



Intro to Cybersecurity

Foundations and Threats

1.1.4 - Password Hashing

How are passwords stored on a system and what is password hashing?

Overview

The student will be able to:

- Recognize authentication vocabulary terms
- Understand methods of secure password storage
- Define has as a method of one-way conversion

Grade Level(s)

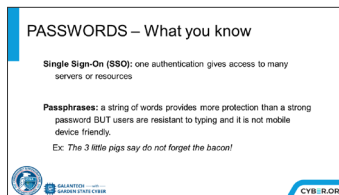
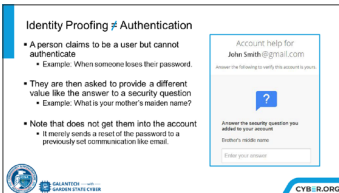
6, 7, 8, 9, 10, 11, 12

Cyber Connections

- Access Control
- Authentication

This content is based upon work supported by the US Department of Homeland Security's Cybersecurity & Infrastructure Security Agency under the Cybersecurity Education Training and Assistance Program (CETAP).

Teacher Notes:



Password Hashing

Slide 1 - Intro Slide

Slide 2 - Identity Proofing = Authentication

When you have forgotten your password and need to be given access to the reset option, you first must identify yourself as being the correct user. This is called Identity Proofing and it is NOT the same as Authentication. Why? Walk through the steps in the slide above and you will see that when you are done with Identity Proofing, you still are not able to log into the Gmail account. At the end of identify proofing the vendor has sent a reset link to the phone or alternate email that you have listed in your security options – if you are a hacker and answered the identity proofing questions correctly, it won't matter because you don't have access to my phone so your hack will stop there.

Slide 3 - PASSWORDS - what you know

Terms about passwords:



Single Sign-On = this is used in organizations to make it easier for users to get access to all different kinds of data. It used to be that you logged onto your computer, then if you want a file from a shared drive you would need to authenticate there and then to get on the Internet you would need to authenticate with the Proxy filter. By configuring Single Sign-on, the organization only requires you to log onto your computer and then it keeps track of what data you can access without challenging you anymore for authentication repeats.

Passphrase = Complicated passwords like 14t\$24tS32gR are hard for users to remember. Instead, using a string of words or sentence (aka passphrase) offers the same or better security AND it would be easier to remember. But it needs to be long enough which is a problem because users really don't like to type and passphrases are quite difficult to use on a mobile device like a phone. The example above is very easy to remember and even includes a number, special character and upper/lower case letters. But I don't want to type that several times a day or try it on my phone.

Teacher Notes:

Where are Passwords stored?

- Windows machine: SAM registry hive holds all account details like usernames and passwords
- Linux: store usernames, passwords, some settings
 - `/etc/passwd` = default file, not used for passwords
 - Not secure, can be hacked because more than `root` can access this file.
 - `/etc/shadow` = hidden file used to store usernames, passwords, some settings
 - ONLY `root` user has access





Password Storage & Hashing

- Passwords should never be stored in plain text
- Most common method of safely storing passwords is to HASH the password



Hashing – what is it??

Definition: a special mathematical function that performs one-way conversion.



Hashing

- Hashing is intended as one-way conversion. Once the algorithm is processed, the hashed password is stored by the system.
- Authentication happens when a hashed password is compared against the original hash stored by the system – this will confirm that you have the correct original plaintext.
- MD5, SHA256, and SHA512 are commonly used hashing algorithms



Slide 4 - Where are Passwords Stored?

When you authenticate, you provide your username and password. The system checks to see if that is the same information that it has stored. Did you ever wonder where that is stored? If it is on a local computer, then it is stored in the SAM hive for Windows and the `/etc/shadow` file for Linux/Unix/Mac. A newbie user might believe that the `/etc/passwd` file holds password data – but that file hasn't been used for password storage for a very long time.

Modern versions of Linux do not store passwords in the `/etc/passwd` file.

Slide 5 - Password Storage & Hashing

One important point is that your password should NEVER be stored in plain text. Most operating systems, networks and websites will store the password in an unreadable form called a HASH. Here's how this works:

1. you click to create a new account
2. you are asked for a username
3. you are prompted to create a password – you type your password in and click submit
4. the system applies an algorithm to your password to turn it into a hash (more about this is in a minute)
5. your username and hashed password are put in system storage or they are transferred across the network to be put in network storage.

Slide 6 - Hashing

Point out that no one in a network has ever seen your password - and they can't access it from a database. If you forget your password and you try calling the network admin at your school to get it, the admin CAN'T - the only thing they can do is reset it so you can make a new one.

This is because only password hashes are stored in the system.

Let's look at how hashing works in a little more detail.


Slide 7 - A simplified version of Hashing

The example above is just a very simplified example of how hashing works. The key is to prove that a hash is not an encryption of the password that could then be decrypted. Hashes are conversions of text to a string of hex characters and there is no key that would make it reversible.

Teacher Notes:

A simplified version of Hashing

- Password: *this*
- ASCII Values $t = 116, h = 104, i = 105, s = 115$
- These values are multiplied by 2 to get the calculated numbers, which would be 232, 208, 210, 230.
- These numbers are added together then divided by 10 → $(232+208+210+230)/10 = 88$
- This gives you a hash of 88.
- But there are other number/letter combinations that would produce this hash.
- The actual hashing algorithms are much more complex and do much more than just performing arithmetic operations on relatively small numbers. It is extremely difficult to determine the data that was hashed, even if you know the algorithm used to generate the hash.

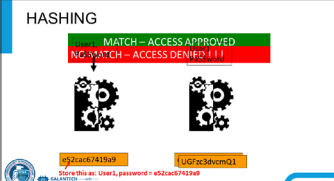


CYBER.ORG


Walkthrough of slide:

1. My password is “this”
2. When I type that into the computer it translates each of the password letters into the ASCII values.
3. Then the hashing algorithm I am applying takes those decimal numbers and multiply each by 2. This gives me the new numbers 232, 208, 210 and 230.
4. Then my algorithm says we take those new numbers, add them up and divide the sum by 10. My hashing algorithm is complete so the hash of my password = 88.
5. If you discover that my password hash is 88, that doesn’t help you in any way BECAUSE there are lots of ways to arrive at the number 88 – you could do $4 \times 22, 8 \times 11$ and many more complicated ways. So just because you know my hash doesn’t mean you have any chance of knowing my password. And that’s the point – we are essentially storing our passwords using an encoding system that can’t be reversed.

HASHING



Store this as: User1,password = e52cac67419a9



CYBER.ORG

Slide 8

Work through this animation to show the process for authentication with hashes.

Each bullet is a click:

- You submit your username (User1) and password (Pa\$\$word) – Pa\$\$word goes through the hashing algorithm and results in the hash e52cac67419a9
- The system understands that it should store the username and hash together
- An attacker tries to log in with your username and the wrong password (FakePass) – FakePass goes through the hashing algorithm and results in the hash UGFzc3dvcmQ1 – that does NOT match what is stored for User1 so access is denied.
- You log in with your username and password (Pa\$\$word) – since the resulting hash matches what is in storage, your access is approved.

Teacher Notes:

Attacking Hashed Passwords

- **Define term: Hash collision** - when two or more source data convert to the same hash value.
- **Birthday Attack** - in a group of >22 people, the chance that two people share a birthday is greater than 50%.
- This attack takes advantage of the fact that when collecting a large number of hashes, it is likely that there will be multiple data items (like passwords) that have the same hash value.
- With a "hash collision" collection of data, you could reverse engineer the key used for hashing.



GALANTECH
GARDEN STATE CYBER

CYBER.ORG

Slide 9 - Attacking Hashed Passwords with Rainbow Tables

Hashing was a wonderful security innovation to improve upon storing plaintext passwords. But unfortunately, hashing is not immune to attack. There are 3 key ways to attack hashed passwords.

#1 Rainbow Tables - as passwords became longer and more complex, the dictionary and brute force attacks became too time consuming. So someone came up with the bright idea of "pre-staging" an attack by taking an algorithm and hashing every word in the dictionaries and every possible combination of characters. That all compiles into a HUGE file called a Rainbow table. The attacker takes your hashed password and then using specialized software to compare it to every hash in the Rainbow table. This is the sort of function that computers do VERY well and VERY quickly so the Rainbow table attack quickly became a favorite password attack. The only downside to performing a Rainbow table attack is that the file is actually so big it was difficult to download.

But fortunately, security professionals came up with an effective way to stop a Rainbow table attack. It's called "adding a salt." The intent is that even if two passwords are the same, because we add a unique set of characters every time before we apply the algorithm, the result will be two different hashes. See example in slide above but even better is the explanation given in the video. Video is 4:53 min. Video is in lesson folder as an mp4.

So, rainbow tables are not much of a threat anymore because best practice is that all passwords are hashed with **randomized** salts. Unless you know the hashing algorithm used AND you know the exact salt used for that password, you are out of luck.

<https://youtu.be/8ZtInClXe1Q?t=328> stop at 9:04

Slide 10 - Attacking Hashed Passwords

#2 The Birthday attack is based on a very cool mathematical rule called the Birthday Paradox which says that in a group of at least 23 people, there is a better than 50% chance that two people will have the same day/month. When the group size reaches 70%, there is a 99 % chance of two people sharing a birthday. Take a minute to survey the room and see if there are any birthday duplicates - if you have a large class the answer is probably yes!

The Birthday Attack is similar - it says that for any hashing algorithm there is a mathematical likelihood that two different passwords will come out with the same hash. When that happens it is called a "hash collision" and it can be used to determine what hashing algorithm was used - and once I know that, I can reverse the hash to plaintext.

Attacking Hashed Passwords

- **Pass the Hash attack** - this attack doesn't try to crack the password. Instead, the attacker gets to the SAM file or etc/shadow file to dump the hashes from system storage.
- Using special software, the attacker logs in with the username and password hash instead of the text password.



GALANTECH
GARDEN STATE CYBER

CYBER.ORG

Teacher Notes:

Attacking Hashed Passwords with Rainbow Tables

- Rainbow Tables - Definition: a file of pre-computed hash values for every possible combination of characters. **VERY BIG FILE!!**
- SALT = method of protecting against Rainbow Table hacks.
- Normal password storage:
 $md5 \times (\text{password}) = \text{hash of } 544dc35b9af765de108327d$
- Salted password storage:
 $md5 \times (\text{randomstring} + \text{password}) = f6685bc394c7ec1140eac5ff$



STOP Video at 5:24 min. **CYBER.ORG**

Intro to Cybersecurity

Activity – Hashing and Salts with CyberChef



CYBER.ORG

Remember, I said that hashing isn't intended to be reversed – it's supposed to be a one-way conversion. But that doesn't mean it's impossible to reverse!

Slide 11 - Attacking Hashed Passwords with Rainbow Tables

#3 Pass the Hash attack - since the system we are trying to log into never actually receives the plaintext password, some smart attackers realized that they don't need to reverse the hash to plaintext. They don't actually care what your password is, they just want to log in as you.

That's how the Pass the Hash attack was created. They created some specialized software so that the system would skip the few steps where the user puts in his plaintext password and just go straight to the part where the username/hash are accepted.

Slide 12 - ACTIVITY

This is a fairly quick activity where students work with a partner to prove that adding a salt to a password will protect against Rainbow Attacks. Students will use the CyberChef app which is available through a browser. This is a secure tool will be used for other activities throughout the course. Provide students with the link to the CyberChef introductory video or alternatively you can show the video to the class before students start this lab. Video: <https://vimeo.com/588599799> (7:32 min).