Cracking the Code: A Guide to Password Security

Section 1 - Introduction

Where are passwords stored?

When we talk about cryptographic attacks, the easier way to interpret this is cracking passwords. Now, don't get too excited. First, let's ensure we understand what we're talking about when we talk about password cracking. A well-designed password-based authentication system doesn’t store a user’s actual password. This would make it far too easy for a hacker or a malicious insider to gain access to all the user accounts on the system. When you have some kind of server a web server, an FTP server, an SSH server, a game server, an operating system that's sharing folders, it doesn't matter what it is you are going to have to have a list of usernames and passwords stored somewhere on that server system. Now, if you're going to store them, which you must do, then when someone logs in, they're going to type in a username and password and then come to that server.

What is password hashing?

You have to store the password. So how do you store it? Well, you could just store it in clear text. You could literally have a list somewhere on your hard drive that says, Username: John Doe and then whatever his password is, and another user and whatever his password is, and another user and whatever her password is. We could do that.

But the downside is that if a bad guy gets to that server, he could get easy access to our passwords. So traditionally, what we do with a password is that when we create a new user and have them type in a password, the password is never stored on the hard drive, we just hash it. We make a hash of the password, which is the result of sending the password and a random value called a salt through a hash function. Hash functions are designed to be one-way, meaning that it is very difficult to determine the input that produces a given output. Since hash functions are also deterministic (meaning that the same input produces the same output), comparing two password hashes (the stored one and the hash of the password provided by a user) is almost as good as comparing the real passwords.

Now, if we've got a hash of the password sitting on the server and somebody who's a client wants to log in, the server is going to say, please type in your username and password.

So, they type in a username and password on their side, and then that is hashed. So, the hash comes over the internet and then gets to the server. Then the server compares to the hashes, and that's how it logs in. We would never use clear text except in the most primitive of situations.

Hacking passwords means hacking hashes

The important thing to understand here is that if you want to get into cryptographic attacks, if you want to hack passwords, what you're really doing is hacking hashes.

There are a couple of things that come into play here. Number one, you have to be able to get to that list of hashes. One of the hardest jobs of cryptographic attacks is how you get to that server and how you grab those usernames and password lists. You don't know what the passwords are, but how do you at least get the list? That varies for each one that's out there.

If you want to get your Windows system, it has its own set of passwords and hashes. If you want to get to an FTP server, it depends on the brand. They have their own usernames and passwords. The biggest part of cryptographic attacks really isn't hacking the hashes, the biggest part is getting to them.

You can’t reverse a hash

The second thing we need to talk about is that if the password is stored in a hash, there is no way for you to reverse that hash to figure out what the password is. It's just not going to happen.

What we're going to do instead is we're going to generate hashes until we get the hash that we have a copy of, and now that we have the copy, we know what this hash is because we generated it ourselves and then we know what the password is.

Password cracking refers to the process of extracting passwords from the associated password hash. This can be accomplished in a few different ways:

Dictionary attack: Most people use weak and common passwords. Taking a list of words and adding a few permutations like substituting $ for s enables a password cracker to learn a lot of passwords very quickly.

Brute-force guessing attack: There are only so many potential passwords of a given length. While slow, a brute-force attack (trying all possible password combinations) guarantees that an attacker will crack the password eventually.

Hybrid attack: A hybrid attack mixes these two techniques. It starts by checking to see if a password can be cracked using a dictionary attack, then moves on to a brute-force attack if it is unsuccessful.

So when we're talking about cryptographic attacks and in particular we're going to talk about brute-force attacks, dictionary attacks, rainbow tables and all that kind of stuff. keep in mind what we're doing more than anything else is generating hashes and making a comparison. When we compare the right ones, we can finally say that we have the password.

Section 2 - Implementation

Finding list of hashes in FreeSSH

To go through this process and the best way to do this is to pick an arbitrary server to attack. In this case, I'm going to use a program called Free SSH. Free SSH is a wonderful little SSH and Telnet server, nothing special about it, but one of the things that's kind of fun is that it's got these user accounts.

So I'm going to add a user called John Doe, and there are all these different ways I can store stuff. I could use NT, which is the Windows operating system that's on. In this case, I'm going to use a password stored as a SHA1 hash. I'm going to give it a dangerously simple password, and I'm going to call it j-o-h-n, all lowercase. Do not try such a small password at home.

Now I've got this John Doe in here, and I want to go ahead now, and I want to first of all, I have to figure out where this John Doe password is. To do that, all I've done there is got Free SSH to save that particular one. I did some research for this old program and found some documentation that says all the passwords with the SHA-1 hash are stored in a little file right here (C:\Program Files (x86)\freeSSHd\FreeSSHDService). So I could open this file up and scroll it around. Here, I can see some other user accounts. But here's the John Doe account right here, and that is the actual hash that is storing that password of j-o-h-n.

Using Cain and Abel for a brute-force attack

Now that I have the hash, I need some tool that I could take this hash value and throw it in and say, keep running a bunch of hashes until you find one that matches that. And that process, which we call a brute-force attack, can be done in all kinds of different ways. Now, for this one example, I'm going to use an old program called Cain and Abel.

Now, I need to warn you a couple of things about Cain and Abel before we get started with this. First of all, Cain and Abel is a very, very powerful tool, but it's very dated, so even though I'm running a modern Windows system here, there are a lot of features of Windows that really just don't come into play anymore.

The other thing is that anytime we talk about cryptographic tools like this, they're not instantly easy to use. It would be kind of like someone saying, "Hey, let's go ahead and make an accounting spreadsheet," and I hand you Excel. Sure, it's a good tool, but you really have to understand what's going on. So, there are a lot of steps in here that I'm familiar with because I’m familiar with this tool, but you have to do a little experimentation on yourself.

So we take a look at this, and there is a cracker function right here. You see that, and it says, what do you want to crack? These are different kinds of hashes because that's mainly what we're hashing in this world. So I know this is a SHA-1 hash because that's how the Free SSH store stuff.

Here's my SHA-1 hash tool. What I'm going to have to do, first, is go over, and I'm going to grab this hash. I'm just doing a regular old copy, and I need to put it into the cracker. And what I've done now is I've inserted this SHA-1 hash into it.

So now, let's go ahead and start cracking. So what we're going to do first is we're going to do brute force. We're basically going to say, Cain and Abel, I want you to start with the letter "a," make an SHA-1 hash, make the letter "b," make a SHA-1 hash, go through all those, then do "aa," then do "ab," then do "abcd." Get the idea that this could take a little bit of time? Well, it absolutely does.

So what we're going to do is a brute-force attack. Now, you'll notice that I've got a lot of options here, and all of these crackers have some tool like this, so it's going to say, just use lowercase and numbers. Now, for the sake of brevity, what I'm going to do here is I'm going to make it even simpler than that, and I'm just going to say, just use lowercase letters.

Cracking the simple password “john”

So what I've done here is I've got it knocked down to a maximum of eight characters, and let's go ahead and start it and see what happens.

If you take a look right here, it was pretty much instantaneous, but you'll see it found the password is john. So that is one example of brute force.

Now keep in mind, one more time, what brute force is doing. It's literally generated based on the predefined character set that I set up for it. I said, let's start with just the letters of the alphabet and just lowercase. And then it ground through them. So you can see that it went through just about a trillion iterations in a very, very short amount of time.

Using Cain and Abel for a dictionary attack

So what we want to do is to go ahead and do an attack, but let's make some assumptions. One of the things we know about people is that I don't think I've never met anybody who used a password that was 12XF9L&2, right?

What we do as human beings is we tend to use dictionary words — John47, Timmy22 and 1Johnny5 — and we turned all the O's into zeros, and you know all that stuff. Well, if we know that, we can do another kind of attack called a dictionary attack.

A dictionary attack starts by using a text file that is filled with dictionary words. It'll take those dictionary words and then it'll manipulate them. For example, if I put the word john in the dictionary, I could tell the cracker to go, don't do just john, but do capital J-O-H-N and then make JOHN1, JOHN2, JOHN47, all that type of stuff.

So a dictionary attack will always, always start with a text file that's full of dictionary words. So let's try a dictionary attack. All right, so let's go ahead and grab that hash one more time, and I'm going to go ahead and plug it in here. So here's my hash.

Now, the whole idea behind a dictionary attack is we have to feed the attacking tool a dictionary. So I have a very simplified one. If you take a look right here, I've made a little file called dictionary.txt, and you could see that I have all of about what, nine words in there.

Keep in mind that you can download dictionaries from the internet that have hundreds and hundreds of thousands of different words in there, so dictionaries can be massive, massive devices.

Let's go ahead and do the dictionary attack. I'm going to go ahead and select Dictionary Attack this time, and you'll see that I've already preselected that text file.

Now, this is a pretty handy tool because he always remembers where you left off, and I've done an attack before. He remembers I'm at the end of the file, so I got to go through this little process to tell it to go back to the beginning.

What I'm going to do is start it, and boom, you can see, almost instantly, I got the answer. Now, I made this one easy because the password is just four characters and they're all lowercase alphas, so for brevity, it works out well. But let's look with a little bit more detail here.

You'll notice that you tell these crackers how to deal with the words. So for example, here's one where it says if the word is in all lowercase, then try it also as uppercase or vice versa if the word in the dictionary is all uppercase.

Here, I could say, do case permutations. Now watch if I click that. It turns a couple of these off because now it's saying, change just the second letter to uppercase, change the third letter, or however that might be.

And the other one right here at the bottom might be familiar to some of you guys, which is to add two numbers to the end. So whatever the word is, if it's john, do john1, john2, john3, john4, all the way up to john99.

Using online sources for a cracking easy password

Even though there are different password cracking tools available there are also some websites which do the same if provided a hash of a password. These are quite beneficial to use rather than the traditional tools if the password is a commonly used one or simple password. For example, if we look at a website called CrackStation. Here we can give it with multiple hashed passwords it will use it's massive pre-computed lookup tables to crack password hashes. These tables store a mapping between the hash of a password, and the correct password for that hash. The hash values are indexed so that it is possible to quickly search the database for a given hash. If the hash is present in the database, the password can be recovered in a fraction of a second. And there are similar websites to this and the downside of this is "This only works for "unsalted" hashes".

Here in the case of John Doe with the help of CrackStation I was able to crack the password (j-o-h-n) in a fraction of a second. So when we have hashes that are similar to the password as the above we can use these websites to weed out the easy ones and focus more on the tougher passwords.

Section 3 - Conclusion

I wonder how many of you guys out there said, "Oh yeah, this thing would probably crack my password," based on that. So dictionary attacks are fantastic. They speed the process up simply because they take advantage of the fact that human beings tend to use words they're familiar with as part of their password. At the top of every one of these dictionaries is password and 1, 2, 3, and 1, 2, 3, 4. So don't even bother with those. They'll have you hacked in milliseconds literally.

Why you should a complex password

Let's take a look at this one more time, and imagine this time, let's say I had a big, complicated password.

So let's change the predefined character set, so it's going to be lowercase alphabet, uppercase alphabet and numbers. Do you see that right there? Now, watch the keyspace as I start to bring it up. You see that? I'm already up to exponential notations.

So that is a really good example of why we use complex passwords. We use complex passwords to make cryptographic attacks harder.

That's one example. In this particular example, what we did is we simply ran a brute-force attack. As you can see, when things get complicated a brute-force attack can become incredibly onerous. Now, this is just a regular, middle-of-the-road desktop system. If I wanted to, I could buy computer systems or build one myself that use graphics processors and all this extra power, and they can calculate a lot faster, but it still becomes very difficult.

How to create a password that's hard to crack

There are many password-cracking tools available in the market. These tools try to crack passwords with different password-cracking algorithms. Most of the password cracking tools are available for free. So, you should always try to have a strong password that is hard to crack. These are a few tips you can try while creating a password.

- *The longer the password, the harder it is to crack*: Password length is the most important factor. The complexity of a brute force password guessing attack grows exponentially with the length of the password. A random seven-character password can be cracked in minutes, while a ten-character one takes hundreds of years.
- *Always use a combination of characters, numbers and special characters*: Using a variety of characters also makes brute-force password-guessing more difficult, since it means that crackers need to try a wider variety of options for each character of the password. Incorporate numbers and special characters and not just at the end of the password or as a letter substitution (like @ for a).
- *Variety in passwords*: Credential stuffing attacks use bots to test if passwords stolen from one online account are also used for other accounts. A data breach at a tiny company could compromise a bank account if the same credentials are used. Use a long, random, and unique password for all online accounts.

What to avoid while selecting your password

Cybercriminals and password cracker developers know all of the “clever” tricks that people use to create their passwords. A few common password mistakes that should be avoided include:

- *Using a dictionary word*: Dictionary attacks are designed to test every word in the dictionary (and common permutations) in seconds.
- *Using personal information*: A pet’s name, relative’s name, birthplace, favorite sport and so on are all dictionary words. Even if they weren’t, tools exist to grab this information from social media and build a wordlist from it for an attack.
- *Using patterns*: Passwords like 1111111, 12345678, qwerty and asdfgh are some of the most commonly used ones in existence. They’re also included in every password cracker’s wordlist.
- *Using character substitutions*: Character substitutions like 4 for A and $ for S are well-known. Dictionary attacks test for these substitutions automatically.
- *Using numbers and special characters only at the end*: Most people put their required numbers and special characters at the end of the password. These patterns are built into password crackers.
- *Using common passwords*: Every year, companies like Splashdata publish lists of the most commonly used passwords. They create these lists by cracking breached passwords, just like an attacker would. Never use the passwords on these lists or anything like them.
- *Using anything but a random password*: Passwords should be long, random, and unique. Use a password manager to securely generate and store passwords for online accounts.

Section 4 – References

- https://www.infosecinstitute.com/
- https://crackstation.net/