With funding from

Password Process

Participants Perspective

Today I learned how to protect my information online **by** layering complexity into a password and **by** understanding how algorithms use passwords to secure data.

Learning Goal

- Understand how layering complexity in passwords makes them better at protecting personal data.
- Understand how passwords can be used to secure data by using and creating simple forms of hashing.

Logistics

- Suggested length: 1-2 hours
- Age range: Grade 7 +
- Additional requirements: Participants will need rudimentary math skills (+,-,X,÷) and knowledge of the alphabet. Basic understanding of the term "function" in programming.

Part Number	Section Title	Group Size	Materials
Opening Hook	Worst Password	Whole group	● "Worst Passwords of 2016" List - attached ● Whiteboard or flip paper ● Markers
Part 1	Password building	Individuals, then partners. Whole group discussion	● Whiteboard or flip paper ● Markers ● Paper & pencils ● Photocopies of attached Password Scoring table (enough for each pair or small group)
Part 2	2Hard2Hack	Partners	● Method of showing a video (Youtube) to class ● Paper & pencils ● Photocopies of attached Rudimentary Hash Function (enough for each pair or small group)
Part 3	Closing discussion	Whole group	● Method of showing a video (Youtube) to class

Safety Considerations

- This unplugged activity is a low risk activity based on limited exposure to hazards beyond the standard classroom.
- **Passwords**: Students will be discussing password safety and should not share any actual passwords with other participants. All passwords should be hypothetical and should not be used after the lesson is completed.
- **YouTube**: Participants may come across materials that is not appropriate (including but not limited to violent or sexual images, racist/sexist commentary, and so on). Pre-plan the YouTube viewing that will occur. We suggest using websites like TubeChop to clip YouTube videos to remove ads, and avoid the "suggested viewing" and comment sections of the usual YouTube sites.

Framework Connections

Toolsets	Skillsets	Mindsets
Knowledge Developing an understanding of password composition and the elements that make a password secure. Understanding how passwords are hashed and stored to make a secure method of password verification.	**Digital Skills** Data management and Safety Algorithm development-building a hash function	**Digital Intelligence** Digital safety: By understanding how passwords are used, participants develop a better ability to secure their data online. Understanding the implementation of hashing functions to protect data.
Resources Password strength scoring table. Development of own simple hashing function.	**STEM Skills** Problem solving - Reverse engineering of others hash value Algorithm development and implementation	**Computational Thinking** Understanding cause and effect relationships between password complexity, mathematical hash operations and computational efficiency.
Experiences Compete against others to create a secure but useable password. Computational thinking - Using simple hashing functions and developing own hashing function to understand how password security processes work.	**Essential Employability and Life Skills** Security awareness developing a community of safety Understanding of cause and effect relationships	**Digital Action** Awareness of simple processes that can be used to secure data. Spread awareness of best practices for password design Digital security implementation

Nuts and Bolts

Hook: Worst Password

- Have participants take a minute to consider the worst possible password (easiest to guess) that they can think of (not one they actually use hopefully). Once they each have an example or two have them write them on a whiteboard or flip paper so that all the examples can be shared with the group.

- Discuss as a group what aspects of these passwords make them bad? Length? Simplicity? This will help participants to begin to consider what makes a good or a bad password. How does commonality or composition contribute to this?
- Share with the participants the list of "Worst Passwords of 2016", included below. Note how all (except #18) of the top 20 are simple words or sequences of numbers with very few layers of complexity. Modern computers can attempt > 2 billion passwords a second, that is why building more complexity into the password is important.
- Ask the group to brainstorm possible ways to add in complexity to the password. Write the participants' ideas on the whiteboard to document them. Fill in the gaps with some common methods that include:
    - Mixture of upper and lower case letters
    - Numbers
    - Symbols
    - Increasing the length, while remaining something that is memorable
    - Significant complexity comes when you layer all 4 of the above with:
        - No consecutive letters, numbers or symbols (fff, GGG, @@@, 333)
        - No sequential letters, numbers or symbols (bcd, 3456, !@#$%)
        - No personal references (people, animals, birthdays or places)
        - A passphrase with a number of words interspersed with numbers and symbols

Part 1: Password building

- Participants should now take a few minutes to create a new password, one they would like to share with others, to participate in the password challenge competition. This actively allows them to apply the discussion into practice.
    - Note that students should not use any passwords that they actually use and passwords created today should not be used in the future.
- Once they are ready, they should write this password on a slip of paper and exchange with their partner. With their partner's password, each participant now scores the new password based on the attached Password Scoring Table. This Scoring Table evaluates the strength of the password to give it a score based on complexity and use of various features.
- Once completed participants should write their password and score on the whiteboard for discussion:
    - Were you surprised by your score and where it fell in terms of the rest of the participants?
    - What made the "best" password for the group so good?
    - What makes a password something you can remember?
- Give the participants a minute or two to re-work their password. Can they alter it easily to get a better score? This gives participants another opportunity to apply their knowledge and improve their skill set.

Part 2: 2Hard2Hack

- Ask the participants to consider the WHY? Why do these features make a good password? What makes a password hard to hack? What is even done with the passwords once they are entered into a website? Once the participants have described some ideas, introduce the concept of hashing. This can be done using either a video describing the process or verbally:
    - Hashing Video: https://www.youtube.com/watch?v=KFPkmhcSlo4, OR
    - Use the attached Hashing introduction
- Have the participants try using a rudimentary hash function using the attached Hashing Function sheet. Participants will use their newly created passwords and hash them using the hashing table included. This process will show them how a hash function could process inputs. This builds towards an understanding of the need for hashing functions, why they are different and how they can work to change a password.
- Discuss how this rudimentary hash function has issues with collisions. This is when the hash function produces the same hash for 2 (or more) different inputs. The simpler the hash function is the more likely it is that this will happen. Ideally a hash function can produce more characters in its output than characters that were put in. This can be done in a number of ways, including introducing math operations.
- Once the participants have used the Hashing function sheet, it is time for them to build their own hashing function. Working with a partner construct a hashing function to protect passwords. Consider how the function might deal with:
    - Various lengths of passwords? Long passwords
    - Different character types - UPPER case, lower case, symbols, numbers, spaces?
    - Repeating letters,numbers or symbols
    - Similar passwords, such as 'Dog' and 'Bog'
    - How can you reduce the number of collisions your hashing function makes? Consider adding in some mathematical functions that calculate a number to replace a character with.
    - Test out the hash function using the following word list:
        - Cat, Hat, Car, PASSword, PA$$worD
- Each participant should now use their hash function to process a new secret password. Partners then exchange their hashed values and try to decipher it knowing how the hash function works. This is key to seeing how difficult a hashing functions (even simple ones) are to break through. Most participants will struggle to find the original input password.

Part 3: Closing Discussion

- Discuss how well this process worked.
    - Where you successful at deciphering the hash of your partner?
    - If you receive someone elses hash and did not know the hash function, would you be able to determine the original password?
    - Use of computers allows billions of combinations a second, would your hashing function stand up to this type of attack? What changes might you need to make?
- Describe the drawbacks of using simple hash functions:
    - Collisions are the biggest issue. The simpler the hash function is the more likely it is that this will happen and the more likely that the security of the data can be compromised . As a result professional security hashes use very long and very

complicated hashing functions. As these are computed by programs and not humans, they can be processed very quickly, despite the complexity.
    - Additionally, it is desirable for hash functions to be slow to process. The average user is not going to notice a half second wait when then enter their password during login, but if a hacker is trying 2 billion passwords a second extending each one by a half a second makes the process a lot longer.
- Close the activity with a summary video (if possible). This video is an extension of the first and describes how hashes can be made more secure, leading to hackers looking for other ways to attempt to steal passwords. https://www.youtube.com/watch?v=RtUvMJFP_IE

Modifications and Extensions

Ways to make this activity more approachable:

- Share the Password scoring table with participants before the activity.
- Work through the rudimentary hashing function with the participants.
- Work in groups of 4 instead of groups of 2.
- Instead of designing their own hashing function, participants can improve the rudimentary one to reduce the number of collisions. This could be done as a group activity.

Ways to make this activity more challenging:

- Limit password length to 10 characters for the password competition.
- Have participants swap hashing functions and try to “break them” by finding collisions.
- Learn more about ciphering and how hashing can be further secured using encryption. See Actua lesson - Codemakers Level up with CS

Assessment and Evaluation

- Scoring of the passwords gives participants an opportunity to compare how they have been able to apply the information versus other participants.
- Self evaluation of skills in layered in as participants can compare their hash values to others
- Working in partnerships helps to self-evaluate level of understanding

Credits, Kudos, Shoutouts

- Discovery News - For good summaries of the hashing process:
    - https://www.youtube.com/watch?v=RtUvMJFP_IE
    - https://www.youtube.com/watch?v=KFPkmhcSlo4
- TeamID/SplashID for developing listing and image of “Worst Passwords of 2016”:
    - https://www.teamsid.com/worst-passwords-2016/?nabe=4561770576609280:1,5716650381017088:0,5767892520140800:2
- Password scoring webpage: http://www.passwordmeter.com/

Terms of Use

Prior to using this activity or parts thereof, you agree and understand that:

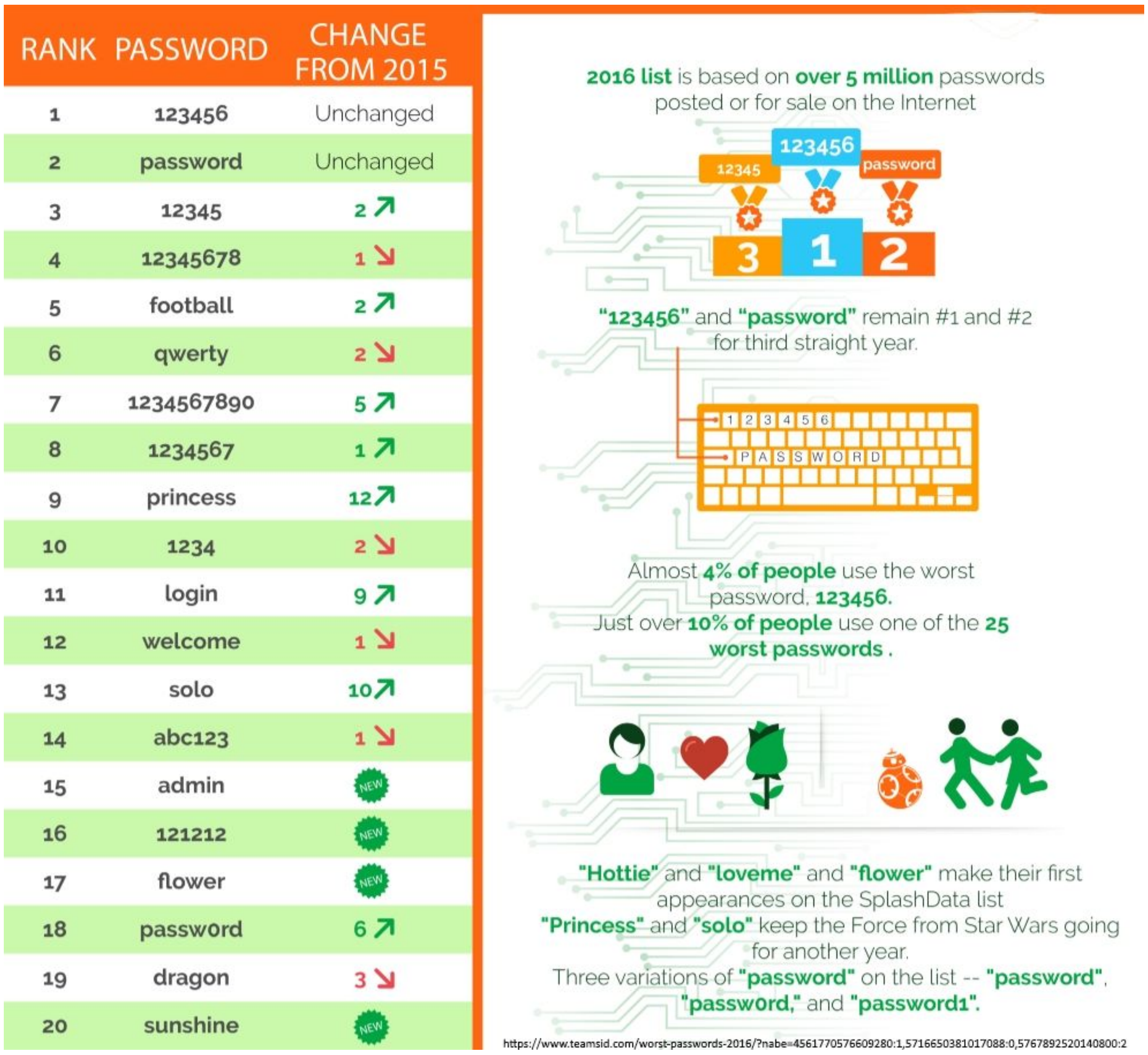
- It is your responsibility to review all aspects of this activity and ensure safety measures are in place for the protection of all involved parties.

About Actua

Actua is Canada's leading science, technology, engineering and mathematics (STEM) youth outreach network representing a growing network of university and college based members. Each year 250,000 young Canadians in over 500 communities nationwide are inspired through hands-on educational workshops, camps and community outreach initiatives. Actua focuses on the engagement of underrepresented youth through specialized programs for Indigenous youth, girls and young women, at-risk youth and youth living in Northern and remote communities. For more information, please visit us online at www.actua.ca and on social media: Twitter, Facebook, Instagram and YouTube!

With funding from

Worst Passwords of 2016:



RANK	PASSWORD	CHANGE FROM 2015
1	123456	Unchanged
2	password	Unchanged
3	12345	2 ↗
4	12345678	1 ↘
5	football	2 ↗
6	qwerty	2 ↘
7	1234567890	5 ↗
8	1234567	1 ↗
9	princess	12 ↗
10	1234	2 ↘
11	login	9 ↗
12	welcome	1 ↘
13	solo	10 ↗
14	abc123	1 ↘
15	admin	NEW
16	121212	NEW
17	flower	NEW
18	passw0rd	6 ↗
19	dragon	3 ↘
20	sunshine	NEW

2016 list is based on **over 5 million** passwords posted or for sale on the Internet

12345
123456
password
3
1
2
1 2 3 4 5 6
P A S S W O R D

"123456" and **"password"** remain #1 and #2 for third straight year.

Almost **4% of people** use the worst password, **123456.**
Just over **10% of people** use one of the **25 worst passwords**.

"Hottie" and **"loveme"** and **"flower"** make their first appearances on the SplashData list
"Princess" and **"solo"** keep the Force from Star Wars going for another year.
Three variations of **"password"** on the list -- **"password"**, **"passw0rd,"** and **"password1"**.

https://www.teamsid.com/worst-passwords-2016/?nabe=4561770576609280:1,5716650381017088:0,5767892520140800:2

Partial list and image from TeamID/SplashID "Worst Passwords of 2016". Original can be found on the Teamsid webpage:

- https://www.teamsid.com/worst-passwords-2016/?nabe=4561770576609280:1,5716650381017088:0,5767892520140800:2

With funding from

Password Scoring Table		
Item	**Scoring Algorithm**	**Score**
# of Characters used for password - This number is N and is used for other items below	If Nis more than 8, Score = N, OR If N is less than 8, Score = N x 4	
If both UPPER and lower case letters are used	$\frac{N}{2}$ + 2	+
If more than 1 numbers are used	$\frac{N}{2}$ + 4	+
If more than 1 symbols are used	$\frac{N}{2}$ + 6	+
Numbers AND symbols used in the middle of the password (not just first and last characters)	$\frac{N}{2}$ + 4	+
Memorable string, Not random, not too long, (markers choice)	N x 2	+
Repeat characters (i.e. aa, 444, %%%%)	(# repeated) x 2	-
Sequential characters (i.e. abc, 123456, !@#$%)	(# repeated) x 2	-
Password uses names, pets, dates or other known personal information of the creator, or is common (123456, password...)	N x 2	-
	TOTAL:	SUM

EXAMPLE Scoring

EXAMPLE Password Scoring Table for “Bunny3Hug$%^”		
Item	**Scoring Algorithm**	**Score**
# of Characters used for password	If Nis more than 8, Score = N, OR If N is less than 8, Score = N x 4	**44**
If both UPPER and lower case letters are used	$\frac{N}{2}$ + 2	**+7.5**
If more than 1 numbers are used	$\frac{N}{2}$ + 4	**+0**
If more than 1 symbols are used	$\frac{N}{2}$ + 6	**+11.5**
Numbers AND symbols used in the middle of the password (not just first and last characters)	$\frac{N}{2}$ + 4	**+0**
Memorable string, Not random, not too long (markers choice)	N	**+11**
Repeat characters (i.e. aa, 444, %%%%)	(# repeated) x 2	**-4**
Sequential characters (i.e. abc, 123456, !@#$%)	(# repeated) x 2	**-6**
Password uses names, pets, dates or other known personal information of the creator ,or is common	N x 2	**-0**
	TOTAL:	**64**

Other Examples:

“Frank1112345” - 48 + 8 + 10 + 0 + 0 + 0 - 6 - 10 - 24 = 26
“H4P&y1” - 6 + 5 + 7 + 0 + 7 + 6 - 0 - 0 - 0 = 31
“123456” - 6 + 0 + 5 + 0 + 0 + 6 - 0 - 12 - 12 = - 7

Hashing Introduction:

Hashing is a computer process that uses a function* to secretly convert data in one form into another form. A hashing function will take your password and convert it into a sequence of letters, number and symbols that look nothing like the original password. The resulting sequence of letters, numbers and symbols is called a hash. You can think of a hash as a code name for your real password.

Hashes are nice because they don't have to be protected like your password. You can say a code name out loud and people around you don't know what the code name stands for. The hash can be sent publicly from one place to another without someone being able to see it and know exactly what your password.

A few important details about hashing functions:

1. Hashing functions should always creates the **same hash for the same input.** Everytime you put in your password, the same hash comes out.
2. Proper hashing functions will make very **different hashes for 2 similar words**. Cat, Cats, bat & bats should all look very different when hashed.
3. Hash functions typically **create an output of a set length, regardless of the input length**. A password of 5 letters and a password of 15 letters should both have hashes that are the same size. Lots of hashing function outputs are 64, 128 or 256 characters in length.
4. The hash is typically **not human readable and not easily decipherable.** Even with lots of hashes to look at, it is hard to reverse engineer what the function is doing, making it difficult to discover what the original password is.
5. When a new password is created, the hash function is used and **only hashed passwords are stored** in a table associated with that users name. When you enter your password, the app or website immediately (and privately) hashes your password and compares it to the hash for your user name. This means that hackers intercepting this operation or with access to the user tables, only have the hashed form and do not know the original input.

function* - A function is a section of computer code that performs a specific task. A function will typically have both inputs and outputs. These can be numbers, letters, human input, variables or other specific information to help the function complete its task. The inputs will enter into the function, the function will complete a number of steps based off of the input and then return an output.

With funding from

Rudimentary Hashing Function:

1. Use the following table to find the value of each character in the password:

Character	Hash value	Character	Hash value	Character	Hash value
b,c,d	1	i,o,u	9	X,Y,Z	&
f,g,h	2	" "(space) or blank	0	A,E	*
j,k,l	3	B,C,D	!	I,O,U	(
m,n,p	4	F,G,H	@	1,2,3,4,5	Q
q,r,s	5	J,K,L	#	6,7,8,9,0	W
t,v,w	6	M,N,P	$	First symbol	E
x,y,z	7	Q,R,S	%	Second symbol	R
a,e	8	T,V,W	^	Further symbols	T

2. Place the value in the order of the characters in the password, if the password is longer than 8 characters, ignore the last characters. While this is poor in practice, it makes for quicker process times:

	Hashing function output							
Password:	**C1**	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**	**C8**
"Cat1"	**!**	**8**	**6**	**Q**	**0**	**0**	**0**	**0**
"Car1"	**!**	**8**	**5**	**Q**	**0**	**0**	**0**	**0**
"B4$ke77"	**!**	**Q**	**E**	**3**	**8**	**W**	**W**	**0**
"Mikki$$$"	**$**	**9**	**3**	**3**	**9**	**E**	**R**	**T**

3. Note that this hashing function is very simple and it is easy for it to produce the same output for different inputs (hash(Baseball7) = hash (Derecell9)). This is called a collision and should be avoided, typically by making the hashing function more complex.