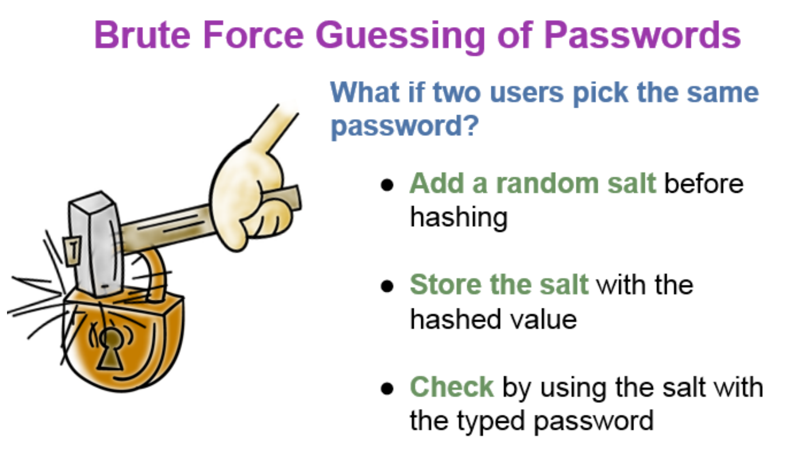
- Try the popular passwords first
- Create a rainbow table

Passwords are not really random.

So when somebody's trying a brute force attack, they don't have to sort of search for things in the dark. What they really, if they're smart, they would start with more popular passwords. So the 12345 or password or whatever it is.

So you absolutely can sort of reduce the amount of work that you have to do by trying out more popular passwords first.

In fact, what you can do is create what is called a rainbow table, which is nothing but potential passwords and their hash values and as a table that you create. The hashed value table for each possible password and so this is, you don't even have to run your hash function when you're doing this brute force guessing. You're just doing a lookup in a rainbow table saying this password let me see what it's corresponding hash value is and does that match some entry in the rainbow table without having to compute the hash function. You can have these other tricks you can use if that hacker is use you rainbow table or be smart and try more popular passwords first.



Brute Force Guessing of Passwords

What if two users pick the same password?

- **Add a random salt** before hashing
- **Store the salt** with the hashed value
- **Check** by using the salt with the typed password

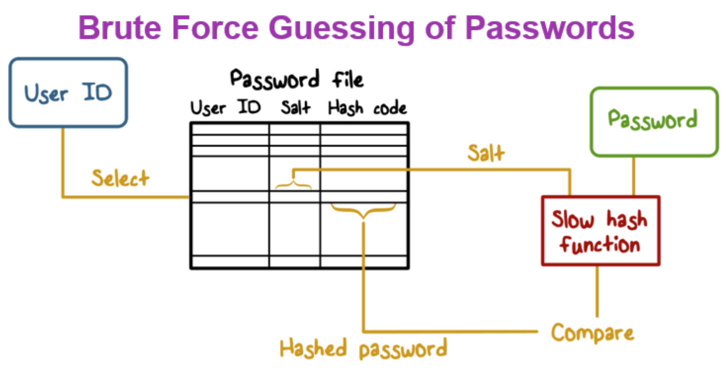
The couple of other things related to passwords and their guessing and things like that, we said, what if two users pick the same password?

If you have common passwords, then of course 12345 is taken by lots and lots of users. How do we avoid this problem? In one of the quiz questions we say, what would happen if the hash value is the same. It's sort of related here. If they do have the same password, they'll have the same hash value. And by looking at the hash values, we can see that from one person's password we can login as the other user and that would be a problem.

Fortunately, we have a solution to that, we add a random number before we do the hashing. We take the password strength, add this random salt. The random salt will be different for different users, that makes the hash value different for different users. If two users pick the same password, without the salt, the hash value will be the same, but with the salt, the hash value will be different. And we will know whether the passwords are the same or they're different because the hash values could be different, either because of the passwords being different or the salt being different.

You'll have to store the salt because later on when we do the check, we're going to get a password. We'll need the salt before we can run the hash again. So we can compare it with the hashed value. So we're going to need the salt. Where do you store the salt? You store it in the file where you store the hashed value. Whoever has access to the hashed value would also have access to the salt, which is fine. Actually not a problem.

So this checking, when we have salt, is done by combining or computing the hash function with both the salt and the password.



To see how that is really done, let's just look at how unique started for example.

So you provide a User ID and your password. We use the User ID to select the entry in the password file for this user that we're talking about. The salt is stored here. The hash value's stored here, isn't it? This is sort of the data structure we keep.

To log in, you're going to supply your password, so we start on this side later on. So this is what is set up initially for create the user account or set up the password. At that time, we determine the salt and computer hash value and things like that.

So later on a user comes types the password. Well, they're saying who they are. You give your user id, your log in name. Based on that, we're going to find the hash value and the salt. So the salt and the password are going to be put through this hash function. And it's slow because if somebody's doing brute force, it pays to slow them down. So we passed these two things through the hash function. So this is typed by the user based on the claim by the user, who they are. This is what we find in the file that we have, where we store the salt value. So the two are passed as input to this. The result is compared with a stored hash value. If the two match then we allow authentication is successful. If they don't match, then authentication fails.

So this is what happens when we have makers of salt to deal with different users making the same password and the result having the same hash value. And that would be a problem. When you add salt, the rainbow based, rainbow table based attack, think about what would happen to that.

Unique PINs Quiz

How many unique four digits PINs are possible?

Check the correct answer:

- ☐ 1,000
- ☐ 100,000
- ☐ 10,000
- ☐ 1,000,000

A lot of us use PINs on our smart phones or ATM access and things like that. So if the PIN is four digits, how many different PINs are possible? If somebody was doing a brute force, they may have to try all possible values. How many of those possible values are there?

✓ 10,000

For each position we have ten possibilities, zero through nine. So position one, there are ten possibilities. Two, there are ten. Three, another ten. So, it's ten times ten times ten times ten, because there are four digits in the pin. So that's ten to the four, or 10,000 is the answer that we get. So, there are ten thousand PINs if you have four-digit PINs. So if you're doing a Brute Force Attack, that's a fairly small number as we saw how many operations we can do per second.

Brute Force Quiz

A randomly chosen password has six characters that include upper and lower case letters, digits (0-9) and 10 special characters (examples are +, ; etc.). In the worst case, how many attempts must a brute-force method make to determine a password when its hashed value is available?

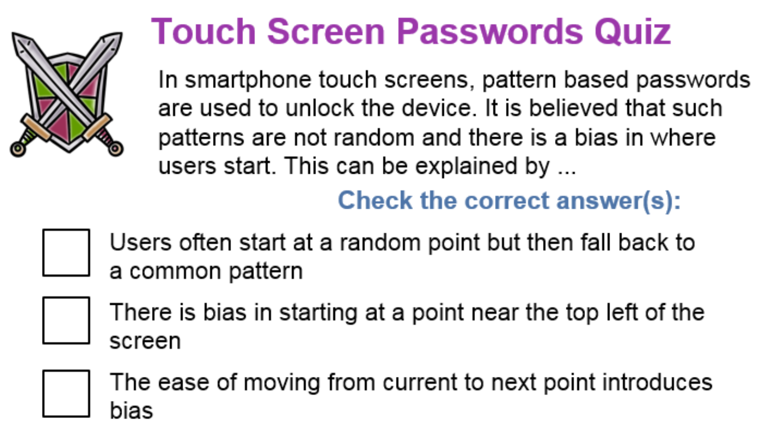
Check the correct answer:

☐ 6^{72} ☐ 62^{6} ☐ 72^{6}

So here we're saying, password has six characters, it must include an upper, lower letters, digits and some special characters. We're adding those. Plus or semicolon or something like that. In the worst case, how many attempts must a brute force method make to determine a password when its hash value is available? You have the hash value of the password, you know the password is six characters long and you know the structure of the passwords. They can have upper, lowercase letters, digits and some special characters. So how many total such passwords are possible that we may have to try in the worst case?

✓ 72^6

It's going to be some number multiplied 6 times. So, the exponent is going to be 6. And n to the power 6 whatever that number n is. It's not the first option. So let's say 26 letters upper and lower case will make that 26 times 2, or 52 digits, makes it 10 more. That's 62, and the special characters, if you add 10 of them, that's 72. The answer is going to be the last one. It's 72 to the power of 6, because each position we have 72 possibilities and there are 6 positions in the password, 6 characters in a password.



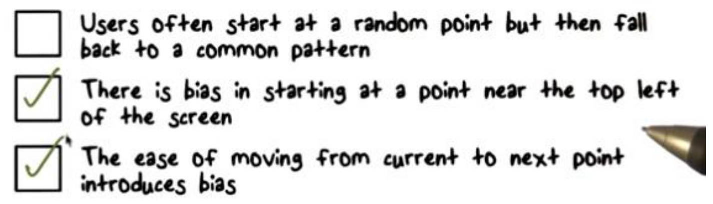
Touch Screen Passwords Quiz

In smartphone touch screens, pattern based passwords are used to unlock the device. It is believed that such patterns are not random and there is a bias in where users start. This can be explained by ...

Check the correct answer(s):

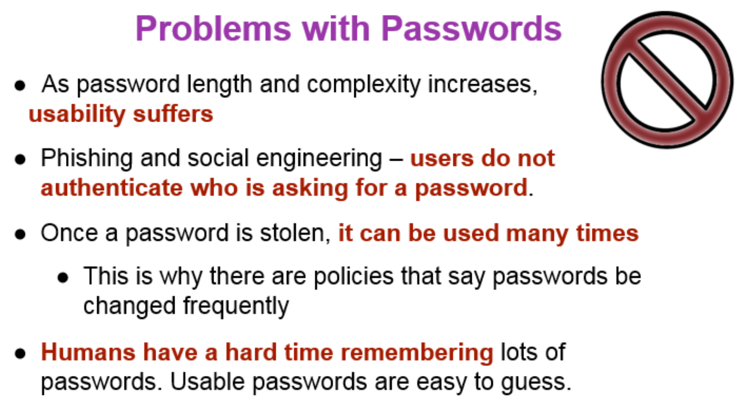
- ☐ Users often start at a random point but then fall back to a common pattern
- ☐ There is bias in starting at a point near the top left of the screen
- ☐ The ease of moving from current to next point introduces bias

The question is are these patterns random or there is some inherent bias that people have which makes certain patterns more popular? So in other words, what's the one, two, three, four, five equivalent when you're talking about this unlock patterns on touch screens? So check the correct answer or answers.



- ☐ Users often start at a random point but then fall back to a common pattern
- ✓ There is bias in starting at a point near the top left of the screen
- ✓ The ease of moving from current to next point introduces bias

Is it true that users often start at the random point but then fall back to a common pattern? Okay, actually users typically start at the left upper corner. Because stuff in English and many other languages, we go left to right. So, sort of, that bias is still there. So there is, people really don't start at a random point. I won't check that answer. There's a bias in starting at a point near the top left, okay? We just said that so we're going to check this answer and the next one is the ease of moving from current to next point. Okay, going right or down rather than diagonal in the opposite direction or something like that. The ease of moving from your current to next actually also introduces bias, an easier pattern on you and how you sort of enter it. So second and third options make those patterns less random. There are more common patterns because of ease of use reasons, because of the bias in sort of where people like to start, and things like that.



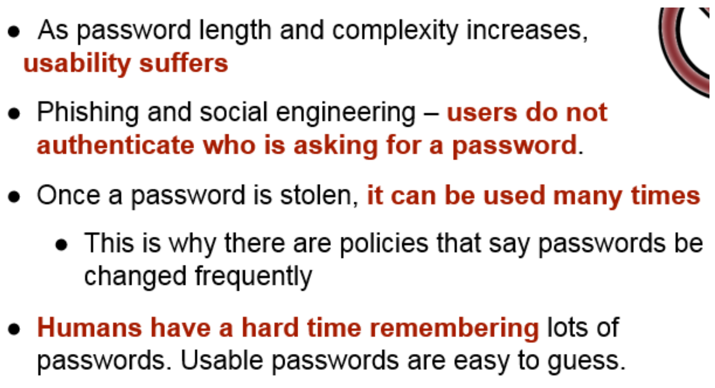
Problems with Passwords

- As password length and complexity increases, **usability suffers**
- Phishing and social engineering – **users do not authenticate who is asking for a password**.
- Once a password is stolen, **it can be used many times**
  - This is why there are policies that say passwords be changed frequently
- **Humans have a hard time remembering** lots of passwords. Usable passwords are easy to guess.

Although we use passwords almost universally, very commonly, there are plenty of problems with passwords. People been predicting the demise of passwords for a long time because of these problems, but they sort of refuse to go away. So what are some of these problems?

For passwords to be strong, and by strong we mean harder to guess, they need to be long and they need to have these upper, lower digits, special characters and things like that, so we are increasing the space of possible passwords but that increases complexity for users and usability suffers.

Even if you have good passwords, there are problems with sort of the trusted paths discussion that we had. This is sort of related to that. Phishing and social engineering attacks, really users don't authenticate who is really that their giving their password to. So this idea of mutual authentication, you trying to convince that it is really you. It's Alice. Do you assume the system is the true system? Do you ever ask the system to authenticate itself to you? In particular if you're using a remote service, your bank or something like that, how do you know it's really your bank? It could be someone else that just looks like your bank. And those are the sort of phishing attacks that we have.

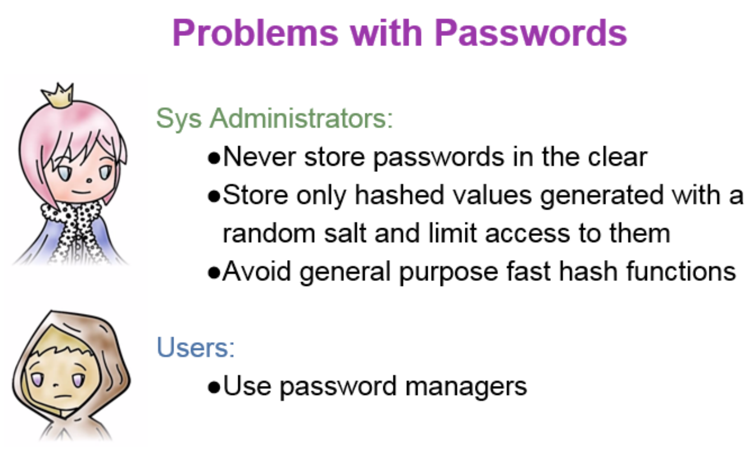


- As password length and complexity increases, **usability suffers**
- Phishing and social engineering – **users do not authenticate who is asking for a password**.
- Once a password is stolen, **it can be used many times**
  - This is why there are policies that say passwords be changed frequently
- **Humans have a hard time remembering** lots of passwords. Usable passwords are easy to guess.

Unfortunately once a password is stolen, it can be used many times. Not a one time use sort of a thing, and they can gain access to your account again and again.

And the way we address that is by having policies that say you must change your password every 90 days, or every six months, or whatever it is. Of course, usability's going to suffer because of that.

And finally, humans are humans, and they have a hard time remembering lots of passwords. These days it's not just one system that we have. You have your bank, you have your mail, you have various merchants, retailers, and things like that, so you have lots of passwords to remember. You don't want to have the same password everywhere because when one gets compromised, you're vulnerable to all your services that you access, somebody can get to them with that one stolen password. So if you're going to have different passwords, well that's going It's going to be hard for us to remember so many of those passwords if they're really hard to guess, and they're long and complex.



Problems with Passwords

Sys Administrators:

- Never store passwords in the clear
- Store only hashed values generated with a random salt and limit access to them
- Avoid general purpose fast hash functions

Users:

- Use password managers

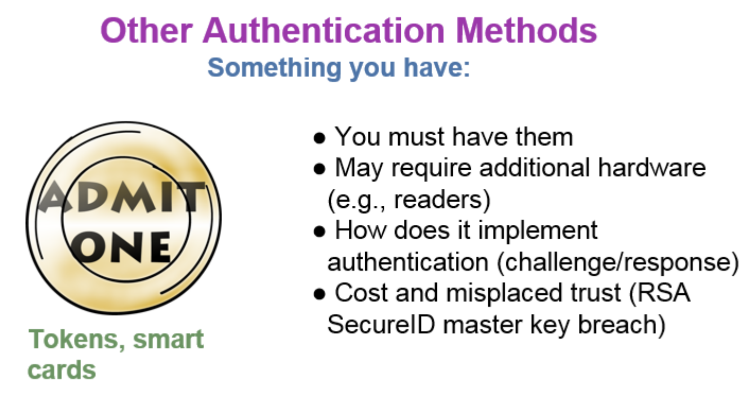
Well all of the many problems, there are of course, some best practices when it comes to passwords.

So when it comes to Sys administrators, obviously the password should never be stored in the clear. So you only store hashed values.

And we use salting, random salt, to avoid the problems, multiple users having the same password and things like that. We have to limit access to the file where these passwords are stored, the hash values are stored, so shadow files we talked about and so on.

And we want to use a function here that is not fast. We have to do it only once it's in it once the user logs in. We don't need to run it a million times. So it doesn't have to be fast, it can be slow. On the other hand, somebody doing a brute force attack has to run it again and again. They have to run it millions of times. So using a slow function actually is helpful, and current implementations actually do that. They use slower hash functions.

What users can do is, a lot of password managers these days, we're talking about having lot of passwords, one for each different service you want to use and hard to remember all of them. Password managers actually can generate complex high entropy. We didn't talk about the word entropy but entropy is sort of high entropy means it's harder to guess, so they can have these difficult to guess passwords and they can keep track of those. You basically have one strong password to access the password manager and they can keep track of your other passwords.



Other Authentication Methods

Something you have:

- You must have them
- May require additional hardware (e.g., readers)
- How does it implement authentication (challenge/response)
- Cost and misplaced trust (RSA SecureID master key breach)

Tokens, smart cards

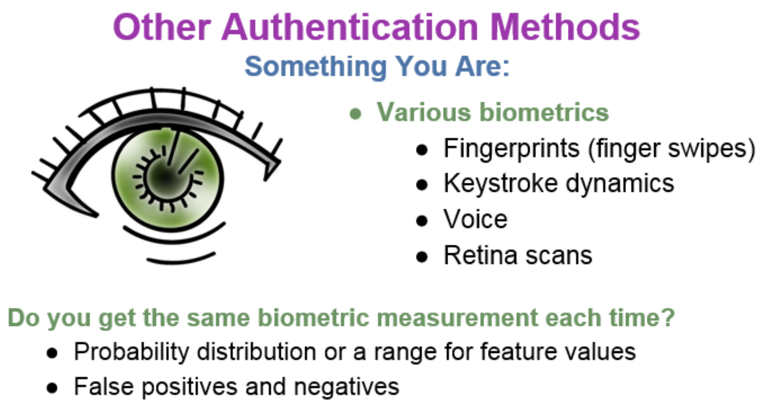
Since passwords have problems and we said that's only one of the ways in which you can do authentication, let's look at some of the other ways in which authentication can be done.

So the second one was something you have. So a token or a smart card or something like that, obviously if it some thing you have, then you must have it on them. If you don't have your token or your smart card or your smartphone, if that's been used as the something you have, you can't obviously access the system. And people sometimes forget these things and don't have them on them when they need to access especially banks, when they give you these tokens or send you some sort of a code through them. If you don't have that with you, you're not going to be able to access your account. So, not having them would cause a problem.

So you do have to have them that that's one of the requirements when you use this method. So obviously, if it's something that you must have, it should be able to talk to the system, or the human has to enter the PIN or whatever code that is sent to them. But if it's a smart card, the smart card has to talk to the system, which requires you have to have a reader, for example. You should be able to swipe the smart card and it should be able to communicate. So that adds the hardware requirement which increases cost. Even if you have it at the other end they should have a reader. If they don't have it then you can't use it with that system.

How is authentication implemented with something you have? If you have a smart card we were just talking about, then it can talk to the system, then there could be some sort of challenge/response kind of a thing. So the system can say, if you really belong to the user that is being claimed, or claim is being made about the identity of a certain user then that user's smart card has a certain secret. Use that secret to sort of maybe respond to my challenge, encrypt it or something like that, or do something to this challenge. And the response should demonstrate that the secret that is stored in this smart card is the one that is used to generate the response. So, sometimes, if you have some processing this smart card. Smart card can do this sort of computation, generate a response that could be sent. So this PIN and chip kind of cards, the chip can do this kind of work, and that's how this would we implemented.

The problem with this is that there is added cost, and of course sometimes there could be misplaced trust in this kind of tokens or cards that we're talking about for something you have. There's of course the very well-known case of the RSA breach, where the master key was compromised. And as a result of that, all the SecureID tokens that RSA stronger authentication relied on, you couldn't count on doing correct authentication using those. So there's added cost we are talking about but there's also vulnerabilities or attacks that can limit the strength that authentication, the stronger authentication that these kind of tokens provide. So, something you have, you don't need to remember those complex passwords we're talking about, but we need to be aware of the fact, and that they have their own set of problems.



Other Authentication Methods

Something You Are:

- Various biometrics
  - Fingerprints (finger swipes)
  - Keystroke dynamics
  - Voice
  - Retina scans

Do you get the same biometric measurement each time?

- Probability distribution or a range for feature values
- False positives and negatives

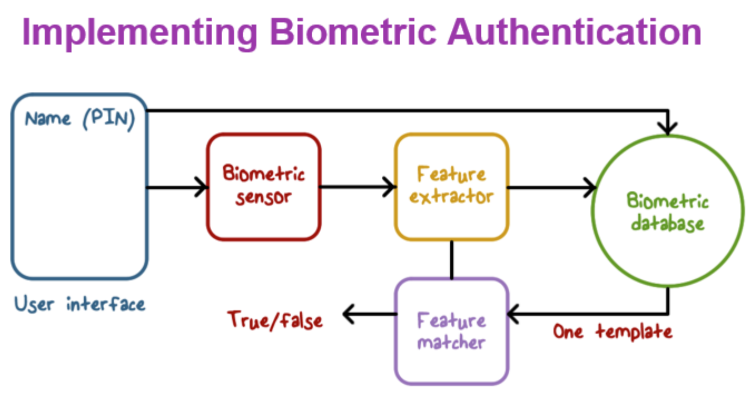
The third method we have is something you are. That's the biometric method. So, what exactly is that?

So obviously, there are a variety of biometrics and it should be unique to you. Somebody else shouldn't be able to use a biometric feature or set of features to impersonate you, so what are some examples?

Fingerprints are one. You swipe your finger for example, touch ID and things like that and they look for the pattern in your fingerprints and based on that they say it's really you. When it comes to keyboard if nothing else, the way you type, or the keystroke dynamics, how fast you are, how long you take from one key to the next depending on what the keys are and things like that. Sometimes that is used as a biometric features derived from this speed at which you type, or pattern of your typing, could be a biometric that's unique to you. Actually seeing devices that do voice biometric authentication. They extract a set of features from your voice which hopefully is unique to you, and that can be used when you talk to the device, it knows that it's really you. And, of course, they're more intrusive and fancier one like retina scan and things like that, which look for sort of unique patterns based on the blood flow in your eye, and things like that. So these are basically what you are.

The way these work, is that you're going to claim it's you and you're going to swipe your finger, or you talk in case of voice. But the biometric measurement should be the same each time. If it's different then of course we're going to have a false negative. It's you but you're not able to login because you have a cold and your voice is a little different.

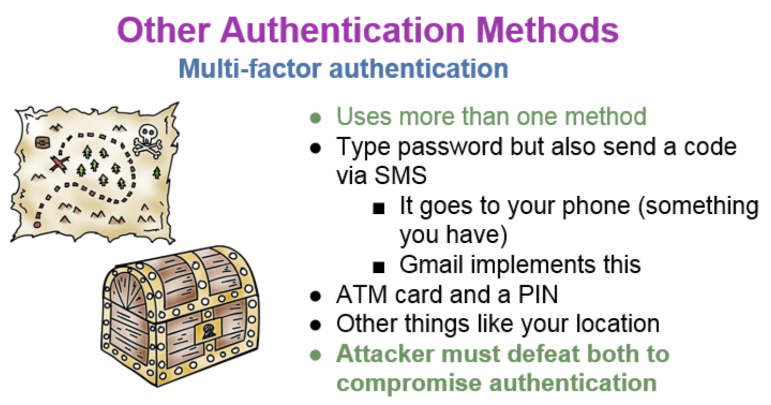
So maybe you don't have exact of these measurements. Maybe you have a probability distribution for your other features that we extract from your biometric measurements. And of course we're going to have false positives and false negatives, as we have with each authentication method that we discussed. Here, if your biometric measurements change beyond a certain limit, then of course you may not be able to login. Resulting in this false negative that we were talking about.



Implementing Biometric Authentication

So the basic way of implementing biometric is there has to be a biometric sensor, there's some user interface where you swipe your finger or camera takes a picture of you or where you talk to it, or wherever it is. That the interface, the sensor, then from the biometric reading we're going to extract a bunch of features that actually describe the reading that we just have from the sensor. And that's passed to a biometric database that stores these features, or something that derives from these features.

So PIN is here, as sort of a separate thing we may have, in addition too, PIN is something you know, so that's the secret. But say if you just sort of going with biometric, you would not have the top line here. So the biometric database seems like a hash values for passwords. We have something that describes the biometric feature set and feature values for a given user. So once you extract the feature we compare those. If there's a match between these two, then we allow authentication. If there isn't a match then we're going to say that authentication is not allowed. So the idea is that you still have the enrollment problem. Somebody has to have a description of your biometric features, which is the database where we'll be storing it. The sensor has to read it, the features have to be extracted, the comparison or match has to happen. And that's how biometric authentication works. So that when you're typing a password, we sort of reading through the interface and the sensor and then it's sort of similar to what we had before.



Other Authentication Methods

Multi-factor authentication

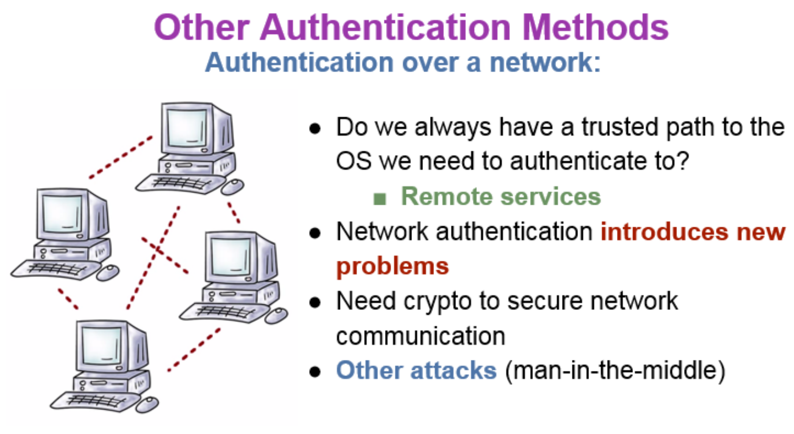
- Uses more than one method
- Type password but also send a code via SMS
  - It goes to your phone (something you have)
  - Gmail implements this
- ATM card and a PIN
- Other things like your location
- **Attacker must defeat both to compromise authentication**

We talked about these different methods, something you know, something you have, those are called factors. The first one is sort of the secret you know factor. The second is something you have factor. The third is who you are factor.

So it doesn't have to be just one factor. You can actually have a multi-factor authentication. One of the quiz questions, the smart phone and PIN for example, the password and the PIN that is sent or the code that is sent to you, the password is something you know. The PIN that is sent to you on the smart phone is based on the smart phone which is something you have, for example. So you can use more than one factor to actually make authentication stronger. So, as we said we use more than one method, you type password but also send code via SMS, and this code goes to your phone, actually Gmail when you create an account uses this. These are called phone verified accounts or PVAs, mainly to provide protection against bots that create a lot of these accounts. So this is multi-factor, because it's not just a password, but also this code that comes to a device that you have. So that's the second factor.

Your ATM card and PIN that you use to get money out of an ATM machine is actually a two factor. The PIN is something you know, the card itself is something you have. So these are two factors. There's another example of a multi-factor authentication.

The other methods that use things like where you are, your location and IP address of the machine, OS fingerprint, these sort of add a little bit of more to the authentication process. So when you have multi factor authentication of course that attacker must defeat both to successfully compromise the authentication method we have in place.



Other Authentication Methods

Authentication over a network:

- Do we always have a trusted path to the OS we need to authenticate to?
  - **Remote services**
- Network authentication **introduces new problems**
- Need crypto to secure network communication
- **Other attacks** (man-in-the-middle)

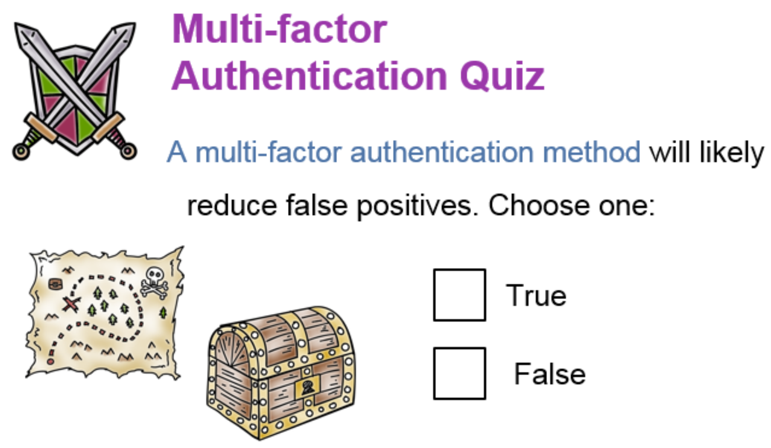
So the authentication we talked about sort of going to the system, having a trusted path to the system, providing the password, fortunately, a lot of times you do authentication over the network where the system may be thousands of miles away from you. In fact, a lot of remote services we access whether it's our email or our banking system and so on are of course, we don't walk up to them.

So, what happens in this case? We don't have a trusted path to the system, remember we sent physical wires from your keyboard to your machine right in front of you in close proximity, that's how we get that trusted path. The network is open and, obviously, the same thing doesn't hold there. So we don't have a trusted path, especially when we access in remote services.

So network authentication introduces a new problem. But how do you deal with the absence of this trusted part that we've been talking about?

Well, actually, we need crypto to secure the network communication. So when you talk about network security in a different lesson, we're going to address this.

Of course you have to worry about different kind of threats, such as man-in-the-middle and things like that. But the reason I want to mention, although we're going to cover this in network security topic, is that authentication relies on this fundamental requirement of this trusted path. In a network, you're not going to have that. So you'll have to do something else about the lack of the trusted path, or some method you have to find to secure the network over which you're going to send the evidence for authenticating yourself.



Multi-factor Authentication Quiz

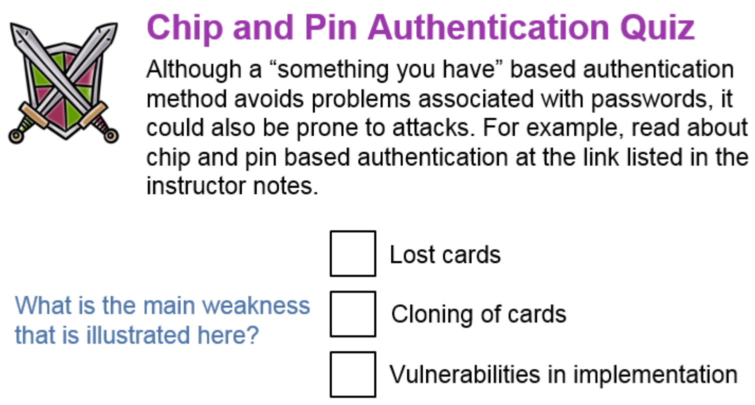
A multi-factor authentication method will likely reduce false positives. Choose one:

- ☐ True
- ☐ False

Remember multi-factor means you're using more than one factor. You could be using a card and a password or a PIN, so if you do that the first question says the authentication, multi-factor authentication method will likely reduce false positives. Is that true or false?

✓ True

So I'm going to say it's true and the reason is an attacker who is able to login as you is the one who is going to cause a false positives in it. Somebody else logs in as you, that's a false positive. For them to succeed in logging in as you, now they have to successfully attack two different factors. They have to guess your password. They have to steal your card. If they have to work harder, you increasing the complexity for them, reducing the likelihood they succeed at it, so you're going to have fewer false positives.



Chip and Pin Authentication Quiz

Although a "something you have" based authentication method avoids problems associated with passwords, it could also be prone to attacks. For example, read about chip and pin based authentication at the link listed in the instructor notes.

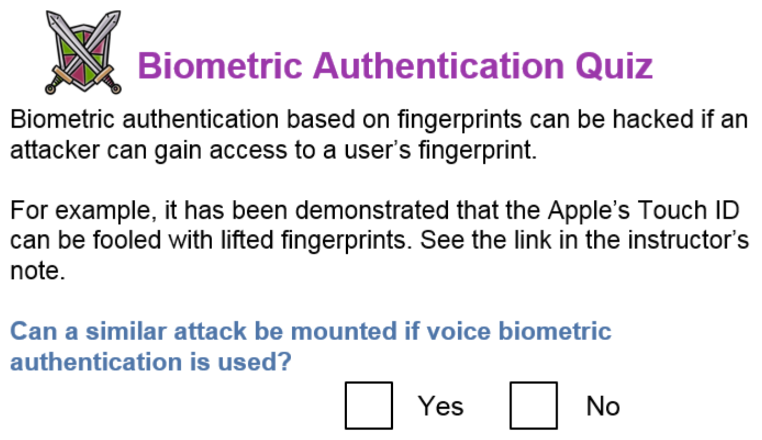
What is the main weakness that is illustrated here?

- ☐ Lost cards
- ☐ Cloning of cards
- ☐ Vulnerabilities in implementation

I want you to read about a chip and pin based authentication attack that's known, was reported by some researchers. And essentially the idea is that when you're chip and pin, there's multifactor, two factor authent, pin is something you know. Chip is in a card that you have. So it's two factor something you have and something you know. So what is the main weakness in this sort of an authentication method based on this research paper and stories that have come after it was presented?

✓ Vulnerabilities in implementation

Lost card is hopefully is not as big a problem because there's still a PIN, isn't it. So just because they have a card, you lose it, the PIN still protects you. Cloning of cards is hard because of the chip that we have is in it. And the way it works, obviously, in the old days all you had to do, the credit card number was encoded in a magnetic strip that was very easy to clone. But these chip-based cards, cloning is harder, so that's not a weakness. The third option is what this link, or the story, or the paper I mentioned. The protocol actually is implemented had some problems. Supposed to generate random numbers and answers in each transaction. Supposed to have random nonce but the implementation had some flaws, because of which, it didn't do that. And those could be exploited. So this is really an Implementation issue, in the protocol, that is based on this chip and pin cards. And that is the weakness that we are exploring in this question.



Biometric Authentication Quiz

Biometric authentication based on fingerprints can be hacked if an attacker can gain access to a user's fingerprint.

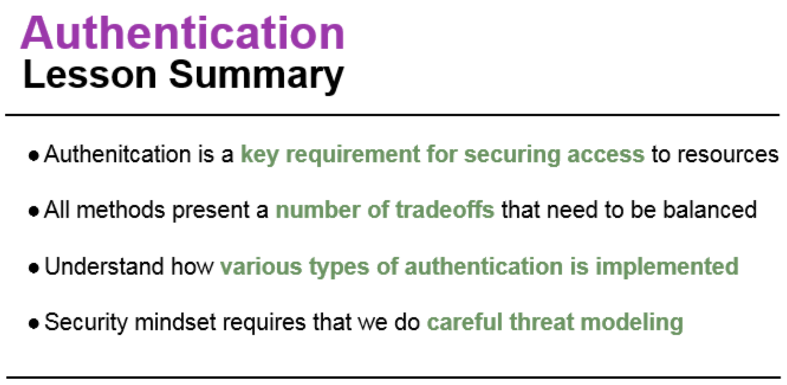
For example, it has been demonstrated that the Apple's Touch ID can be fooled with lifted fingerprints. See the link in the instructor's note.

Can a similar attack be mounted if voice biometric authentication is used?

☐ Yes ☐ No

So once you read about the Apple's Touch ID attack, question is can a similar attack be mounted when voice biometric is used.

[✓ Yes] The answer to this is, yes. They're different ways of doing voice biometrics. Either someone can record your voice and play it, that's one way, or they're also sort of voice synthesis kind of attacks people can do if they have some segment of voice from you that can build a model for you and so, similarly, these could be repeated. So just because it's something you are, in the end we're extracting a set of features from you that we make available to the system. If somebody else can grab those set of values, well, then they can fool the system into believing that is really you, and defeat the biometric indication method. So, it's interesting read how you defeat fingerprint, the touch ID that users fingerprints but similar tax people have explored on voice space authentication and so on. So yes we make hopefully we make the task harder for the attacker but there is never a full proof, completely 100% secure authentication solution.



Authentication Lesson Summary

- Authenitcation is a **key requirement for securing access** to resources
- All methods present a **number of tradeoffs** that need to be balanced
- Understand how **various types of authentication is implemented**
- Security mindset requires that we do **careful threat modeling**

We studied a number of different authentication methods and their implementation in a computer system. Our security mindset approach requires that we consider any threats that may exist against these methods and their implementations. So, we did consider the strengths and weaknesses that exist for each of these methods and how we use them in computer systems.