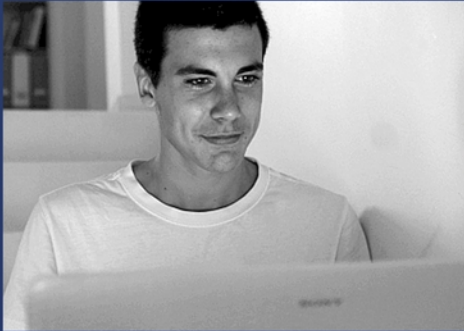


Hacker HighSchool

SECURITY AWARENESS FOR TEENS



LESSON 11 HACKING PASSWORDS



HACKING IS LEARNING
www.hackerhighschool.org

ISECOM

Creative Commons 3.3 Attribution-Non-Commercial-NoDerivs ISECOM

WWW.ISECOM.ORG - WWW.OSSTMM.ORG - WWW.HACKERHIGHSCHOOL.ORG - WWW.BADPEOPLEPROJECT.ORG - WWW.OSSTMMTRAINING.ORG



WARNING

The Hacker Highschool Project is a learning tool and as with any learning tool there are dangers. Some lessons, if abused, may result in physical injury. Some additional dangers may also exist where there is not enough research on possible effects of emanations from particular technologies. Students using these lessons should be supervised yet encouraged to learn, try, and do. However ISECOM cannot accept responsibility for how any information herein is abused.

The following lessons and workbooks are open and publicly available under the following terms and conditions of ISECOM:

All works in the Hacker Highschool Project are provided for non-commercial use with elementary school students, junior high school students, and high school students whether in a public institution, private institution, or a part of home-schooling. These materials may not be reproduced for sale in any form. The provision of any class, course, training, or camp with these materials for which a fee is charged is expressly forbidden without a license, including college classes, university classes, trade-school classes, summer or computer camps, and similar. To purchase a license, visit the LICENSE section of the HHS web page at <http://www.hackerhighschool.org/licensing.html>.

The Hacker Highschool Project Project is an open community effort and if you find value in this project, we ask that you support us through the purchase of a license, a donation, or sponsorship.



Table of Contents

WARNING.....	2
Introduction.....	5
Pins, Passwords and Personal Poop.....	6
I Don't Get It.....	6
Feed Your Head: CAPTCHA and Passwords.....	7
Methods of Authentication: Passwords.....	7
Strings of Characters: Something You Know.....	8
Strings of Characters Plus a Token: Something You Know plus Something You Have.....	8
Biometric Authentication: Something You Are.....	8
Game On: Hungry for Knowledge.....	9
Password Biology: SWAT.....	11
Strengths.....	11
Weaknesses.....	12
Advantages.....	13
Threats.....	13
Human Biology.....	16
Memory.....	16
Bad Habits.....	16
We Want Change.....	17
Feed Your Head: Bad Passwords Are Your Problem.....	18
Hashed Passwords and Salts.....	18
Hashes (MD5, SHA-1, SHA-2 and SHA-3).....	18
Salt.....	19
Key Derivation Functions.....	19
Super Duper Complex Cryptography.....	20
Password Cracking.....	21
Use the (Brute) Force.....	21
Dictionary attacks.....	22
Somewhere Over the Rainbow.....	23
Network-based Password Cracking.....	25
Wireless Password Cracking.....	29
Cracked Up.....	30
The Soft Side.....	31
Build a Great Password.....	31
Feed Your Head: Cor Issues.....	32
Size Matters.....	33
Other Methods.....	33
Check, Please.....	34
Change Up.....	35
Conclusion.....	36





Contributors

Marta Barceló, ISECOM

Pete Herzog, ISECOM

Kim Truett, ISECOM

Chuck Truett, ISECOM

Jaume Abella, ISECOM

Greg Playle, ISECOM

Bob Monroe, ISECOM

Cor Rosielle, ISECOM

J. Agustín Zaballos

Mario Platt

Mark D'Angelo

Simone Onofri

Marco Ivaldi, ISECOM

The logo for ISECOM, with 'ISEC' in dark blue, 'O' in yellow, and 'OM' in dark blue.



Introduction

To enter your home, you need a key for the locked door (unless you can pass through walls, which would be really cool). If you have a locker at school or in gym class, you either have a key lock or have a number combination lock to get inside the locker. If you want books from your library, you have to have a library card to prove who you are. You need drivers licenses, school IDs, and lunch passes to prove you are who you say you are and that you've got a lunch coming.

There are hundreds of ways we protect our stuff, using locks or guards, fences and gates or elaborate video surveillance cameras. Security professionals focus on physical security and digital security to maintain confidentiality, control access, to authenticate users, and to enforce non repudiation. The really good security professionals use combat attack dogs trained in at least ten forms of martial arts along with special training from the FBI, the CIA, MI5, and the KGB. These dogs are super vicious and will rip out your jugular if you don't rub their bellies first. If you have a dog treat, they'll play fetch for hours on end. But there's nothing as close to our hearts as our password.

In this lesson, we're going to take a hard look at how passwords work, what they can do, and what they can't do. Like any other tool, passwords have many uses but also have flaws if they are not used properly. Each day, passwords to accounts are compromised or stolen or just guessed by someone who shouldn't have access to that account. Having a weak password is much like leaving your house unlocked with all the windows left wide open.

Over the past forty years, countless studies have been conducted on passwords. Believe it or not users create and never change really crappy passwords all the time. Study after study shows how 99.9% of all Internet users have weak (easily cracked) passwords and use these lousy passwords for multiple accounts. Worse yet, we also choose bad login names. So, most of the users on the Internet have easy to guess passwords and we use the same login names. Isn't that convenient?

Hacking is easier and easier since more people add themselves to the Internet each day. What is worse (and things do get worse) is that names and passwords are the primary access control used by most websites. A long running survey by Microsoft shows that most common passwords are six to eight digits long, and user names are on average only six digits long.

Before we start hunting for passwords, we need to know quite a bit more about how passwords work. We also need to learn how to crack passwords and how to create strong passwords. This lesson can get kind of complicated because cryptology is involved here. If you don't do well with math or if complex algorithms make you nauseous, read slowly and breathe deep. Grab a paper bag in case you need to hyperventilate.

We are going to get into the weeds on passwords, digging deep down where the really fun stuff lives. Hold onto your seat belt because it's going to be a wild ride.



Pins, Passwords and Personal Poop

You are unique, just like everyone else in the world. You look different, smell different, think different, walk and talk different than every person you've ever met. Sometimes you smell so different that others don't want to be around you. It is easy for you to see your own uniqueness and for you to recognize the differences in those around you. Once in a while you might confuse someone with somebody else but overall we humans are excellent at recognizing faces in a crowd. The problem is how do you prove you are who you say you are to a stranger? Even worse, how do you prove you are you to a computer?

Computers are not that good at being able to spot a single person out of a crowd like we are. Programs are getting better but they are no match for our matching abilities. When you meet somebody for the first time we usually introduce ourselves by our name, followed by some greeting or body contact like a hand shake, hug, kiss on the cheek or punch in the nose (depending on the situation). This introduction is based on your own culture and how those around you greet each other. Some cultures bow at the waist, others hug and kiss one another, while even more just nod and smile. This is how we introduce ourselves and start our first impression.

Deep inside our minds we are sampling their smell, the color of their eyes, the hair style they have, what sort of clothes they are wearing and a million other subconscious sensory inputs are taking place. Based on previous personal experience, we catalog each new contact and form an opinion about whether we like or don't like that person. This forms a trust bond in our emotions as to whether we are pleased to see this person or we really aren't sure about them. Computers don't have this ability, which is good for the some of us who are pretty unlikeable.

Systems need a way to determine if the user is who they say they are. We call this **authentication** which is part of **access control**. Yeah, this is where passwords come in. We need to verify your identification for you to access this device. Usually this is done in combination with a user name. Using just a password and a user name is single factor authentication. In the spectrum of security, we are looking for something that only you know, something you are and something you have. A password is something you know and your retina is something you are. If a token, a fob, credentials, digital certificates or an ID card is also required, than that is something you have. The same holds true for using your smartphone to verify a text with a code because your phone is something you have. Plus that provides two-factor authentication.

Biometrics approaches this by using your physical features to prove your identification through eye blood vessel patterns, finger prints, voice input, breath, facial features, earlobes, microbes on your body, and other aspects of your body that are unique to you. Passwords are a cheap and easy method to partially prove you are who you say you are. These are the basic flaws with passwords: they are easy to forget, easy to lose, easy to steal, easy to replace (which is also, oddly, an advantage), hard to remember, and too easily direct blame at the person who has their account hacked (which is also an advantage but not for password owner). Then again, if your biometric password is stolen, you can't just change your fingers.

I Don't Get It

So basically, encryption is a two-way mechanism. You start with a plaintext written letter, encrypt it to cyphertext and send it to whomever you want. They receive it, decipher it with the key you gave them earlier and they read your original plaintext written letter.

Hackers realized years ago that to keep passwords really secure, you couldn't store them encrypted, because given enough time (and computing power) you can **crack** (decipher) them. So instead of encrypting passwords, systems started to use **hashing**.

Hashing involves taking something (a file, a password, a document) and running a series of operations on it, leaving you with a series of characters called a **digest** or a **hash**. Hashing is a **one way function**, because unlike encryption, what you get at the end of the

process is not reversible. This means that starting from the digest or hash, you will never be able to re-create the original data. (In theory.) But the same data, using the same hashing algorithm, will always generate the same digest. Not only that, but if you change so much as one letter in a 1000 page document, the digest will be completely different.

Feed Your Head: CAPTCHA and Passwords

CAPTCHA has single-handedly kept more morons from posting stupid opinions on blogs and websites than any other technology ever and thereby keeps the Internet from crumbling under the full weight of the stupid part of humanity.

You know CAPTCHA. It's that box with hard-to-read words that you need to type in before posting to websites. It's interesting because it adds a new channel to authentication, the human channel, and it adds entropy for slowing down the attack by making a complicated extra step in the process. It's meant to prevent SPAMMERS and password cracking attacks. If a password is a key then CAPTCHA is a special twist of the key if I may. That's good. Too bad most people find reading mutilated letters off a paisley background more than enough entropy to slow them to a stop.



CAPTCHA is impossible for some and frustrating for the rest of the people. Too bad computers didn't have the same trouble. If we are at risk of the Matrix or Skynet it will be because of CAPTCHA. It's so annoying that if the Terminators wanted to end humanity they would put CAPTCHA on our Internet-connected refrigerators and watch us starve to death.

Methods of Authentication: Passwords

The first step in obtaining proper access to any important information or secure area is to prove you should have access to it. In order to gain that access, you need to authenticate yourself to the system. Authenticating is just fancy word for proving you are who you say you are. You know who you are but strangers and networks need some proof that you're not pretending to be somebody else. Depending on the sensitivity of access, security folks look for two or three items. This is known as **multi-factor authentication**. We are looking for **something you know**, **something you have** and **something you are**.

Something you know is just a piece of information that only you should know. For example, only you and a few select other people would know your favorite color, favorite food, all-time greatest movie, or your password. This information could include the school you last



attended, the name of your pet, the sport and position you like to play, favorite music or book and so forth.

Something you have might be the token you were given for access to a network. It could be your key to the lock or even an identification card. This is usually something that you would physically have in your possession. You must have that on you to gain access to that network as part of multi-factor authentication.

Something you are is unique only to you. This is a big topic for biometrics because it can involve your fingerprint, which have been proven to not be unique and are easily spoofed. However, there are other things about you that are much more provable such as the blood vessels in your iris, the speed at which you type on a keyboard, your thought patterns, your behavior and all kinds of other aspects that make you who you are.

Passwords come into play as something you know. The logic behind using passwords is that only an authorized agent would know the correct password to gain access. (See any problems there?)

There are three main types of authentication:

Strings of Characters: Something You Know

At the most basic level, authentication uses passwords, which are words or strings of characters (numbers and symbols). A keyboard or keypad allows entry of these types of passwords. These passwords range from the simplest – such as the three digit codes used on some garage door openers – to the more complicated combinations of characters, numbers and symbols that are recommended for protecting highly confidential information.

The problem with character based passwords is that the more complicated the password is, the harder it is to remember. The classic solution is writing the password down on a slip of paper next to the computer screen. If you're going to write down a password, keep the paper in your wallet or purse! People rarely lose their wallet or purse and if they do, changing your passwords will be a snap compared to replacing all their credit cards and their drivers license.

Strings of Characters Plus a Token: Something You Know plus Something You Have

The next level in authentication (**two-factor authentication**) is to require a password plus a **token** of some type. An example of this is the ATM, which requires a card – the token – plus a personal identification number or PIN. This is considerably more secure, because if you lack either item, you are denied access. So make sure you don't keep your PIN in your wallet too!

There are all kinds of token based systems out there. One of the current attack trends is for attackers to attach a card scanner (a **skimmer**) in front of an ATM machine's card reader. When a victim inserts their card into the ATM, they enter their PIN. The skimmer reads the magnetic strip off the victim's card and also captures the PIN. The victim is completely unaware of this crime until their bank account is emptied.

The point here is that tokens can be hacked and have been hacked.

Biometric Authentication: Something You Are

The third level in authentication is biometrics. This is the use of non-reproducible biological features, such as fingerprints or facial features, to control access. An example of this is the retinal scan, in which the retina – which is the interior surface of the back of the eye – is photographed. The retina contains a unique pattern of blood vessels that are easily seen and this pattern is compared to a reference. Biometric authentication is the most sophisticated and is considered safer, but it has failed. How about copying someone's fingerprint with a gummy bear, and then using it to authenticate as another person? (Been done.) And hey, if they just need your retina, that's not so tricky, is it?



But biometrics can also be invasive (retina scan are, iris scan are better), dangerous (think about germ propagation), and discriminatory (if a user has some fingers missing, an eye missing etc.).

A reality TV show demonstrated the use of clear plastic tape to copy a fingerprint and fool a fingerprint scanner. Lately, the whole idea that fingerprints were unique to every individual has come under fire. There have been several recent criminal cases where one person's fingerprint matched another person. Like most science based on human characteristics, there's still plenty of work to be done before biometrics can be considered fool-proof.

Let's also keep in mind that science can only prove that something is wrong, it can never prove that something is right. All theories are just theories until they are proven wrong. That is the job of science.

Exercises

- 11.1 Is the SIM card in your cellphone an authentication token or a tracking device? Pull the SIM card out of your phone and see if you can still connect to another cellphone.
- 11.2 Going CSI, try to capture your own or a friend's fingerprint using clear tape. What additional materials do you need to get a "good" print?



Game On: Hungry for Knowledge

Mokoa's stomach groaned next to the gold-fish tank. Several small orange swimmers looked at the teen with worry, hoping he wasn't in the mood for seafood. Gerbils across the aisle hid their apple and vegetable bits behind their water bottle. Even though the pet store was filled with food, none of it was human food.

Mokoa complained to himself that he shouldn't have eaten his lunch so early. It was late afternoon and he was running the store by himself. He also forgot to get a snack from his upstairs room before he started his shift. Normally this wouldn't be a problem because he could just run upstairs and grab a quick bite. However, every time he started up the stairs, another customer would enter the store. Every time.

It was one of those days where a certain customer would come into the store and just ask questions or want to see one thing after another, never buying anything. Mokoa had to treat them with respect and reply to every question or request, because he was the pet store owners grandson. And he was on shift.

But he was also very hungry.

The demanding customer asked to see the bottom bag in a pile of 50 lb. bags of dry dog food. The stack was ten bags tall. The teen pet seller smiled outwardly but truly wanted to throw a bag of food on top of the annoying customer. Using two ladders, Mokoa had just moved the eighth dog food bag when the customer changed his mind. He didn't want the last bag anymore. He now wanted the 50 lb. bag that was on the bottom of the new pile now.


As he stood next to the ladder trying to figure out if he could throw a 50 lb. bag on the customer and tell the police a convincing story, Jace stepped through the front door.

Trumpets blew. Heavenly lights flared behind her. Angels sung in chorus. Jace was here.

"Jace, you are the best thing I've seen all day," Mokoa announced throughout the store.

She squinted her light sensitive eyes, unsure if Mokoa was joking or had some sort of illness. Jace approached him to check his forehead temperature.

Mokoa looked at his best friend and said, "I'll be right back. Hold down the store for me,



will ya.”

He was upstairs in a single step.

Jace turned to the customer and asked, “What is this all about?”

By the time Mokoia returned to the downstairs store, his milk mustache was almost gone. He almost forgot about Jace, now that his tummy was close to being full. Mokoia held a plate with three cookies on it. As he looked around the store, he didn't see Jace but he did notice that the piles of dog food were neatly stacked in their original location. The nagging customer was gone so the teen grabbed a chair and started to sit behind the cash register desk.

“Hey, ouch! Move that thing. I was here first,” Jace yelled.

Mokoia saw Jace laying on the floor using her knapsack as a pillow.

“What are you doing down there,” he asked as he set the plate on the chair.

“Resting, where you been?”

“I needed some food. When you came in, I finally had a chance to grab a bite. Thanks for covering for me,” Mokoia said. “Where did that customer go? Did he buy anything?”

Jace gathered herself up and said, “Naw, he was just looking. I got him to put those bags away before he left, though. Hey, thanks for bringing me some cookies for my effort.”

“Hands off. They're mine. Don't touch or else,” he said in a defensive posture.

“Or else what,” Jace replied.

“Or else I'll do something,” Mokoia said.

Jace mocked, “You'll do something, I'll bet.”

Mokoia guarded the cookie plate and said, “I will and I'll take that bet.”

“What bet,” Jace knew Mokoia was losing this argument. Now was a good time to show him who is the brains in this team. “Tell you what, I'll bet you those three cookies that I can guess your laptop password in under 90 seconds,” she said.

Mokoia replied, “And what if you don't guess it in 90 seconds. What do I get out of the bet?”

Jace thought about for a second before she replied, “Then, I'll teach you how to crack passwords. Deal?”

“Deal,” he replied. He always wanted to learn some of her skills but she never taught him anything.

Mokoia grabbed his laptop and handed it over to Jace. She smiled back.

The teen hacker inspected the outside of the machine and began her lecture, “First off, guessing passwords isn't terribly difficult. The more you know about the victim, the better guesses you can make to unlock that password. For example, knowing your name is a step towards knowing your username. Many times, usernames and passwords are the same or very similar. People are lazy; they don't want to remember too many things.”

“With something portable like a laptop, a keyboard, or a phone, the password is sometimes written on the outside of the device. Users love to keep passwords on post-its under their keyboard or next to their computer screen. It looks like you didn't write your password on the outside of your computer so you did good there,” Jace said as she spied Mokoia for a reaction.

With the laptop turn on and ready for the password, Jace studied the screen. “I will need to know how many attempts I can try and guess your password before it locks me out. Most people disable this security feature and allow anyone to guess their password forever. I suggest you set password attempts to between five and ten attempts. In fact I remember telling you this when you bought this computer last year. Let's see if you



listened to my advice," Jace said as she tried a few basic passwords.

"Nope, you didn't set a password limit," she said with an evil laugh. Jace cracked her knuckles and started to whistle the tune from Clint Eastwood's movie *The Good, the Bad, and the Ugly*.

Game continues...

Password Biology: SWAT

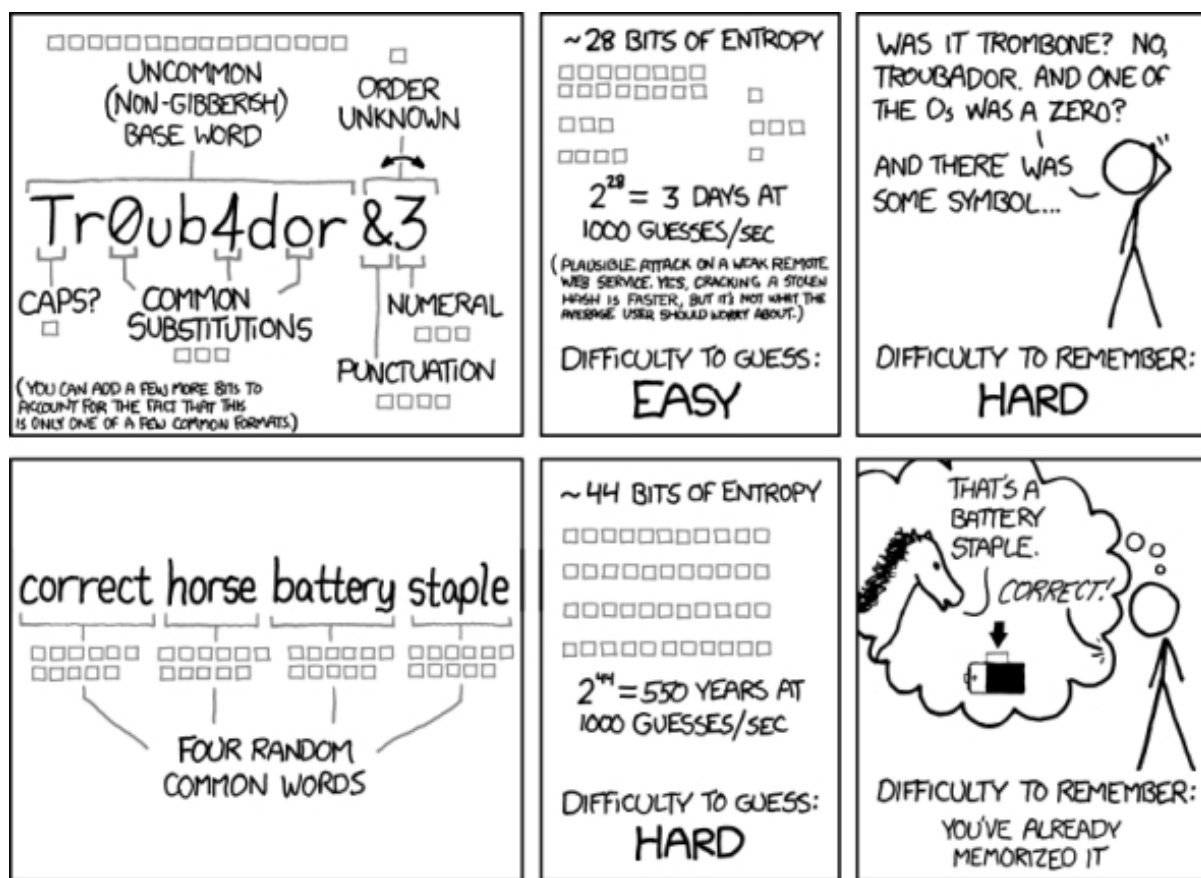
Passwords rule most of the digital world. In particular, passwords rule the entire Internet world. (When was the last time a website asked for your fingerprint?) To take a closer look at passwords we need a tool to open them up. The toolset we'll be using today is called SWAT. **SWAT** stands for Strengths, Weaknesses, Advantages, and Threats. Put on your lab coats and slap on those protective goggles because it's going to get messy.

Strengths

Size matters when it comes to passwords. The longer the password, the more difficult it becomes to crack. At a certain length, a password becomes essentially too expensive to crack. Hackers are busy people, very busy. They want to direct their computing power toward easy passwords. If you have a password that is six digits long, an attacker will focus on you instead of someone whose password is twelve digits long. Cracking a password requires time and resources. For each additional character added to a password, complexity increases exponentially.

The word **entropy** describes the difficulty of cracking any password using brute force, pure guessing, using a collection of common passwords or just relying on dictionaries. This is why many systems limit the number of times you can enter an incorrect password before the system locks you out for a specific time. When it comes to passwords (because the term is also used in physics), entropy is usually measured in bits: the number of bits required to make up a password. For instance, an 8-character password requires 8 bits per character (in English, but not in all languages – some need 16 bits per character) so $8 \times 8 = 64$, meaning an 8-character password space has 64 bits of entropy. And to put it simply, the more entropy, the stronger the password.

Another major strength for using passwords is they are customizable for each user. Passwords are not stored, only the password hash is stored. If you haven't met MD5 yet, we'll get into the hash stuff a little later. Just be sure to know that any change to a password, such as capitalizing a letter in it, will produce a completely different hash output.



THROUGH 20 YEARS OF EFFORT, WE'VE SUCCESSFULLY TRAINED EVERYONE TO USE PASSWORDS THAT ARE HARD FOR HUMANS TO REMEMBER, BUT EASY FOR COMPUTERS TO GUESS.

Figure 11.1: Password Strength (courtesy of <https://xkcd.com/>)

The XKCD cartoon above is famous in the password-cracking world because it's right in some respects (longer is better, easier to remember is better), but also wrong in some ways (using common words makes cracking easier, and using any simple pattern like all lowercase letters is catastrophically weak). The whole phenomenon has been discussed so much it's commonly known as "that horse-battery-staple thing."

Weaknesses

Everything that was just covered as a strength is also a weakness. The size of a password adds additional security but it also adds complexity. Humans are lazy creatures; we want to make things as simple as possible for ourselves. From a security professional view, security should be invisible and effortless to the user. Security should not hamper or slow down work but should operate in the background. Anytime you add additional steps or requirements to the work environment, you slow down work performance. You might have noticed that security is not anywhere near that goal at this point.

As the need for stronger passwords are enforced by the boss, people will always find a way to get around such mandates. The average adult can easily memorize about seven things. Asking a user to memorize twelve characters means that the person will have to either create a password that ends with 12345 or one that they will write down. Worse yet,



they will reuse passwords over and over again. Think of what will happen when a user is told they must have a password that is eighteen characters long! You don't really want to be on either side of that conversation. That's like boss-level passwording.

On top of mandating a twelve or eighteen digit password, add in a requirement to change that password every ninety days (without repeating any of the characters from your previous ten passwords, welcome to the U.S. Department of Defense). You and you users are going to be overwhelmed with memorization. Go ahead, try to enforce this on your family.

Managing ten or twenty users accounts is fairly easy, but most big organizations have hundreds or thousands of users. Managing passwords and logins for larger groups becomes a full time job. Sooner or later it will be a highly hackable mess. From a password management standpoint, removing an account by deleting the user and their password is inexpensive and fairly easy to do. Any hacker knows that administrators often fail to do this, which makes for easy fishing.

Over the years several organizations have been caught storing passwords in clear text. One recent example was a major attack on Sony Entertainment in November 2014. Thousands of usernames and passwords were discovered and released to the public by attackers that showed Sony had stored this information in their networks as easy to read text. Then the Yahoo attack showed us how millions of passwords could be lost the same way. So your best bet is to not store passwords at all but rather store the hash values instead. Even then, the hash values need to be secure too.

Advantages

How much does a password cost? Each password costs an organization a tiny bit of computation and storage (to hold it and look it up) but not much more than air costs you to breathe. Other forms of authentication can cost an organization a small fortune to install, maintain and operate. A lost password is easy to reset or replace. Passwords can be generated and distributed *en masse* in no time at all.

Users can be allowed to customize their own password as long as it meets expected security requirements. Passwords can't get lost like a physical token can. If a token goes missing, the token replacement requirements can be expensive and harmful to the longevity of the employee (you can get fired). Passwords can be replaced in a matter of seconds and a compromised password can be revoked easily.

If more than one person decides to share a password for easy access, that account can be audited and taken off-line. The password would be changed, as well as the login. Passwords can be traceable to the user if suspicious events need to be investigated.

Of course, this also means a criminal could hide her actions behind someone else's account. Isn't that handy? Unless you're the victim.

Threats

The last tool in our tool set is Threats. There are many areas of concern to acknowledge when it comes to password threats. The easiest of these threats to attempt would be password guessing. Thanks to the Internet, there are many resources available for guessing passwords. For example, fire up your search engine of choice and search for "common passwords." Start compiling the common passwords from your results into a single location and before you know it you'll have a great password dictionary to assist you with password guessing.

Social Engineering is another great way to determine a users password. You would be amazed at how many people use the names of their children, spouse's name, pet names, anniversary or birth dates, favorite color, etc. for their password or as a combination to make up their password. All of this information can be gathered from someone through casual conversation or examining the articles in their office or cube. The best way to prevent against this form of attack would be avoiding the use of any information that may be common knowledge.



Dumpster Diving and shoulder surfing can also provide quite a bit of useful information about someone that can be used to guess their password. Being aware of your surroundings, using a screen protector, not facing your monitor towards a window and using a cross-cut shredder can assist you in protecting against those threats.

With the large amount of social media platforms now available, it's getting even easier to find out information about people. You can probably get someone's favorite color, birthday, kids names and spouse's name right from their Facebook page. Be cautious of the information that you share via social media and keep strong privacy controls in order to help reduce the threat of attackers phishing for information that can be used to guess passwords.

Offline Password Cracking is a threat to all of you who think your password is so good! If the attacker has the password file then they have all the time in the world to run dictionaries and rainbow tables against it.

Phishing may be the easiest and most popular way to get account details. In a phishing attack the cracker sends a link to a fake login page via email or chat. The unwitting victim clicks, sees a familiar login page and enters their credentials. Often from there they're redirected to the real login page, where they will be confusingly not logged in. Oh well, try it again. But by now the cracker has the victim's login details.

Password Sniffing means just waiting for you to type a password and grabbing off your system, network, or even right off your keyboard. While keeping your computer clean of key capture dongles that fit between your keyboard and PC or malware that waits for passwords will help, it won't do any good if the attacker is between you and the server you connect to. In that case, encryption is your friend. On web sites, enable **HTTPS Only** in your browser to ensure that the site uses an encrypted connection whenever data is exchanged. Otherwise it's too easy to grab your password straight from your connection to the server.

Shoulder Surfing is nothing more than someone looking over your shoulder while you type a PIN or password. Of course they can be standing three blocks away and have a really good camera lens.

Stupid Friends are exactly what it sounds like. Take care of sharing your password, also with friends. On some web sites terms of use – these are an **Indemnification control**, see the OSSTMM for details – you are responsible for all activities generated with your credential.

Keyloggers, RATs and other malware can set up a keylogger or RAT server on the victim. Your boss can do it too, using either software or a slick little piece of hardware that plugs in between your keyboard and your computer. The keylogger records every key stroke of the victim. Everything you type, the keylogger records and sends to the cracker. Some malware will look for the existing web browser's client password list and copy it to a remote cracker, making the passwords easily accessible if they aren't encrypted.

Consider this: when you answer **password reset questions**, do you use the real answer? We know we don't and we recommend that you do not either. Password reset questions are a great way for attackers to guess your password. Next time you answer one of those questions, think of something other than the real answer to use. For example, come up with a strategy that says, each time you see the question, "What was the make of your first car?" you answer it with the word "Chocolates." We all know there are a limited number of car manufacturers, so it would be pretty easy for a person to rip through a list of car makers. We seriously doubt anybody would be guessing "Chocolates" for the make of your first car.

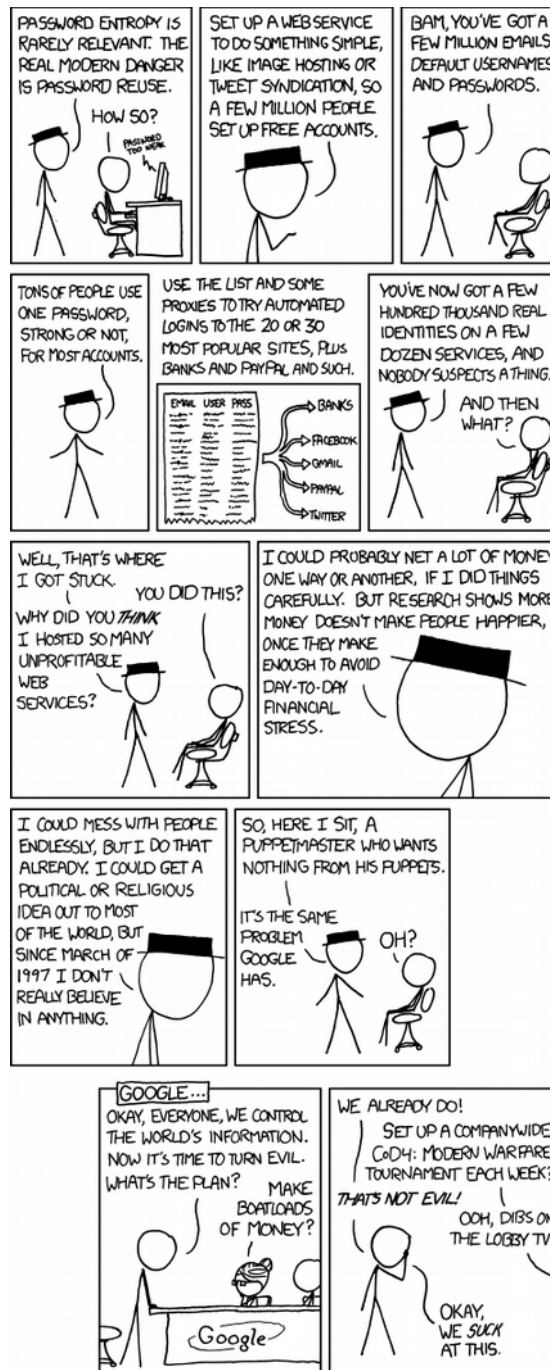


Figure 11.2: Password Reuse (courtesy of <https://xkcd.com/>)

In order to help decrease the number of password threats you are exposing yourself to, get in the habit of following best practices such as memorizing passwords instead of writing them down, not reusing passwords and creating strong passwords. The creation of strong passwords will be discussed later in this lesson.

Another important threat is the password reuse. A long time ago in a galaxy far, far away an important website was compromised via SQL Injection and attacker found passwords stored with weak hash algorithms and e-mail addresses users used to register. The attacker tried – with a lot of success – e-mails and associated password on various social network and posted malicious messages containing a malware. This is an important lesson: never use the same passwords on different services, are you using same passwords? Change them now! Have you difficulties to generate or remember complex passwords? Use hints in this lesson and a good password manager.



Human Biology

Memory

We store every bit of sensory information our body interacts with for a fraction of a second in the cranial cortex (your noggin). Some of that input is moved into our short-term memory, such as parts of a conversation with your parents or your math teacher showing you your grades. The sensory data that catches our attention, like the school bell ringing or your smelly socks under the bed, will stay in short-term memory throughout the cortex. This short bit of excitement stays in our memory for just a few seconds, perhaps a minute.

Studies from the Harvard School of Medicine indicate that the average adult can memorize nine random items, characters, numbers, or other unimportant items for up to a week. After that week, our capacity for memory drops down to seven items. If the collection of random items are not utilized, or rehearsed, these seven items slowly fade in less than three weeks.

If we do not attach some sort of meaning to random objects for later retrieval, even if its movement (because muscle memory rocks!), those items will not be moved into our long-term memory. This plays an important role when working with lists, tasks, combinations, events, recollection, and passwords. If we can associate new information with old memories or past events, those new memories will become locked in long-term brain storage.

Basically, humans can't be expected to memorize long strings of characters and random digits if there's no real meaning behind them. This is one of major faults with computer generated passwords when you are strictly forbidden to write them down. (Welcome to black ops.) How can you be expected to remember a string of eight, ten, or twelve characters when your short-term memory is maxed out within one minute? Try telling this to your mom when you forgot her birthday, again.

The problem is most networks have a password update policy that is almost impossible to comply with. These policies require the user to have a password of nine characters or more but not more than twelve. The password must be a combination of upper and lower case words, have numbers and special characters. The system stores your previous ten passwords and you must pick a new password every 45 days. This is where a password manager comes in.

One of the problems with such a policy is that people can't remember such long passwords, nor do they have the creativity to make new ones and remember them every 45 days. Plus, passwords usually start off with a capital letter and end with number or special characters. That makes cracking this policy even easier.

Bad Habits

As our lives become more dependent on fast electronic devices and a speedier world, where we are expected to do more in less time, our attention span narrows. School teachers report that thirty years ago they could focus the entire class on a single subject for over an hour. Now, these same instructors are reporting that they have to change topics or refocus the students every eight minutes. Our attention span is shorter as we are getting accustomed to instantaneous stimulation ever minute of the day. The worse part of all this constant content, is that we think we can perform more activities at the same time, or multitask. College students and new workers who enter the world confidently proclaim their skills of conducting four and five tasks at the same time, with the same level of accuracy as focusing on a single task.

Research at the Massachusetts Institute of Technology (MIT) tells a completely different story. Three independent studies found that individuals who think they can perform many tasks at the same time, with a high level of accuracy, fail miserably when tested. The test subjects were carefully selected to ensure they had scored fairly high on academic tests and used smart-phones to tweet, socialize, research, email, answer calls and other activities we associate with the "tech generation."



Each test group was given one main task to perform such as writing a policy letter or typing a memo without errors. As the testing progressed, the test subjects began to receive messages, texts, requests for chat, emails that needed to be answered with researched information, and other multitasking activities at set intervals. The test subjects failed to perform a single task. Each task was full of simple errors and many tasks included answers that belonged to a different task.

The next phase of this study added one more critical element sponsored by the National Traffic Safety Council (NTSC), driving a simulated car. One in ten students were able to back out of a simulated driveway and move down the road one block before they crashed due to multitasking. Again, each task that was added to the main objective of driving a car was nothing more than answering a call, checking an email and looking up a traffic report on a smart-phone with a cup of water in the test subject's lap. Sound familiar? That's because millions of drivers are over confident in their ability to accomplish several tasks at one time. Our brains were not designed to handle the complexity brought on by modern technology. Instead of making life easier, it just makes our lives run at a faster pace.

We Want Change

We are creatures of habit. We want to conform to our environment just as we want to be accepted by our peers. We also need to make our lives comfortable but interesting. This combination causes mankind to resist change, more specifically, abrupt change.

In the world of security we preach to our users the basic principles of using long passwords, using upper and lower case, changing passwords, and not using the same password for all accounts. However loud and long security folks bang this message into everyone's head, it's still hard to change habits. Resistance to change is what makes hacking passwords so easy, unless some incentive to use strong passwords is applied (like keeping your job).

Incentives can be either positive or negative. A positive incentive would be something as simple as a little banner telling a user that their password is strong or a pat on the back by their supervisor. A negative incentive would be getting fired because their account was hacked due to a weak password. If you're a security professional, you will need to find a balance between the two forms of incentives. If you're hacking the system, you can use them to your advantage.

Humans respond better and longer to a positive incentive than a negative one. Think about it this way: do you like being yelled at for having a bad grade or would you rather be given a cash reward for earning good grades? Cash, right? In the security arena, humans have been and always will be the weak link for a security environment. You will need to "sell security" to your peers, your friends, and your family. You can start by looking at the current password policy your organization has and work up from there. What – you don't have one?



Feed Your Head: Bad Passwords Are Your Problem

Many places claim that a criminal hacker got into an account because the user made a bad password. That's just shifting blame. The truth is that the password problem isn't a people problem. It's an authentication problem.

The thing is that people are people. It's too easy (and cheap) to put the responsibility on just regular people for making uncrackable passwords despite security researchers (and anyone with common sense) knowing that really good passwords are too long and too hard to remember so they can't really exist in a realistic, usable state. But for legal and economic reasons, companies continue to make their users accountable for making and protecting their own passwords so that the server and service owners can't be blamed for giving access to the wrong person.

So how you prove who you are, how you sign up, make passwords, retrieve lost passwords, change accounts, etc. is in YOUR slippery hands.

But you only have a problem if you're careless with your accounts, share them, get phished, click on something you shouldn't, get malware on your system, or any part of your communications are hijacked. So no problem because that like NEVER happens! (Note: see every major criminal hack in the last 5 years and a book on sarcasm).

So yes, the current scheme of login and password that we all know and suffer under is a careful blend of legal deniability, fiscal sensibility, and a hefty pinch of what most parents of the world call "not thinking things through". You don't ask regular people to make irregular passwords. But it's done all the time so, you still need to know how to make unbeatable passwords and protect them.

Hashed Passwords and Salts

Once everyone discovered that storing passwords in plain text was a bad, really, really bad idea, some smart folks came up with a better system. Instead of keeping passwords lying around in plain text, let's encrypt them. The problems with encryption are the computing power required to encrypt and decrypt the password, and safely storing the encryption keys. Ha ha, got your keys! Now we have ALL your passwords.

Next, some smart folks came up with the idea of using a one-way **hash** algorithm on a password to create a hash string. This hash value can't (in theory) be reproduced through reverse engineering, hence the name "one-way." Easy, right?

Hashes (MD5, SHA-1, SHA-2 and SHA-3)

A hash is a mathematical formula that uses a password and sometimes random data (a **salt**) to return a single value. Think of a hash as a locksmith creating a key to fit one lock. The locksmith uses unique materials and special tools to make each key so that only one key will ever fit that one lock. This **hash key** is built using the password as part of the material and some added information (the salt) to make the rest of the key. The hash key or **hash string** will only match one particular lock. If you try to create a duplicate key, the fake key won't match because only one key will ever fit that one lock. It's pretty tight.

Let's try this another way. Take a can of colored paints, blue, green, red, yellow, brown, white, purple and mix some of each paint into a small jar. Don't worry about how much of each paint you add, just pour, mix, repeat. Whatever color that jar becomes with all those colors mixed in, is fairly original. It would be difficult to duplicate this process and make the exact same color.

Now, let's repeat this exercise with a bucket full of sand. In that bucket each grain of sand is slightly different from the next, and there are millions and millions of pieces of sand in there. This means getting another bucket full of the exact size, color, and shape of each



piece of sand in that original bucket would be almost impossible. When you are done with your first bucket of sand, throw it into the ocean. Now, try and locate an exact duplicate bucket of that first sand batch at another beach located on the other side of the world. A hash value is almost (yes, almost) impossible to repeat ever again.

Those grains of sand or exact mixes of color are what are used to formulate an answer, not the sand or paint themselves. If one grain of sand in the new bucket is slightly different from the original, the calculations will be wrong. Any one change to the data will result in a completely different calculation answer: The lock will not open.

Since nothing has to be encrypted and decrypted, it's easier to calculate a hash value when a user types in a password. Instead of looking up the user's password, the password's hash value is calculated and compared to a stored hash.

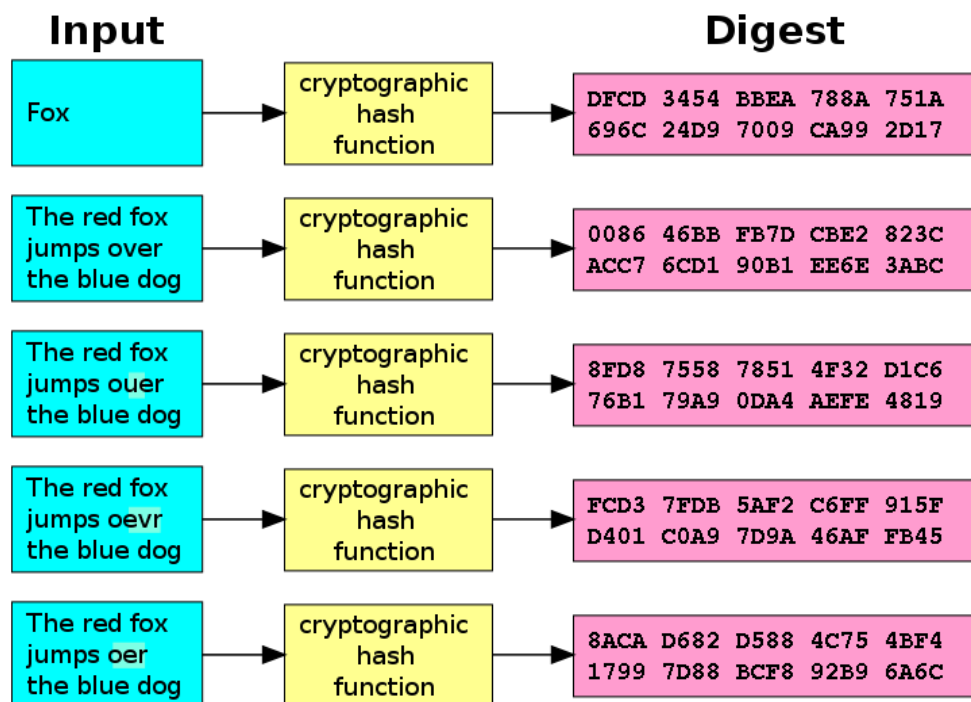


Figure 11.3: Cryptographic Hash Key Function. Notice how changes to Input alters digest (output). Courtesy of Wiki Commons.

Salt

The random bits of data that are added to the hash algorithm, like a key for computations, is called the **salt**. A salt can be used to help create the hash value when combined with the original password, since it adds additional levels of complexity to the answer key. All this hashing and salting creates a one-way hash value. A salt is usually a string of random data from 48 to 128 bits long. The longer the salt, the higher the level of complexity and difficulty to crack.

Using a salt with the hash value makes many password attack methods slower, such as the good old dictionary attack and a friendly brute force attack. More on these attacks later in this lesson.

One of the attack techniques mitigated by salts is **rainbow tables**. With this technique an attacker pre-generates huge lists of hashes. In order to add complexity and prevent this attack, you can generate a specific salt for each user (so that an attacker would need to pre-compute a rainbow table for each user).

Key Derivation Functions

A step forward in hash functions are **key derivation functions**. These are used to derive keys from a string, a salt and a (big) number of iterations in a formula something like this:



DK=KDF(Key, Salt, Iterations)

where DK is the derived key, KDF is the key derivation function, Key is the original key or password, Salt is the cryptographic salt and Iterations refers to the number of times the subfunction is run.

The whole point is to make hash generation slow. But not too slow! "Fast" hash algorithms let brute-force and dictionary attacks run wild; slow algorithms make them too slow to be practical in many cases. Too slow algorithms introduce <http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2006-5229>.

Super Duper Complex Cryptography

Take a deep breath and turn off your cellphone for this next section. We will do our best to avoid giving you nightmares or headaches about complex mathematics. If you want, you can just skip this section if you scare easily or are prone to fits of rage.

Cryptology means altering information or data to make it incomprehensible and unreadable unless you possess some secret knowledge. One of the ways we do this is encryption.

Back when passwords were encrypted, there were lots of places to get attacked. The effectiveness of encryption, usually described as its **strength**, ranges from very weak to extremely robust.

At the weakest, passwords were simply **encoded**. This produces a password that is not readable directly, but, given the key, can easily be translated using a computer, pen and paper, or a plastic decoder ring from a cereal box. An example of this is the **ROT13 cypher**. ROT13 replaces every letter in a text with the letter that is 13 places away from it in the alphabet. For example "ABC" becomes "NOP."

Even when using algorithms that can more accurately be called encryption, the encryption is weak if the key used to generate it is weak. Using ROT13 as an example, with the 13 place differential as the key, then ROT13 has an extremely weak key. ROT13 can be strengthened by using a different key. You could use ROT10, replacing each letter with the one ten places forward, or you could use ROT-2, replacing each letter with the one two places before it. You could strengthen it even more, by varying the differential, such as ROT π , where the first letter is shifted 3 places; the second, 1 place; the third, 4 places; the fourth, 1 place; and so on, using π (3.14159265...) to provide a constantly varying differential.

Because of these possible variations, when you are encrypting any type of information, you must be sure that you are using a reliable method of encryption and that the key – your contribution to the encryption – will provide you with a robust result.

You must also remember that a good system of encryption is useless without good passwords, just as good passwords are useless without good encryption.

Exercises

11.3 In Linux, in a terminal, create a new file using the touch command:

```
touch testfile.txt
```

Now put some text in it by using the redirect character (>):

```
echo This is a test > testfile.txt
```

Now use the MD5sum command to generate the unique hash or digest value of that file:

```
md5sum testfile.txt
```

You will get a long hash value back. Look at it carefully, then add to the text in your test file using the append operator (>>):

```
echo . >> testfile.txt
```

Run the md5sum command again. Compare the new hash value to the one you got before.

11.4 Here is a list of fruits encoded using the ROT13 cypher. Try to decode them:

- a) nccyr
- b) benatr
- c) yrzba
- d) jngrezryba
- e) gbzngb

11.5 Is there a web page that will allow you to decode the ROT13 encoded words automatically?

Password Cracking

Password cracking for anything other than stuff you own is against the law. If it's your password, then it's your information. Once you password protect something, and then forget your password, you are stuck. Or in another common scenario, you're taking over management of a network and didn't get all the password information from the last admin. This is where password recovery becomes really useful.

Sometimes, for research or analysis, experts crack giant lists of password hashes, which is called **hashcracking**. What they're trying to find are **plains**: plain-text passwords. Methods for cracking fall into just a few categories. These include:

Use the (Brute) Force

This is one of the most misused terms in password cracking. While in a generic sense you can say this term means trying millions of passwords until you find the one you're looking for, that's not accurate in our world. (Show people how 1337 you are by knowing what it really is.) Actually, brute-force attacks try every possible combination in the keyspace, or at least as many as it takes to find the password. They don't start with a list; they generate all the permutations on the fly. This means that brute-force attacks are very thorough on the one hand, but can take a very long time to run on the other. That's why experienced password crackers will try brute-forcing passwords of up to seven or eight characters in lower-case only, then switch to other attacks for longer and more complex passwords. You'd think people wouldn't use such simple passwords – but about ten percent of them did in a 2013 experiment by ArsTechnica.

In a brute force attack, you are basically taking a library of words, numbers, or characters and trying to match each one against the actual password. When you are looking for your socks, you can always find one clean sock but you can never find the other matching sock. So, you look through your room picking up every sock you can find and comparing it to the one clean sock you have. You are thinking to yourself, "is it this one, is it this one, is it that one" as you stumble to locate the other matching sock. Brute force works the same way, it just doesn't smell as bad.

Don't be fooled by the simplicity of this attack method. It may take up all of your computing resources and a tremendous amount of time before a match is found, if one is found at all. Be prepared to use another computer while the brute force attack is ongoing. Across the Internet you can find tools that will automate this process for you. This may be kind of them, or it may be a ruse to crack that password for themselves.

Exercises

11.6 First you'll need a **hashdump** file, preferably a huge dump of millions of hashes for you to gleefully crack with your friends. Get lots of potato chips.

You're going to get a truly epic dump of password hashes courtesy of every user of LinkedIn in 2012. (Thanks, everyone.) Use your search engine skills to locate the file: we suggest a search along the lines of "LinkedIn hashdump and passwords" if that doesn't make it too obvious.

Download the file. Unzip it. Put it someplace sensible, somewhere that you can remember.

- 11.7 Now you need a tool that can put that hash dump to use. Go to <http://hashcat.net/hashcat/> and download **hashcat**. This is a serious professional tool, and you're expected to figure it out for yourself. Amazingly, this is in fact possible. On the same page, scroll to the bottom to the Help section and follow the Video link to a video site. There are excellent step-by-step tutorials that you can explore at your leisure, but for now look for one that's under five minutes in the first page of results. One we'll suggest is "Using Hashcat to Bruteforce Encrypted Hashes" at <https://www.youtube.com/watch?v=RMg2uaKxhdA>, which gives you a good example of the syntax:

```
hashcat-cli.exe -a 3 -m 0 -o cracked.txt -n 1 --bf-pw-min=2 --bf-pw-max=15 --bf-sc-buf=0123456789 hash.txt
```

where `-a` is the attack type, in this case 3 for brute force,

`-m` is the hash mode, in this case 0 for MD5,

`-o` is the output file for cracked passwords, in this case `cracked.txt`,

`-n` is the number of threads, in this case 1,

`-bf-pw-min=2` means that we're setting the minimum password length to two characters,

`-bf-pw-max=15` means we're looking for up to 15 characters,

`-bf-sc-buf=0123456789` means that the character space buffer holds the characters 0-9,

and `hash.txt` is the name of the file with all the hashes. Substitute the names of the real files you got in the above exercise.

You can get more information, like the numbers for different attack types and hash modes, with the command:

```
hashcat-cli.exe -help
```

See if you can crack some hashes. Particularly try different lists of characters in the character set buffer for lots of fun revelations.

Dictionary attacks

These run through a series of possible dictionary words until one works as a password. This is slow for a different reason than brute force: each word in the dictionary has to be hashed, then compared to the hash you're cracking. Dictionary files, or **word lists**, are available all over the Internet, though the quality varies hugely. A simple word list might just list words alphabetically, but the probability of letters or words showing up isn't the same as alphabetical order, so these lists are always slow. An optimized or custom word list will put more-likely words first, meaning that the solution will probably be found more quickly. Master hashcrackers build their own word lists and guard them jealously. You won't find many or any of these online. You'll have to make friends with somebody who has them. But choose your friends carefully: some people are researchers (hackers) and some people are criminals (crackers). Don't learn the difference the hard way.

As mentioned earlier, humans tend to be creatures of habit. Users will make passwords that are easy to remember. Often, those passwords are common words found in a dictionary. Brute force uses several libraries to compare against the encrypted password, looking for a match. This type of attack works well if you are looking for your own lost password and you just needed a hint.

The key to success in using this attack is to use as large and as many different types of word lists (libraries). And by types we mean industry word lists, names, and especially languages because, you know, there's a few other languages out there besides English.



As a hacker or security professional you will need a fairly decent set of software tools, scripts and hardware to do your job properly. The interesting part of computer security is that hackers openly share their tools across the Internet, yet security companies charge big bucks for their security tools and knowledge. Open Source software is provided to anyone for free as long as proper credit is given and the software is not commercialized in anyway. Pretty cool, huh?

Exercises

- 11.8 Now you need some word lists. Openwall offers several sample word lists at http://download.openwall.net/pub/word_lists/. Go there and download the all.gz file, which contains (you guessed it) all of the word lists.

Linux and OSX can unzip Gzipped files natively. In Windows you'll need an add-on compression tool like 7zip (<http://www.7-zip.org/>). Get it if you need it. Unzip all.gz and put the output in a logical place.

- 11.9 Now you need a tool that can put those word lists to use. This time we're going to use John the Ripper, so you can get experience with more than one tool. Remember that Hashcat could do the job too.

We'll start with software password crackers. **John the Ripper** at <http://www.openwall.com/john/> is at the top of the list for its success and longevity. John works on several platforms (across platforms) such as UNIX, Windows, OpenVMS, and BeOS. If you build a live CD or bootable USB drive, you must include John because it's that good. Trust us. John is an Open Source package and has been worked on over the years by teams of volunteers.

Get it. Install it. If you're using the Fedora Security Spin, you've already got it.

- 11.10 Open your new friend John and see what you need to do to crack passwords from your list.

Somewhere Over the Rainbow

How do you speed up a dictionary attack? Well, how about pre-hashing all those dictionary files? That would sort of make sense, until you remember that there are lots of different hashing algorithms out there. What you really need to do is hash each password with several different algorithms, and store the results in a table with a column for each algorithm. These tables, with the plain and the hashes of different fixed lengths, looked like a rainbow to somebody somewhere, and that's how we got the name. **Rainbow tables** are an example of **space/time tradeoff**. In this case, the tradeoff is between huge (and we mean really huge) use of disk space, to save huge amounts of time during actual cracking runs.

In practice, creating rainbow tables of entire word lists is impractical precisely because of the issue of file size: you'll fill up that terabyte drive fast. So hashcrackers do something different, something very, very smart. They crack the hash by cracking small parts of it at a time, then assembling the parts. This is wildly clever and intensely mathematical. If you are a hard-core math geek you are gonna love this stuff.

One logical attack is to look for repeated values in calculations. Using a hash (with salt) the word "Stun" could look like **dr23 n9n2 8v84 2lwi**. So, why not look for a pattern in other hash calculations to see if you can find a match?

Rainbow tables do a nice trick. Instead of looking for a whole password (or whatever it is you're trying to decrypt), rainbow tables let you look for fragments or pieces of passwords. Rainbow tables aren't exactly tables; they are a collection of columns in which each column uses a different reduction function. If you assigned each a color you could think of the columns as a curve, or a rainbow. Hence the name rainbow tables. The whole idea is that if the calculated values of any two "colors" matches, you've found part of the password.



From a mathematical perspective, try and wrap your mind around a simple formula of

$$v+x-4= 162$$

both v and x could be any combination of numbers as long as the answer equals 162.

The rainbow tables already have hash values computed and any input you provide (to crack a password) is matched against each column (reducing the values) until a perfect match is found. As we saw with "Stun" above, the hash calculation would look like this: **dr23 n9n2 8v84 2lwi**. If you were to apply this dr23 n9n2 8v84 2lwi string into a rainbow table automated lookup generator, every two digits are analyzed and reduced to find a matching character. While this lookup is occurring, each column is reducing your input, thus speeding up the match finding process. Simple stuff, right?

All the rainbow tables are doing is looking for the answer $v+x-4= 162$. The v and the x values are the unknown factors and the tables have already been calculated to find that type of answer, just on a massive scale.

Since hash values are one-way, it is impossible to reverse engineer any hash string and expect to get a correct answer. Think of a hash calculation as a door that only opens in one direction and traffic can only flow in that same direction. It is highly likely that hash values will be the same somewhere along the way and that is how rainbow tables operate. They compare hash calculations that you provided against a massive collection of predetermined hash values. As the two digit set moves through the table, each column reduces the answer until a match is found.

Exercises

11.11 Rainbow Tables

Get some rainbow Tables from <http://ophcrack.sourceforge.net/tables.php> for Windows XP, Vista or 7.

L0phtcrack

L0phtcrack is a powerful commercial password auditing tool. Help can be found on the official website at <http://www.l0phtcrack.com/help/using.html>. See if you can afford it.

Combinator Attacks

Unfortunately, rainbow tables have a weakness: they're easily defeated when systems use a salt. We'll talk about those below, but the gist is this: when the user creates a password, the system also creates a random salt (a string of a few letters to add to the password), combines password and salt, and only then creates the hash. Every user's salt is unique to them. The salt can be stored unencrypted in the same table as the password hashes, in theory. At least that's the way they used to do it. Obviously it's way too easy to steal both the hash and the salt, if they're in the same table, so best practices seems to tell people to store the salts in a separate table, or in a separate database, or on a separate machine. But that also increases the attack surface and makes new potential vulnerabilities through complexity. In any case, the added salt makes the original password much harder to derive from the hash, effectively defeating rainbow table attacks. Fooyey.

Okay, you broke our rainbow tables? Fine, we'll do something else smart. We specialize in smart. How about we try something like this:

1. First, we've all got machines with powerhouse graphics cards, right? Good: now,
2. Run a brute-force attack for lower-case-only passwords up to six characters, and
3. Repeat for upper-case-only passwords up to six characters. This lets us keep passwords of six characters or less out of our word lists; brute force can test literally all possible combinations.



4. Then run a dictionary attack using, of course, highly optimized word lists. Let this run for at most an hour, so we capture the most likely passwords.
5. Next, let's use that same word list again, but truncate all the words down to seven characters or less, then put a 1 at the end. Try again with a 2 at the end. Try again with a bang (!) at the end. Run all these trials only until cracked-password output slows down.
6. Chop the words in the list down to six and try brute force again with up to two digits or special characters, then three. If you've got a hot graphics card try four, otherwise at the three or four digit length switch to digits only (no special characters). Appending the output of a brute-force attack to the words in a list makes this a **combinator attack**.
7. Transform every password in the word list by making its first character upper-case. This is where people put upper-case letters: at the front of the password. Taking advantage of this kind of pattern turns a brute-force or dictionary attack into a **mask attack**, an especially deadly attack that reduces the keyspace dramatically so that cracking happens much more quickly.

We can run each of these attacks for only as long as they produce lots of plains quickly, then jump to the next attack. We want lots of plains today, not all the plains next month. We're doing research, or we're in a competition, right?

Location, Location, Location

Say you're having a security contest with some friends. If you have physical access to your target's computer, **try looking around**: passwords are often taped to the bottom of keyboards, under mouse pads, posted on personal bulletin boards, in calendars or daily planners, wherever is handy. People tend to post their passwords in plain view but try to be tricky by writing the password backwards. It is not that people are lazy, they just to make their lives as uncomplicated as possible.

There are two commonly used methods to recover hash values in Windows. The first one is an injection using Local Security Authority Subsystem Service (LSASS) and the other is taking a direct reading off of the registry using SAM/System. LSASS reads the hash directly from memory, while SAM reads the local registry hives. There are other methods that can be used to gather hash values but LSASS and registry reads have a higher probability of success. Then the attacker can research and crack those hashes at his leisure off-line. That's especially handy because the attacker can gather several (if not all) the hash strings in a single breach.

Network-based Password Cracking

What about those times when you don't have a password hashdump, but need to recover the password to a database, web application or some other online service? This is when you need a **network password cracking** tool.

THC Hydra

A speedy network authentication cracker which supports plenty of different services, Hydra is at <https://www.thc.org/thc-hydra/>. When you need to brute force crack a remote authentication service, Hydra is often the tool of choice. It can perform speedy dictionary assaults against more than 30 protocols, including telnet, ftp, http, https, smb, several databases and many more.



Figure 11.4: Selecting a target in Hydra

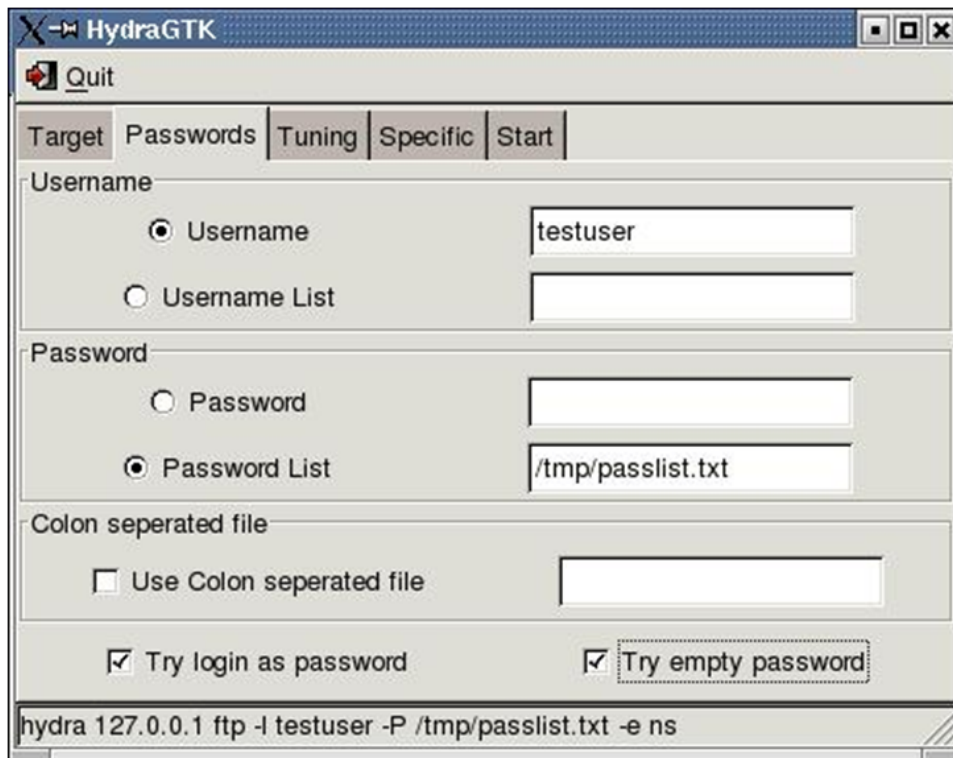


Figure 11.5: Setting up Username and Password lists in Hydra

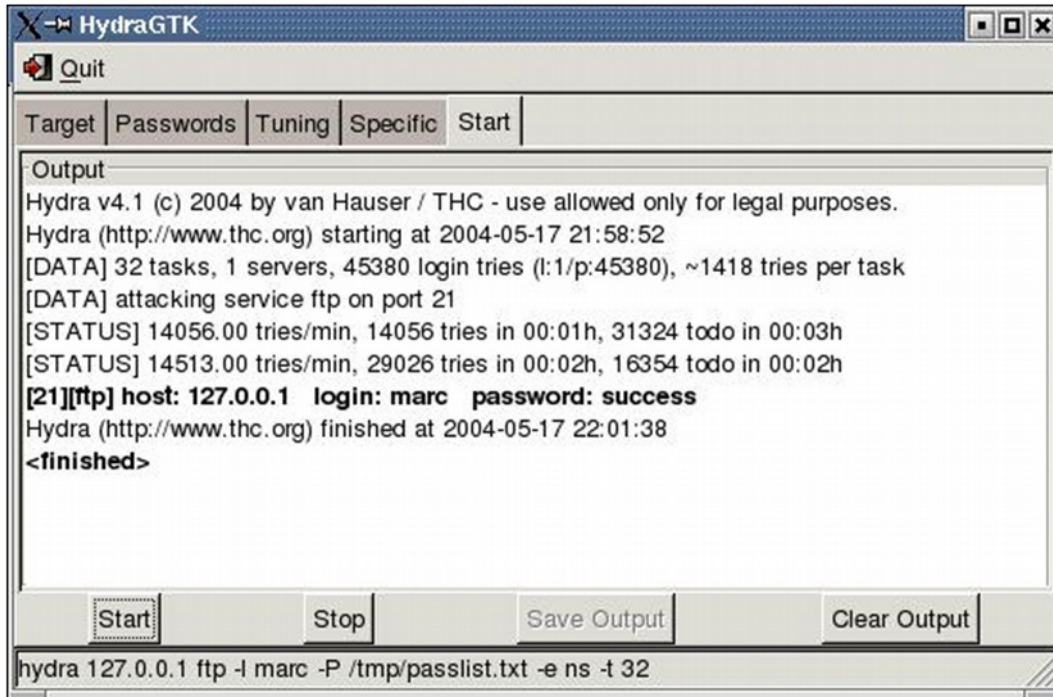


Figure 11.6: The results of a password cracking run in Hydra

Brutus

Brutus is a network brute-force authentication cracker, available at <http://www.hoobie.net/brutus/brutus-download.html>. This Windows-only cracker tests network services on remote systems, trying to guess passwords by using a dictionary and permutations (see Combinator Attack above). It supports HTTP, POP3, FTP, SMB, TELNET, IMAP, NTP and more. Because no source code is available, this is not a truly open-source tool, but it's a very popular one.

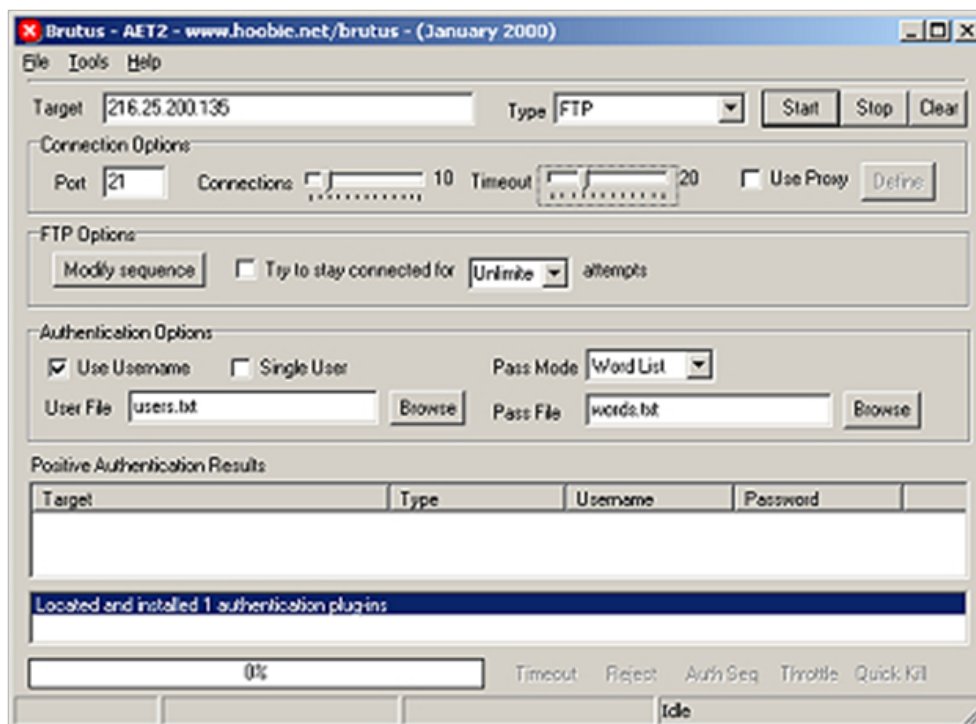


Figure 11.7: Selecting target and options in Brutus

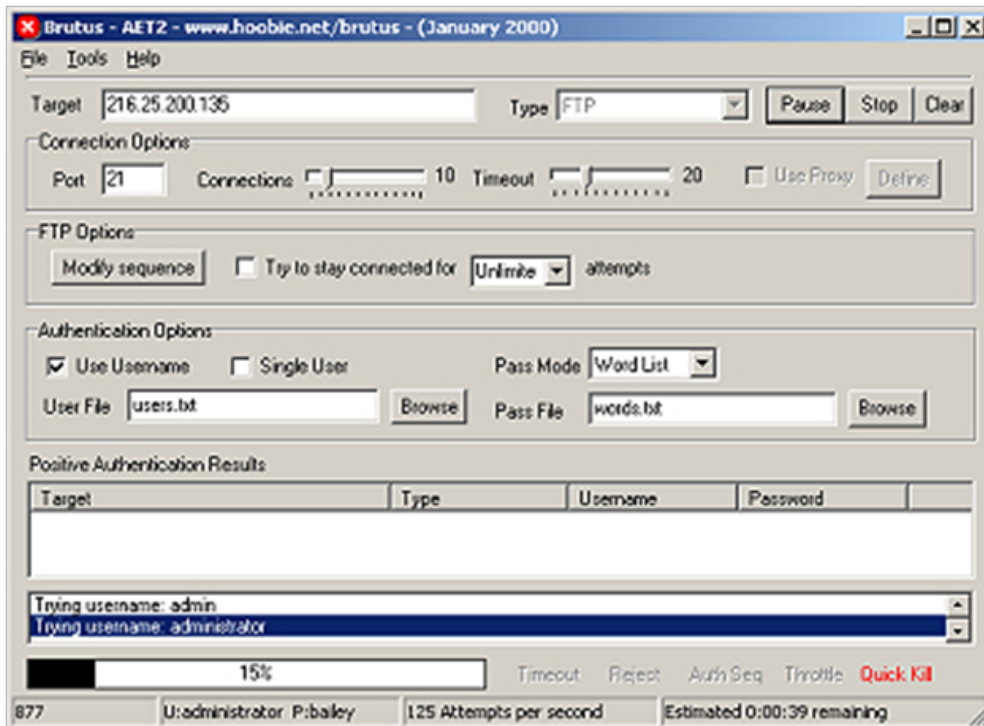


Figure 11.8: A run in progress in Brutus

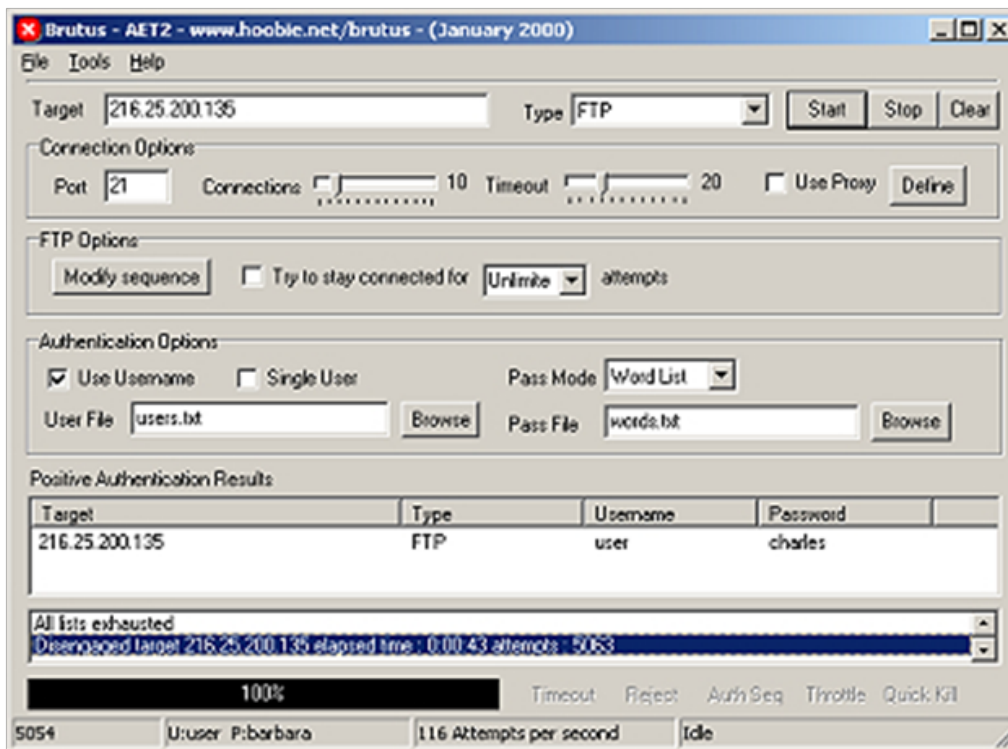


Figure 11.9: A completed test run in Brutus

Wireless Password Cracking

Aircrack-ng - <http://www.aircrack-ng.org/downloads.html>

Once upon a time there was **Aircrack**, an open-source wireless security testing suite. It was good, but this field changes fast, so along came the next-generation suite (which is a fork, or split-off project, of the Aircrack project) called **Aircrack-ng**.

Aircrack-ng contains several tools: **airodump-ng** (a packet capture program), **aireplay-ng** (a packet injection program), **aircrack-ng** (for static WEP and WPA-PSK cracking), and **airdecap-ng** [decrypts WEP/WPA capture files]. The suite can recover a 40-bit through 512-bit WEP keys once encrypted packets have been gathered. It can also assault WPA or WPA2 networks using advanced cryptographic methods or by brute force.

It needs a wireless network interface controller with a driver that supports raw monitoring mode (injection), and can sniff 802.11a, 802.11b and 802.11g traffic. There are versions for Linux and Windows; the Linux version has been ported for **OpenWrt** (a popular open-source wireless access point operating system) as well as the Android, Zaurus and Maemo platforms.

```
C:\aircrack-ng-0.4.2-win\bin>aircrack-ng.exe -a 2 -w dict capture2.cap
Opening capture2.cap
Read 607 packets.

# BSSID          ESSID          Encryption
1 00:06:25:BF:64:99  cuckoo        WPA <1 handshake>

Choosing first network as target.
```

Figure 11.10: Cracking a WEP key with aircrack-ng

```
[00:00:25] 4090 keys tested (160.75 k/s)

KEY FOUND! [ passphrase ]

Master Key      : CC 9D 81 0B 93 70 BE 17 BD 60 18 2E D0 D9 11 EB
                  E7 51 BD 15 4D 92 30 87 3F BF FC 32 04 D2 F5 1B

Transcient Key : 7A C7 4A 43 65 48 0E 21 68 66 2D A8 01 FB 29 37
                  C5 2A A2 3A 78 8F 85 24 F8 A2 26 03 CA 62 43 88
                  03 F3 9B 7D 1F D0 D0 95 DC 83 51 54 69 CB 96 0A
                  24 36 82 C4 80 68 A2 1C A4 E4 9E 2C A7 28 D8 98

EAPOL HMAC     : C3 D0 6C 14 EC B7 74 20 62 05 A0 55 88 38 E8 DB
```

Figure 11.11: A successful WEP key crack

Exercises

11.12 Are you familiar with null-byte on WonderHowTo.com? Get familiar. Go here:

<http://null-byte.wonderhowto.com/how-to/hack-wi-fi-cracking-wpa2-psk-passwords-using-aircrack-ng-0148366/>

Read this step-by-step tutorial. If your instructor can supply a wireless access point as a lab target, or you have your own, see if you can crack its key.

11.13 This article mentions another tool: coWPAtty. Find the linked article and read it. How does coWPAtty compare with aircrack-ng?



Cracked Up

For serious password cracking you will need some additional tools. One of these tools is your average graphics card, known as a GPU for Graphics Processor Unit. These cards are designed for repetitive tasks unlike a CPU, which is designed to all kinds of tasks. Even though your computer, smartphone, tablet and other devices have GPUs built into them, those chips are unlikely to be programmable by the user. They are locked chips that only accelerate the graphics for that device. The type of GPU needed to crack passwords (or mine bitcoins) are higher end gamer cards. A couple of these cards in a computer can crack passwords dozens of times faster but also costs more money.

You will want to take a look at either nVidia Ge-Force or Radeon cards made by AMD. They are similar in speed and price with best results obtained by using several mid range cards in tandem. Performance increases significantly when multiple cards are used instead of a single high end GPU. Some experts argue that the AMD cards are superior to the Ge-Force cards based on the usage of integer instructions in Radeon GPUs. It will be up to you to research which card offers the best password cracking benchmarks if you are going that route. This also applies to the device that you intend to use for your work.

Computers have ample computing power because they have plenty of power available. This isn't the case when it comes to laptops, smartphones, tablets or minicomputers like Raspberry Pis and Beagle Bone Boards. In these cases, you will need to do your homework to locate devices that already have these chips installed. Minicomputers allow you to add on capes, shields, or boards where you can customize your own GPU but power will still be a major consideration. High end gaming laptops have this same issue because they are built for power and speed, not portability. It is the same with any device running a graphic or computing intensive program, you will need plenty of juice. This can be overcome by having a hot swappable battery or a portable battery charger like the Anker. That thing can jump start a car.

Older computers can make great password cracking machines with a few low cost modifications. First of all, you will need to replace the power supply with a much larger one because we've been talking about power for the past three paragraphs. Next, the motherboard will need to handle several fullsize graphics cards, at least three, more if possible. You won't need to worry about the CPU or other hardware because this machine will only be used to crack passwords or mine bitcoins. Heat and ventilation will be another consideration but that can be taken care of by drilling holes in the case, adding cheap fans, clearing out unnecessary hardware, or installing a liquid cooling system. Above all, keep the computer in an area that has good airflow.

If your computer motherboard can't fit the GPU cards, you can always build an additional box (old shoebox or a wooden crate) and add PCI extension cables along with power cords to link them to your machine. Linux OS will be your best bet thanks to the highly customizable interfaces available but Windows will work too.

Hardware may sound unlikely for cracking passwords, but there are amazing ways to use devices in other ways. Video cards were designed to display better graphics than your CPU can, and free up the CPU for other tasks. An interesting thing happened on the way to better video performance: a group figured out that NVidia Cuda and certain ATI Radeon graphics cards would crack passwords. Of course, this required designing software for the purpose of hacking passwords, but it sounded like a great challenge.

Certain video cards provide amazing data acceleration to break strong encryption. A video card really is just one data accelerator after all. With a little tweaking and fine tuning, hackers have stacked (parallel) bunches of video cards together. Students at MIT brought this data acceleration to a whole new level when they added the computing power of the Sony Playstation. You read it right. Using a parallel pattern (like stacking columns of bricks instead of one single column), the students demonstrated a machine that could crack several encryption algorithms that were once thought too complex to break.

This is what hacking is really about, taking technology and making that technology better or creating something entirely new out of it. Real hackers are more like inventors, always



trying to tweak this or that and building new ideas or gadgets out of other electronic or digital parts. Too often the word “hacker” is used when people mean “criminal.” If a carpenter breaks into your house with a hammer, is he a carpenter or a criminal?

The Soft Side

Thanks to the ever increasing computing power of technology, password cracking has gotten much easier. People are often asked to provide an email address to create their username, which makes the first part of hacking an account that much simpler. Since most people use the same email account for all their work, knowing one email account means you probably know the username for every account that person has. Having the username, your next step is to figure out their password. We've talked about human conditions and the flaws of passwords where most people tend to use the same for every account they have. The average person has twenty five accounts.

One of the best software programs out there for password recovery is **oclHashcat** available at <http://hashcat.net/oclhashcat/>. This tool runs on both Linux and Windows and is aimed at either AMD or nVidia's graphics cards. Besides having control over almost every aspect of decrypting passwords, it also provides a watchdog to ensure you don't fry your GPU due to over temperature. Hashcat rightfully claims to be the fastest GPU enabled password cracker that is free. There are limitations such as it cannot crack Truecrypt 6.0 or above but other than that, the software has an amazing track record of recovering lost passwords.

New hash lists can be found dumped at Pastebin.com and other sites. The largest collection of plain text passwords was from a 2009 breach of an online gaming company called Rockyou.com. This is a database of 14.3 million passwords and can be found on several password collection sites. Hash lists have already been run against this massive collection and you can usually find both the password and MD5 hash list on the same site. If you look around long enough you will find over 100 million cracked passwords available online. One of them is probably yours.

Exercises

11.14 Do some research into the pros and cons of LSASS versus gathering hash strings using registry hives. Which method would you use if the target server was running Windows 10 server? Hint: DEFCON.

Build a Great Password

The best passwords:

- cannot be found in a dictionary (any language dictionary)
- contain numbers, letters and those odd symbols on top of the number keys
- contain upper and lower case letters
- are longer – literally, the longer the stronger
- can be remembered and not written on something that can be easily found



Feed Your Head: Cor Issues

Cor's a heavy thinker in the security world and doesn't take facts as fact. He ponders them. Then he ponders them some more. So when Cor writes you to tell you he has some doubts, you can learn as much from his thought process on breaking down a problem as you can about what he's doubting.

When Cor wrote us at Hacker Highschool about the Password Lesson draft, he shared with us he thoughts on passwords. (We stripped out the formalities to get to the grit for you because I'm sure you really don't want to read Cor remarking on Pete's strange trust issues with origami or Cor's feelings about Pete always wearing the exact same shirt in all his conference presentations for the last 10 years. Well, I guess we did tell you after all. Never mind then.)

So pay attention to Cor's process and what he has to say about password complexity and biometrics as passwords:

"Strong passwords are long. That's it. They have to be long. This is the main thing. Further, to allow for more complexity, you should allow as many characters in the character set as possible and no restrictions. The restriction reduces the keyspace."

This Leads to the Bad Idea of Complexity Rules

"It explains why required complexity reduces the password strength. In short, the requirements eliminate perfectly safe passwords because of lay-out. Now I have to say that complexity rules are invented for a reason. That reason is laziness (or stupidity or both). Many people choose weak passwords because they are easy to enter (123456, qwerty, qazwsx, 123qwe, 000000, etc.) or easy to remember (pet's names, kid's names, favorite brand, favorite sports team, dictionary words etc.). To prevent such "weak" passwords someone had the bad idea to require complexity in the passwords. Others had an even worse idea to like this and spread the idea. You can read the explanation below and do the math yourselves to find out that such rules eliminates thousands of billions of perfectly fine eight character passwords."

Bad Passwords are Still Better than Biometrics

"How many easy-to-enter or easy-to-remember bad passwords do you think exists? If you add all simple keyboard sequences, pet names, kid names etc. you can think of and add the English, Dutch and Pakistan word lists together, I doubt if you even reach as much as a million. So how about getting rid of complexity rules and allow all combinations, except the ones on the black list? That black list doesn't have to contain a million existing words and names, it'll do when it contains the top 1000 most used passwords in the leaked password lists that are widely available nowadays."

"Even if you only put the top 100 most used passwords on the black list, the chances that someone brute forces the right password is less than when you use biometrics. You probably all read recently that a phone's fingerprint reader can be fooled by latex fingers (reproduced from fingerprints left on a cup). Realize that affordable biometric equipment had an accuracy worse than 1% in 2011. I suppose that with nowadays technology this increased to 0.1%. This is still pretty inaccurate. A single try on a three digit password gives the same chance to gain access. Therefore today, biometrics are only useful as a second factor in authentication. This might change some time in the future when accuracy has raised to a sufficient level. I'm not sure if that'll happen during my lifetime (and I intend to live at least for another half a century)."

So what do you think about what Cor has to say? Is he right? Discuss this. Or don't and

just think about it.

Size Matters

The longer and more complex the password, the better the password. The best way to do this is to take several words that normally do not occur together to build a passphrase that you can easily remember. For example: This year, my favorite sports team has been playing especially poorly -- they're a "dog." Today, the builders started pouring the "foundation" for the house next door. This happens to be during the "summer." And perhaps I like to play "guitar." The passphrase becomes " DogFoundationSummerGuitar." Notice the mixed case, that is, both upper case and lower case letters.

Now let's add some more complexity by adding some characters and substituting, so there are no "dictionary" words. If we use digits 0 - 9 and special characters (shift-numbers), we can do substitution like zero-for-capital-O. Digit 1 for letter "l." @ or 4 for letter "a," either case. And, to make it better, change the rules for substitution. The first time there is a character "a," use "4." The second time, use "@." And so on. Mangling the passphrase this way, we might end up with:

```
g#ounD@t!on$umM3r&u17ar.
```

If that's too hard, you might split up the words and add numbers and punctuation marks, like this:

```
!Dog#Foundation$Summer&Guitar*.
```

Some people develop a reasonably complex passphrase they use every time they need a password, but to make it unique to that system, website or application, will prepend (add at the beginning) something related to that website. For example, you might want a long password for Facebook. People post all kinds of things to Facebook; they're a bunch of clowns. So:

```
Clowns!Dog#Foundation$Summer&Guitar*.
```

People tell you "don't write your passwords down." Unfortunately, you quickly have more passwords than you can remember. So: get a good encryption program. Choose a decent passphrase for encrypting this collection of files. The passphrase should not be similar to or correlated with your other passwords. Keep an encrypted text file with your user names and passwords. Only open it when you need to remember a password. Keep an encrypted backup of that file somewhere (like a USB drive), so if your disk crashes, you don't lose all your passphrases.

And don't call your password file "Passwords." Give it a different name like "Grocery List" or "urinary tract infection dates." Just don't call it passwords.

Or just use a good password manager.

Other Methods

There are many password generators available on the Internet, but these will generate a nearly impossible to remember password.



Try instead to use a seemingly random string of letters or numbers that you can easily recall.

For example:

gandf3b! (goldilocks and the 3 bears!)

JJPL2c1d (john, jill, paul, lucy, 2 cats, 1 dog – the members of your household)

Or you could create your own algorithm to generate passwords. An algorithm is the designation of a step by step procedure for a calculation. Here's an example for you.

This algorithm for choosing a strong password will be the following:

- The last 2 digits of a birth year
- The first 3 letters of a first name, with the last one being in uppercase
- the symbol '!'
- 3 letters to define the service you could registering in

So, that is the algorithm. Let's build a password for Facebook and Gmail, shall we?

A password for Facebook, following this algorithm would be: 84maR!Fac

The password for Gmail, following this algorithm would be: 84maR!Gma

Easy? That way we just created a 9 character password that contains numbers, letters (lowercase and uppercase) and special characters.

This way you have strong passwords and a different one for every service. Of course, everything has a weakness. The password is only as strong as the protection for the algorithm and the length of the characters. If someone discovers your algorithm, it would mean that all your passwords would have been compromised or potentially stolen.

Search online for password managers. You have several free options that you can try out. The password managers are computer applications that you protect with a master password, and can keep all your other passwords in there so you don't have to remember all of them. That way, it's easy to have strong passwords for everything that is important.

Check, Please

So, you think you have the best password ever created. Your password will withstand ten thousand years of attacks because you built it using the rules we've outlined above. You're a great student of Hacker Highschool, so you applied your knowledge to make an unbreakable password. Before you celebrate, you want to see how well it stands up to a password checker.

Like anything built for quality, you need to test your great creation. How do you do that? You can beat on it with Javascripts that you can trust to apply maximum pressure to your device. Below you will find one of the best password checkers available, and it's free. Go ahead, don't be shy. Hey, why are sweating so much? You look a little nervous.

One key to being a successful security professional is being able to test your own designs. Try and break into your servers or web pages. Locate the weaknesses and correct those vulnerabilities. The same principle applies to passwords. You have to test everything to its limits, no holding back, right? Other attackers aren't going to hold back either so why should you?

Password Strength Checker: <https://www.microsoft.com/security/pc-security/password-checker.aspx>

Password checker

Your online accounts, computer files, and personal information are more secure when you use strong passwords to help protect them.

Test the strength of your passwords: Enter a password in the text box to have Password Checker help determine its strength as you type.

Password:

Strength: Strong

Figure 11.12: Microsoft's free password checker

**Have you figured out why NOT to use free online password checkers?
Because you GIVE YOUR PASSWORD AWAY to them!**

Change Up

Unlike your socks, passwords don't need to be changed on a regular basis. Evidence shows that regular changing is a myth and does not provide better security. Since the attack surface of anything increases the more a human is involved in the security, changing passwords more frequently makes the attack surface increase with frequency. Changing passwords is only a good thing if you use the same password in multiple places. But you're not stupid enough to do that. Are you? No. I can't believe that. Besides that, it's a whole lot easier to use different passwords for different places than it is to locate a clean and matching pair of socks. Look under the bed and you will find that other sock. Trust us. Move the cobwebs out of the way and those old books, you'll find that one sock next to the dirty sandwich plate under your bed.

Note: Passwords and accounts for ex-employees should be removed (revoked) on the same working day as their departure from the organization. This doesn't always happen. If you were a contract penetration tester, this would be one of your prime opportunities!

Exercises

- 11.15 Create a strong password, **that you could remember**, and that scores well at the following web page: www.passwordmeter.com
- 11.16 Look at the Web pages for three different banks and find out what type of password is needed to allow an account holder to access restricted information. Do the banks also offer recommendations that would lead users to create strong passwords?



Conclusion

It has always been difficult to bridge privacy and confidentiality with ease of use for any form of communication. Remember that the Internet was never designed with security in mind. The Internet was design for freedom of information flow to be its primary purpose, even in the event of a failure within the network. Fail-safes were built in to ensure data continued to reach its destination even in the event of a nuclear war. This massive network was built back in a time of flowers and peace signs everywhere but the Soviet Union and United States posed a serious threat to the world. It was called "Mutually Assured Destruction" or MAD because of the nuclear arsenal both countries possessed. But that is another story for another time.

Passwords were introduced when people wanted some sort or privacy on their digital communications. It was an easy and inexpensive band-aid to apply in Unix environments. The young Internet was built to connect university academics, researchers and DoD agencies who funded that research. A username and password allowed a certain amount of security to this fundamental network.

Over time, other commercial organizations and private entities plugged into the Internet, thus expanding the capabilities of this data sharing network. Each company wanted to keep their data separate from prying eyes so passwords were also introduced because they were already part of the Internet culture.

As far back as the 1960's academics and researchers brought up the vulnerabilities of passwords and many advocated for stronger authentication measures. Passwords were inexpensive and easy to use. Year after year password use never faded as it was supposed to. Most security experts fully expected passwords to be replaced with a stronger method. As you can see, that still hasn't happened. The culture of insecure practices continues because it's easier than fixing the problem.

Some powerful organizations have spent millions trying to devise ways to replace passwords with other systems. IBM created a database of personal questions that could be asked before access would be granted. Microsoft worked on using pictures that the user would recognize to gain account privileges. One company came out with voice recognition that was easily tricked using phone conversation recordings. Biometrics has been the Holy Grail of password replacement even though each attempt has been hacked using simple techniques.

In this lesson we showed you lots of information about passwords and how they work (or don't work). Words like "entropy" might not mean much to you at the moment but you will heard it again in your career as a security professional. Hacker Highschool is all about getting you prepared for the real world of cyber security the ISECOM way. You were shown the inner workings of CAPTCHA and how it can be bypassed. Multifactor authentication is another large word thrown around by security vendors to sound cool but it just means there are a couple of different ways to prove trust.

Hacker Highschool expects you to be asking questions about the world around you. We demand that you poke around and finker with things. Questions you should be asking when it comes to passwords include:

How are passwords stored?

Where are they stored?

Are they transmitted or is just the hash transmitted?

Are default passwords hardcoded into systems?

Is there a more intelligent method for brute force password attacks?

Do passwords reset when a system is upgraded or updated?

Be curious about how things have always been operating. Don't expect a book or the Internet to have all the answers. Sometimes you'll have to learn the hard way, by making a mistake. That is okay, though. Experience is gained through mistakes.



Throughout this lesson you were peppered with lots of **Feed Your Head** segments. This was done to highlight some of the highly technical (or really stupid) aspects of passwords. either way, you should have learned something new. We also discussed biometrics, attacks, hacks, habits, key functions, rainbows, wifi cracking, key sizes, hashes and all kinds of other words not usually found together.

Depending on how passwords are implemented, they can either be complex or rudimentary. It is your job to make systems more secure but not more difficult for the user. This isn't an easy task but one we know you are ready for.

Today's teens are in a world with major communication and productivity channels open to them and they don't have the knowledge to defend themselves against the fraud, identity theft, privacy leaks and other attacks made against them just for using the Internet. This is the reason for Hacker Highschool.

The Hacker Highschool project is the development of security and privacy awareness learning materials for junior high and high school students.

Hacker Highschool is a set of lessons and a practical means of making hackers. Beyond just providing cybersecurity awareness and critical Internet skills, we need to teach the young people of today how to be resourceful, creative, and logical, traits synonymous with hackers. The program contains free security and privacy awareness teaching materials and back-end support for teachers of accredited junior high, high schools, and home schooling. There are multiple workbooks available in multiple languages. These are lessons that challenge teens to be as resourceful as hackers, including safe Internet use, web privacy, researching on the internet, avoiding viruses and Trojans, legalities and ethics, and more.

The HHS program is developed by ISECOM, a non-profit, open-source research group focused on security awareness and professional security development and accreditation.