RUHR-UNIVERSITÄT BOCHUM

Improved guessing of composite passwords

Norbert Schmitz

Master's Thesis. March 13, 2012.
Chair for Embedded Security – Prof. Dr.-Ing. Christof Paar
Advisor: Dr. Markus Duermuth

Abstract

Finding new ways to crack passwords is important for the field of IT security, since it is important to be always one step ahead of the attackers. Some researchers argue that choosing a sentence as a password is a secure alternative. These passwords are secure against the currently known attacks. To find a different approach on cracking these passwords we are looking how users are combining their passwords from dictionary words and how we can attack these passwords. We will show that a small dictionary with our approach is more efficient than big dictionaries. Further we will give a thorough overview on password security and how to mitigate attacks on password systems.

Declaration

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, dass alle Stellen der Arbeit, die wörtlich oder sinngemäß aus anderen Quellen übernommen wurden, als solche kenntlich gemacht sind und dass die Arbeit in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegt wurde.

Norbert Schmitz

Contents

1 Introduction

Passwords are part of the authentication system of modern information systems. Ross Anderson described them [And08] as one of the four way to identify that a user is who he claims to be. To authenticate means to show

- who we are – biometric information,
- where we are – a physical location,
- what we have – a token, and
- what we now – a password.

Passwords most likely are the cheapest way to implement authentication – usually its just a view lines of codes. Therefore it has become the most common way of authentication. Everyone uses passwords several times a day: to login to his computer, to get money from an ATM, where it is called PIN, to check the mobile phone mailbox, and in thousands of other ways.

Thus, passwords probably are the most widely used way to authenticate to a machine, but unless a machine, we humans are not always good in memorizing things over a long time. Nearly everyone forgets over time passwords he rarely used. If such a situation happens for a system where it is impossible to reset a password without knowing the old one. A user can only try to crack the password by brute force or a dictionary attack.

But not only users are interested in cracking their own passwords. Also the law enforcement organizations have a vital interest in cracking passwords. In 2008 the Brazilian federal police raided the home of an Brazilian banker and investor [Sof]. They seized several hard-disk which had been encrypted. The experts of the Brazilian policy tried to crack the passwords of these disks. After about 5 month of unsuccessful trying, they asked for help from the FBI. The FBI tried to crack the password for the next twelve month and had to give up at the end. The disk was encrypted by TrueCrypt, an open source disk encryption system, using AES-256.

In another case [Ars], Ramona Fricosu was sued to give the password for her GPG encrypted laptop to the authorities. This started a controversial discussion, if giving the password is similar to incriminate them self. At the end the judge did not need to make a decision about this case. The police was able to guess the password after a co-worker named several possible passwords, One of them opening the encryption.

This leads us to the question, if it is ethical justifyable to research the cracking of passwords. If we learn how to crack passwords, we are able to give the users hints, which passwords are good and which are bad. This will improve the security not only

for individual users but also for companies. If we are ignoring this part of the security, we will give the bad guys some advantage since they actually do not care about ethical questions.

During this work we will look at the graduate thesis [Mat10] of Matt Weir, which also has been published in this paper [WAMG09]. He describes a method how to manipulate a dictionary in the way that the probability to crack passwords increases. To archive this, he analyzes samples from a password leak, learning the patterns users are using to create their passwords. With these patterns he manipulates the words from the dictionary, enhancing the number of hits he gets in a test set. One huge disadvantage in his approach was that he still needed to have the words itself inside his dictionary. If a password would be "iloveyou" he was only able to crack it if this phrase was already in his dictionary. This work engages with a approach to circumvent the drawback. While Weir was only using words from a dictionary, we are first going to enhance the dictionary by itself, by combining its words. Therefore we are analyzing patterns of how words are put together and what role the length of the words are playing in the generation of passwords. After we improved the dictionary we will take the approach of Weir to manipulate the new dictionary in the same way as he did.

1.0.1 Overview

In Chapter 2 we will start with a discussion of passwords in general. We will learn how passwords are handled in different software systems and will discuss how attackers are getting and cracking passwords. In Chapter 3 we will discuss what composite passwords are, how we are creating them and which roles leet speak will take. Chapter 4 describes the experiments we did and present the corresponding results. In Chapter 5 we will present some ways of protection for users, developers and administrators. We will take a look on related work in Chapter 6 and end with our conclusion in Chapter 7.

1.0.2 Ethical consideration

During this thesis we are analyzing leaked passwords. All passwords used in these thesis have been already leaked for a long time. To not harm any users all personal data in the original leaks has been removed before we used them. The password files were taken from [Sku]. For our research on leaked passwords we extracted data from [Pasb]. Pastebin is a site where user can publish text files anonymously.

2 Background

If we want to discuss password guessing we first need to know what passwords are and how they are used in various software systems. Checking the way how attackers crack passwords online and offline will help us to understand how we can protect us against various attacks.

2.1 Passwords in general

Passwords are not words as a linguist would define them. A password is sequences of characters from an alphabet Σ of allowed characters. A password is often limited in the length, defined by the system where it is entered. When entered correctly, a password allows access to a system and/or data. We differentiate between one time passwords and passwords which are used over a certain not necessarily limited amount of time (longterm passwords). For this work we are considering only long term passwords.

Longterm passwords are ofter chosen by the user, where the system to be accessed defines some rules about accepted passwords. A example is the Maestro Card, in Europe also known as EC card. To access an ATM with these cards a user is provided with a PIN, a so called personal identification number. The PIN is a 4 digit number, where in former times the rules stated that the first number must not be zero. In modern days some bank institutions allow their customers to choose the PIN by them self. Using a PIN the key space is the numbers from 0000 to 9999, giving 10000 possible PINs. Since the allowed number of wrong entered PINs is usually limited to 3, an attacker has a chance of 3/10000 to guess the correct one. After 3 consecutive wrongly entered PIN the card is revoked and the attacker has no further chance to access the account.

The human memory is limited to remember lists of seven plus or minus two items [Mil94]. In our times users have to remember a lot of passwords for different applications and websites. If a user uses the same password for more than one site or application we name this password re-usage [FH07]. Re-used passwords are a big security issue. Once a password has been hacked or leaked, an attacker will try the same password for various further sites and applications.

2.2 Handling passwords in software

Software needing an authentication mechanism often uses passwords. Passwords are amongst the simplest and cheapest ways to use an authentication mechanism. Usually it is some lines of code to get this authentication mechanism to work, but it is a very bad idea to put the password inside the code. On one hand a reverse engineer is able to

extract the password, on the other hand changing the password needs a recompilation of the code. Keeping this in mind we will have a look into some implementation of password mechanism in different software systems.

2.2.1 Windows passwords

Windows systems used the LanMan Protocol until Windows NT. The length of the password can be up to 14 characters. Upper case characters are automatically converted to lower case. If the password contains less than 14 characters the leftover characters are filled with zero bytes. The password is divided into two parts each of them containing 7 characters. Each part is used to encrypt the word "KGS!@#$%" with DES. In this case DES is used as a one-way-function to "encrypt" the password. The result is stored as a reference to the password. At first the implementation looks robust, since DES is already used for a long time and it is used as a one-way-function. The flaw nevertheless comes with the implementation. Since the password is divided into two parts the complexity is drastically reduced. An attacker can divide the password representation into two parts and attacks each part separately. Since a single part contains only 7 characters of the password, a brute force attack is feasable (see Section 2.3.3). The second problem is that the uppercase characters are converted to lowercase, by this the number of possible characters is reduced by 26. Altogether it is no longer recommended to use the LanMan for storing passwords. Since Windows Vista the LanMan protocol is turned off by default. Since Windows 2000 the Kerberos protocol is in use to communicate to Active Directories. On Kerberos the attack does no longer work.

2.2.2 TrueCrypt

TrueCrypt is a disk encryption software. It has the ability to encrypt a whole disk, or partitions. Additionally it can create container files, which can be mounted as if they were a partition. For simplicity we will talk only about disks, but the same methods apply to file containers and partitions. To mount a disk the user has to provide a password, different from other software the password is not stored on the disk. Instead it is given together with a 512 bit salt to a key derivation algorithm. The key derivation algorithm creates a key out of it, which is used to decrypt the master key of the disk. To protect the system against brute force and dictionary attacks the key derivation algorithm uses one out of three functions with an iteration between 1000 and 2000 times. Since the key derivation is used only once per mounting, it enhances the time needed for mounting only slightly but it enhances the time for brute forcing or dictionary attacks by the factor of 1000 to 2000.

2.2.3 Web-sites

Web-sites are using passwords in several different ways. While some sites, especially sites using a content management software, use password mechanisms only to grant access to an administration system, others are using them to provide access to the system with personalized data. A good example for the latter are web-mailing systems or sites like

facebook. All of them have one thing in common, they have to store the password in a secure way. There exist basically three ways to store passwords:

- Clear text,
- hashed – passwords are secured by a cryptographic one way function, and
- hashed and salted – before securing the password with a cryptographic one way function the password is extended by a random string, which is stored together with the secured password.

Not only since the attacks on Sony in 2011 we know that web-sites often contain vulnerabilities. An attacker who gets access to the database or file with the stored passwords, is immediately able to use these passwords against other systems. In these days password reuse is still a common behavior by users. Therefor the chances are high that the attacker would be able to use at least some of the credentials he found against other sites, like the web-mailer of the user or sites like Ebay, Amazon or Paypal.

Storing password in clear text enables an attacker to directly use the found credentials against these sites. A user would have no chance to change the passwords on site, where he used the same password.

Hashed passwords instead have to be decoded before they can be used against other sites. To decode hashed passwords, an attacker can use rainbowtables, search engines or attacks like brute force or dictionary attack. Only very strong passwords, which can not be found in dictionaries and which have a certain length are protected for a longer time against usage by the attacker.

Password which have been hashed and salted have the strongest protection against an attacker. Rainbowtables and search engines do not help in this case, since the effort would be to high to store all the possible passwords which would lead to the salted hash found in the leak. However brute force and dictionary attacks are still feasible, but are much slower. In pure hashed password an attacker only needs to hash a possible password once and compare it to all found hashes, on salted hashes he has to do the computation for each and every salt. With this method the complexity is multiplied in the worst case by the number of users giving the users more time to change their password on other sites.

2.2.4 WPA and WPA2

WPA and WPA2 are encryption schemas for W-LAN. They provide security while communicating over the air. In private environments they are used together with pre-shared-keys (PSK). These keys are similar to passwords and allow a user with a mobile device to login to a network. The login process takes several seconds which makes it impossible for an attacker to simply guess the password. There are currently two known attacks against WPA and WPA2. The first is to wait for an user to authenticate in the network and to sniff the handshake. During the handshake the password is used for

encryption. The attacker can try to guess the password and check if he is able to decrypt the handshake.

The second is only applicable against some newer access-points. The attacker uses a mechanism which was introduced by the Wi-Fi Alliance in 2007 and is called WPS. WPS communicates the used PSK to clients which can authenticate them self to the access-point with a PIN. This PIN consists of two block of 4 digits. The last digit of the second block is a checksum. By measuring the response time of an accesspoint the attacker can validate each block. Due to these flaws the attacker needs to test a maximum of 11000 PINs. Depending on the signal quality an attacker can break into a accesspoint within ten hours. Once an accesspoint is cracked with this method even a password change of the PSK has no effect on the attacker. He simple requests the new password with the known WPS PIN.

2.2.5 Default passwords

Default passwords in software are a huge problem, as soon as the systems are connected to the Internet. A lot operators simply forget to change the default password to one of their choice. Sometimes the password change functionality is to much hidden inside the software to find it easily. Sometimes in big companies a lot of administrators have to access a system and the password is not changed on purpose.

On the Internet an attacker can easily find these default password list by a simple google search. One of the many sites is Cirt [Cir] where the passwords are sorted by manufacturer. Attackers can easily combine this information with a google search to find devices connected to the Internet and providing a login screen.

Another way to find devices on the Internet beside google is Shodan [Sho]. Shodan is a site set up specially for the purpose of finding devices online by a special keyword. This site has crawled a lot of IP addresses and stored the HTTP header. This header usually is sufficient to identify a device.

2.2.6 Software flaws

Beside default passwords there is another threat, which is hard to counter. Some companies have flaws in their system which allows the attacker to circumvent a given password. Usually these threats are encountered only by accident. On February the 6th 2012, a hacker encountered such a flaw in the web-cams from Trendnet [Heia]. The user someLuser examined the firmware of a web-cam TV-IP110w and found that using a certain URL he was able to see the stream of the camera even without login in.

On January the 25th Heise published an article about a research from HD Moore about video conferencing systems where the administrators put the system on auto-answer, to make video conferencing more easy [Heib]. This gets even more critical since most of these systems are located in meeting rooms. An attacker knowing the IP of the system could easily participate in such a meeting without the knowledge of the other people.

2.3 Attacker models

For trying to break passwords we will first look at the two main categories “online” and “offline”. There is an overlapping as soon as it goes to guessing, on the other hand the offline part begins as soon as passwords have been leaked.

2.3.1 Online

Online attacks describe attacks against targets which are online. The victims are either online services, where an attacker tries to break in by flaws in the system, or users trying to login.

Phishing

Phishing attacks target the user instead of the web-site. The attacker tries to entice a user to a web-site, the user assumes to be valid, but being under control of an attacker. To convince the user to visit such an phishing site, usually social engineering techniques are used. Social engineering is the art of manipulating someone to make him want to do what we want him to do. One of the many ways to convince a user to visit a site is to tell the user that there is something wrong with his account, but it could be fixed if he logs into it. These websites are nearly undistinguishable from the real web-site the user wants to visit. As soon as the user enters his credentials, like user name and password, the data is stored on the site of the attacker. Depending on the skill of the attacker, the user gets either an error message or the site logs in with the credentials of the user and forwards the data from the real site the user wanted to visit. In all cases the credentials are compromised to the attacker.

Attack by proxy

In this model the attacker gets control over a proxy the user is using. A proxy is a program or server which is in between a browser and the Internet. The proxy gets the requests from the browser and forwards this to the site the user wants to use. Some of the proxies are used for privacy reasons. In these cases the proxy removes information the browser is sending and removes this, to protect the privacy of the user. Other proxies are used to hide the origin of the user. The live stream of BBC, for example, allows only access from within the UK. Therefore it is not possible for users on a business trip to Germany to watch the live stream from his hotel. The use of a proxy within UK would allow a user to still watch the live stream.

Other cases of the use of proxies are the TOR network. In this case a proxy is not necessary. An attacker controlling an exit node in the TOR network, can nevertheless force the use of an proxy.

In all these cases the goal of an attacker is to gain access to the proxy. As soon as he succeeds he can listen to the data the user is sending to a web site. If a user sends credentials over such a proxy the attacker can store this information. The credentials are compromised to the attacker.

SQL injection

SQL injections (SQLi) target a web-site, where data entered by a user is not filtered properly. A web site often has a certain interaction with its users. Usually there is a database used to store some of the data the user is presented. Some of these sites require or at least offer a user a login, to either allow access only to restricted information, or to provide a better usability. In all these cases the credentials of the users are stored in a database.

An attacker tries to find flaws inside the web-site, to control the SQL command the site sends to its database. If the attacker succeeds he can launch nearly any SQL command. One of these commands could be an SQL statement like:

```
SELECT username, password from users
```

If the table users exists the attacker would get all user and password combinations stored in the database. By this the credentials of all users would be compromised to an attacker.

The difficulty in this attack is to know the exact name of the table and the names of the columns where the data is stored in. To get these information the attackers uses SQL commands to acquire this information from the database system. Mysql on of the most famous database in the online world, has special commands to retrieve this data from the system.

To make such attacks easier, special tools like 'sqlmap' exists, where the attacker only needs to provide the URL of the page where the SQLi would be possible. The script extracts all tables from the database and can even be limited to extract only certain data like the database schema or an specific table.

Guessing

Guessing attacks in the online world target a web site or an Internet service which needs certain credentials to login. The first step in a guessing attack is to get a list of valid user names. If this list is not available, the user names have to be guessed too. An attacker could try to perform a google search for the site he wants to attack. With some probability some of the users will be mentioned on some websites. Often users are making forum posts with their email addresses, specially to resolve a problem with a software. Another source of email addresses can be social networks like Facebook, Linkedin or Xing. User provide their email address to other to be reached. Often these email addresses or at least the first part of it is used as a login user name.

The next step for an attacker is to guess the password. The attacker takes a word, where he believes that the word is most likely used as a password by this user. He sends this password as soon as the target site asks for the password and checks if the result is either success or failure. In case of failure the attacker chooses the next most likely word and retries the attack. This is repeated until either no word is left in the dictionary or until the password is found.

Password	Number of occurrences	Percentage
123456	290729	0,89
12345	79076	0,24
123456789	76789	0,24
password	59462	0,18
iloveyou	49952	0,15
princess	33291	0,10
1234567	21725	0,07
rockyou	20901	0,06
12345678	20553	0,06
abc123	16648	0,05

Figure 2.1: The most frequent passwords of the RockYou leak

The passwords used in this guessing attack come from either special knowledge about the victim or from special prepared dictionaries. This knowledge includes if the victim has a crush on another person, the name of the pets he has or already leaked passwords of a person. All this information can be part of a password. Especially leaked passwords become a problem for people who reuse passwords on different sites.

On the other hand special dictionaries are available on the Internet. They are taken from statistical informations of already leaked password. An in deep analysis of the RockYou list showed that some passwords are more likely than other (compare Figure 2.1). Taking this knowledge into consideration the attacker will try first the passwords, which are most likely to be used.

These kinds of attacks are very slow. The attacker needs to open a connection to the target site, wait for the response, send the credentials and wait for the result. Depending on the speed of the network connection, it can take several seconds to perform one single request.

The next problem for the attacker is, that these kinds of attacks can be easily detected by the target site. The targeted site can count each faulty login try. As soon as a threshold is reached, the target site can prohibit further tries for an amount of time or even suspend the account targeted. While the attacker could use a bot net, the target can enhance his protection mechanisms to allow only a small amount of false login attempts from an IP address per hour.

Another protection mechanism for a site is to check the geographical region where an IP address belongs to. While most users usually login only from one country, the target site can validate that the login attempts are from this region. In case they don't the target site could request additional information which are most likely only known to the original user.

Malware

Malware is another attack vector. An attacker tries to convince a target to install malware on his computer. The malware hides in the system and has full control of the attacked computer. The malware can hook itself to any program, every data the users enters from this moment can be captured, stored and later send to the attacker. Common types of malware are trojan horses or viruses. Specially trojan horses are used by an attacker to gain access to online banking and therefore harming a victim on a financial basis.

2.3.2 Social engineering

A social engineering attack tries to convince a user to reveal his credentials to an attacker. Social engineering attacks are somehow between online and offline attacks. An attacker contacts the victim either by mail, telephone or visits him. He then comes up with a story that for some reasons he needs the credentials of the victim to perform a certain action. Kevin Mittnick describes in his book "The art of deception" [KDM03] a lot of ways how to get information from a user by social engineering, ranging from 'just ask for it' to coming up with a story to convince him, that it is to the best of the user to provide the data.

An example for such an story would be a phone call where the attacker tells the victim that he is from tech support and needs the password because he did a mistake and the victim would no longer be able to work, if he would not reset it to the original value.

If the attacker is skilled he will be able to get the data he wants. This attack is not only limited to user password combinations, but is also used as a preparation of a guessing attack. Kevin Mittnick mentioned a case were the attacker tried to get a printed phone register. This was later used for creating the background of further attacks.

2.3.3 Offline

Offline attacks target always passwords which have been leaked during an online attack. The encryption schema of leaked passwords is in all cases a cryptographic one way function. There are basically two kinds of these functions: simple hash functions and salted hash function. By salt we understand additional data which is added to a password in order to make sure that passwords, which are equal, do have different output from the one way function. This can be either some random data, which is stored together with the result of the one way function or generated data from the user name. This data is added either in front or at the end of password, right before it is given to the one way function. Ideally this information differs from user to user.

Due to the way in which one way function work, an attacker has no chance to calculate the password from the results of the one way function. His only hope is to guess the password. Therefore we divide this attack into some subgroups:

Brute force

Brute force describes an attack where the attacker tries each and every possible password a user can enter. On an American keyboard a user can enter 96 characters, which are build out of 26 lower case, 26 upper case letters, 10 numbers, and 34 special characters. This gives 96 possible one character passwords, 96×96 possible two character passwords and so on. This brings us to the formula:

$$96^{passwordlength}$$

Brute force attacks get infeasable already for moderate password length.

Dictionary attack

Using word lists to attack passwords have a much smaller complexity than using brute force and therefore are a lot faster. This advantage is however bought with the number of passwords which could be found. Words which are not in the word list can not be found, since they are not tested by the attacking algorithm. On the other side this form of attack can test longer words without getting problems with time. Often word list attacks are combined with brute force attacks. Depending on the target algorithm and the time an attacker wants to spent he would try to brute force passwords up to the length of 6 to 8 characters and would try to find the rest with an word list attack.

Markov chain

Markov chains are between brute force and word list attacks. They are based on Markov models named after Andrey Markov. Markov presented his idea first in [Mar06] which has been reprinted in [Mar71]. The basic idea is that starting with a character the following characters do not all have the same probability. As an example starting a word with a 'W' there is a much bigger probability that an 'h' is following than that there comes a 'z'. By calculating and evaluating this probability a Markov chain can create words which have a high probability to belong to the target language. The probability increases even more if not only one but two characters before are taken into consideration.

Rainbowtables

Rainbowtables are brute force attacks where the attacker uses the memory to speed up the brute force. An attacker pre-calculates the results of a hash algorithm. To avoid storing all the password/hash combinations the attacker uses a reduction algorithm to extract the next password and hashes this again. This loop is performed for a fixed number of cycles. After these cycles the start word and the last hash are stored in a database.

As soon as an attacker wants to crack a special hash he checks if the hash is already stored in the database. If this is not the case he uses the extract function to get a next password from the hash. Afterward he check if the new hash is in the database. These

steps are performed until either an hash is found or the predefined number of iterations are performed.

When a hash has been found, the attacker starts with the start word of the cycle where the hash was found, and performs all steps of the cycle till he finds the hash value he is looking for. The last result of the extraction function before the hash is found is the password the attacker was looking for.

2.4 Tools

On the Internet there are several tools specialized on password cracking. We are describing here the two most common tools. More tools can be found at [Sec], most of them are specialized on a certain target.

2.4.1 John the Ripper

John the Ripper (JtR) is one if not the most popular password cracker for Unix and hashes. JtR first published during the 1990's. At this time it was intended to crack passwords on Unix systems. Over the time more and more functionality was added bringing it to a stage where it is able to crack more than 40 different types of passwords systems. JtR comes with the ability to use dictionaries, brute force and Markov chains as attack vectors. It supports configurable rules to adapt the attack against special company rules of the target. In the community addition it offers support for GPU (currently very limited) and to use multiprocessor and even multiple hosts via an interface to the mips system, a system for parallel computing.

JtR is specially famous for its high performance on cracking. Performance measures on a HP nx9720 showed that it was able to test more than 1.8 million passwords per second against a single MD5 hash value.

2.4.2 Cain & Abel

Cain and able is a software to specially attack the windows password system. It is able to sniff network traffic and extract the credential data, like the user name and the encrypted password. It can perform various attacks on the encrypted passwords, like brute force, dictionary attacks and can even use rainbowtables to extract the password.

2.4.3 Elcomsoft

Beside the just described tools, one company is worth to be mentioned when it comes to password recovery. Elcomsoft [Elc] is a company founded in 1990. It is located in Moscow, Russia. Over the years they created a set of tools to crack passwords nearly all popular applications. Their tools are able to work in a cluster environment and even support the use of graphic cards to crack passwords.

2.5 Attackers motivation

The motivation of an attacker can be seen in three different ways.

2.5.1 The good

At first it seems that the word attacker can be hardly seen in a good way. An attacker is usually at first someone starting a threat. But even that threats have a bad taste, there are attackers which act on legitimate basis. One of theses attackers are pennetration testers. A penetration tester is a person or a group of persons who are hired by a victim to test its security. As part of these security tests the penetration tester sometimes approaches passwords in encrypted and unencrypted form. To test the strength of a password, he has only a limited time and therefore is in the need to decrypt the password fast.

2.5.2 The bad

The bad site of the attackers motivation are criminal based. The goal is to gain profit out of the data stolen. The attacker tries to impersonate a victim to make further attacks on others more probable. Having the credentials for an email account, the attacker can analyse the social background of people associated with the victim. He can reply on emails and create by this a better background for social engineering attacks.

2.5.3 The ugly

The ugly site of the motivation, or better to say the grey site is an attacker who's goal is not to harm people but, who is willing to accept a certain risk for their victims. One example for this are hacker groups which are leaking passwords just for fame, or providing SQLi to the public without respecting the ethics on "full disclosure". But even if an attacker tries to inform a company about a problem on their site, some companies react in strange ways, by sending lawyers instead of fixing their problem. By thinking that a lawyer will prevent a bad reputation, they ignore that the problem still exists and needs to be fixed.

3 Composite passwords

There are a lot of ways to crack a password. They range from brute force, rainbowtables, to dictionary based attacks. Especially in the first two kinds, the attack vectors can only crack passwords up to a certain length. This limitation is usually somewhere between 8 to 10 characters. Matt Weir was one of the first who improved attacks based on dictionary with patterns [WAMG09]. Therefore he took a dictionary and a sample of the passwords he wanted to attack and generated patterns out of it. He used these pattern to modify the dictionary to get better results. The improvement was impressive.

The disadvantage of this approach nevertheless was, that he could find only passwords, where the word part had to be in his dictionary. If he tried to guess a combined password, like 'ILoveJesus' this word had to be in his dictionary.

3.1 Basic idea and overview

Passwords like "ILoveYou" or "CorrectHorseBatteryStapple" [xkc] are not covered by dictionaries at all. These passwords are a combination of words. Some of them make sense in the way that they are sentences ("ILoveYou"), some of them make only sense to the user using them. Only a few of them are found in special dictionaries for cracking, but only, if they already occurred somewhere in a password list. The phrase "iloveyou" is an perfect example for this. This password is part of many guessing dictionaries. But what about the other phrases? They are usually not covered. Therefore IT security specialists started to recommend using sentences as a password. The clear advantage is that sentences are usually long. Which make them un-crackable in brute forcing.

Our work improves Weir's approach. If we look at sentences they are at first a combination of words. While all combinations of words lead to an extreme big dictionary, we need to find a balance between the size of the dictionary and the number of words we are going to crack. Therefor we will start by learning how word are combined in the sample. The assumption is, that users are choosing the passwords in a similar way.

For the analysis the RockYou list described in Section 4.1.1 will be used. A first analysis showed that not only sentences were used as a password, but also combinations of the same word. To cover all cases of combinations we are analyzing them without taking linguistic grammar into consideration. This will result in sentences which do not make sense for a reader. On the other hand we can be sure that we will cover all kinds of combinations.

We create an intermediate dictionary by enhancing a starting dictionary. To achive this we learn patterns how passwords are combined. With these patterns we are combining words from a starting dictionary and add them to it. We call this enhanced dictionary

an intermediate dictionary. To keep the balance between the number of words inside the dictionary and the words, which can be cracked with them, they are set into relation to the words this pattern would have been able to crack.

The intermediate dictionary will later be used in a modification algorithm. This modification algorithm is an implementation of the algorithm Matt Weir described in his paper [WAMG09]. The algorithm takes its own patterns and modifies the words from the dictionary to

- change cases,
- add numbers, and
- add special characters.

The resulting dictionary is then used to "crack" the test set. While we are using only clear text passwords, the cracking is a matching of words. An attacker who is going to hashed passwords will use tools like JtR instead of the cracker used in this work. The cracker in this work is mainly enhanced to create measurement data to and not to do a cracking in the sense of cracking hash values.

3.2 Dictionaries

A dictionary is a list of words. During a guessing attack these words are tested against passwords.

3.2.1 Available dictionaries

One of the key components of the approach is an existing dictionary. To get a good starting point I looked for already existing dictionaries on the Internet. To evaluate if a dictionary is suitable, we have chosen some basic criteria:

- The dictionary should not contain word which are not in the target language.
- The dictionary should not contain composite words which are not in the target language.
- The dictionary should contain names.
- The dictionary should contain company names.
- The dictionary should contain names from novels.

Keeping these criteria in mind we looked for dictionaries available on the Internet.

Ispell

Ispell is an open source software which implements a spell checking mechanism. Together with the software a dictionary is delivered. To implement spell checking on a computer valid words have to be identified. Therefore a dictonary is included together with a set of rules how to change this word according to the grammar.

While there are a lot of spell checking dictionaries, the Ispell dictionary was the one with the lowest amount of formatting rules inside, which made the work a lot easier. Checking the criteria I found that the Ispell dictionary is mostly clean. One drawback was that it contains the alphabet itself. To avoid problems with this, we reduced the list of one letter words to "a" and "i". The last three criteria are not matched.

Wikipedia

Wikipedia is the most famous encyclopedia on the Internet. It was founded on January the 15th 2001 by Jimmy Wales and Larry Singer. As of January 2012 it has 3.8 million articles in the English language and is available in 282 languages.

The idea behind Wikipedia is, that every one can participate and write, modify and enhance articles. A core team of volunteer administrators have the power of editorial control. The content of Wikipedia is available under the Creative Commons Attribution-Share-Alike License.

We extracted the dictionary from Wikipedia as follows: We found a database on words used in Wikipedia at University of Leipzig [Uni]. After a short analysis of the database, we encountered that additional information was needed to use the word in this project. We downloaded the English part of the Wikipedia via the official link [Wikb] and started our own analysis.

We found that there are a lot of words inside the dump which did not make sense at all. So we were wondering why a word like 'AaaaaAAaaaAAAaaAAAAaAAAAA' was inside. It belonged to a video game [Wika] for Microsoft Windows. As a result the decision was taken to create a new extraction of the Wikipedia.

Gutenberg

The project Gutenberg is a collection of public domain text. Due to copyright regulations any text who's author is more that 70 year dead is automatically going to public domain. For translated texts the translator has also to be dead for at least 70 years.

The project was started by Michael Hart in July 1971 to make library books, belonging to public domain, available in an electronic version. Since 1971 more than 38000 ebooks have been published. New books are added by volunteer who scan the books manually. After an automatic optical character recognition (OCR) the scans are provided via a distributed proofreading process, where volunteers correct the scanning errors.

An additional source for ebooks are authors who make their books available under a special license to be published on the projects pages. All books at Project Gutenberg can be downloaded and read for free.

The text files from Project Gutenberg matches mainly the names and names from novels criteria. Due to the writing style of some authors the first two criteria are not matched.

Special purpose dictionaries

The Internet contains a lot of special purpose dictionaries, from which some are special made for password cracking, mostly containing already leaked passwords. While they are actually very useful for password cracking in general, they do not fit to our purpose. Checking these dictionaries, we can find that they contain names but contain also a lot of combined words. Unfortunately they are too small for our purposes as well.

A good source of dictionaries is Darknet [Dar]. They also link to the University of Oxford, where a lot of dictionaries are available on an FTP server [Oxf].

3.2.2 Building the dictionary

I downloaded the Wikipedia database dump from the download URL[Wikb] provided by the project. A tool called wikiprep extracted the articles from the XML file to single files. A set of sed script helped to remove the XML and special Wikipedia formatting. At the end there were only the words left. These word were sorted and counted. Words which had a count of less then 200 were dropped to remove rare words and words which were not wanted like 'AaaaaAAaaaAAAaaAAAAaAAAAA'.

Number dictionary

The usage of a number dictionary was in the beginning not obvious. The first runs of the attacks showed nevertheless that the dictionary used to attack the RockYou list was growing extremely fast. The first investigation showed that some parts of the dictionary had been extended with numbers of the size eight and more digits. To get a feeling how many numbers where really used we extracted the numbers from the RockYou list and created Table 3.1.

3.2.3 Results on building the dictionary

As a result of the investigations on the number dictionary, we took the decision to brute force numbers with 6 digits and less. A coverage of nearly 45% justifies the brute force. For numbers with less digits we created a script to extract the numbers from a sample. These extracted numbers can after wards be used as the number dictionary, assuming that there will be a certain probability that the numbers do not show up only once. Using this algorithm the number of passwords is drastically reduced when they contain numbers with more than 6 digits.

Number of digits	Number of different numbers	percentage covered
1	10	100%
2	100	100%
3	1000	100%
4	10000	100%
5	76017	76.0170%
6	448223	44.8223%
7	595503	5.9550%
8	486267	0.4863%
9	338707	0.0339%
10	504644	0.0050%
11	115121	0.0001%
12	41513	<< 0.0001%

Figure 3.1: Numbers used in RockYou list

3.3 Implementation

The implementation of the system was divided into several steps, to make the setup modular. By this we are able to analyze the input data once and compare the result of the numbers with and without our modifications. The first step is always the 'Dataset analyzer', the step of 'creating the intermediate dictionary' is optional. The intermediate dictionary will contain a modified starting dictionary. The last step is the dictionary modifier. This step implements the algorithm of Matt Weir.

To make the further implementation easier the dictionaries have to be partitioned by the size of the words. The dictionary is provided by a directory. Inside there is a file for each word size containing the words in alphabetical order. Some small Unix scripts take care of the sorting and the creation of the dictionary.

3.3.1 Dataset analyzer

The dataset analyzer takes as an input a dictionary and a sample of the passwords which are to be cracked. One of its main goals is to analyze the efficiency of the dictionary. To achive the goal, the first step is to load the dictionary into the memory.

The words from the sample are taken and analyzed if they contain one or more words of the dictionary. To achive this a word from the sample is read. In this word each sequence of letters is isolated. Inside the sequences the algorithms identifies single words from the dictionary. Starting with the longest maximum length the algorithm look if this sequence exists as a word. If the word could not be found the sequence is divided into two parts. The position where to part the sequence is moved from the longest to the shortest position. Both parts are treated as separate sequences and therefore analyzed separately. As an example the algorithm tries to recognize the sequence 'ILoveJesus'. To make it better readable the words in this example have capital letters.

It starts with the complete sequence 'ILoveJesus'. If this is not found the next sequences

are 'ILoveJesu' and 's'. If the first part is not identified as a word the algorithm continues with 'ILoveJes' and 'us' and so on till 'I' and 'LoveJesus'. 'I' is a regular word in the English language. The algorithm continues with the identification of 'LoveJesus' where it will identify 'Love' and 'Jesus'.

If the sequence has been completely identified the pattern found is stored inside an internal structure. In the above example the identified structure of the sequence is '1‖4‖5' as 'I' has one letter, 'Love' has 4 and 'Jesus' has 5 letters. Together with the pattern a counter is stored. If the pattern has already occurred it is incremented by one.

To enhance the processing of the just collected data, the percentage of the found patterns compared to the wordbook size is calculated. Since the wordbook is already in the memory, the size is of the wordbooks for each word length is a side product. This word count is set into relation of the pattern, showing how efficient a pattern would be. This means that for our example we set the number of words the pattern reflects '1‖4‖5' in relation to the number of words we need to create to get all words of the pattern. In this case it is the number of one letter words multiplied with the number of four letter words multiplied with the number of five letter words.

Beside analyzing and identifying words the second goal of the algorithm is to create a pattern of the samples. This pattern will be used later in the DictionaryModification algorithm. Each line of the sample is passed into a Pattern class. This class identifies four kinds of characters:

- small letters (c),
- capital letters (C),
- numbers (n), and
- special characters (s).

Similar to the word patterns the character patterns are stored in a list where the patterns are aggregated.

At the end the WordBook Evaluation algorithms stored three files to the hard-disk:

- DictEval – the list of word pattern found together with the number of words it contains,
- DictEvalFlt – the list of word pattern found in relation to the words needed to build them, and
- PatternList – the list of character patterns found

3.3.2 Creating the intermediate dictionary

The intermediate dictionary is a dictionary which is enhanced with combined words. The algorithm takes the results of the Wordbook Analyzer and creates a new dictionary by combining the words according to the patterns stored in the 'DictEval' and 'DictEvalFlt' file. For the further analysis and evaluation two algorithm where created.

The first algorithm creates a intermediate dictionary which reflects the data from the DictEval set. It sorts the pattern by the number of words which can be build from the pattern.

The second algorithm creates a intermediate dictionary with respect to the calculated percentage from the DictEvalFlt file.

3.3.3 Dictionary modifier

The Dictionary Modifier modules is an implementation of the algorithm of Matt Weir. Matt Weir describes in his paper [WAMG09] a way to manipulate a given dictionary to increase the number of hits a given dictionary gives, under the assumption that we can learn a pattern set from a sample.

The generation of the sample is a side product from the Dataset Analyzer. For the evaluation two versions of the algorithms were used.

Both versions take a directory as input. The directory contains the dictionary divided into files according to the password size. The difference comes with reading the Patternlist file. The basic algorithm takes the list of pattern together with the number of occurrences and stores it into the memory. The enhanced version instead sets the pattern in relation to the number of passwords which are to be generated by the pattern. To achive this the enhanced algorithm reads an index files, containing the number of passwords for a given length and calculates the expected size of the list this pattern is generated. The number of possible hits is divided by this size and stored together with the pattern in a list. Both versions sort the pattern list by the additional information stored together with the pattern. The next steps are valid for both versions.

The algorithm takes the pattern from the sorted list up to a given number of steps. This number is either provided by the user as a parameter, or the default value of 40 is taken. For each pattern, the pattern is divided into its parts, where a new part starts when the letter in the pattern changes. In this step “c” and “C” are treated as the same letter. Let n_i be the number of letters in part i. Each part will be replaced by a set of possible representatives with the size n_i for part i. The algorithm generates the cross product of all these sets.

In case of letters c or C the set is taken from the dictionary containing the words with size n_i. For special characters (s) all possible combinations from the alphabet "§$%&()=?\¨-_.:,;" are taken. In case of numbers (n) for the size up to size 8 the numbers are enumerated. For numbers with a size n_i bigger than eight a special number dictionary has to be provided.

During the analysis of the RockYou database it showed up that numbers up to the size of 4 digits were 100% included in the RockYou set. Further investigations showed that numbers with more digits were represented with a lower coverage, see Figure 3.2. To reduce the size of the resulting modified dictionary I did draw the line at eight characters. To control this behavior the user can provide a number dictionary even for less then 9 digits. If such a dictionary is present the dictionary has a higher priority than the brute forcing of the numbers.

number of digits	percentage
4	100%
5	76%
6	44%
7	6%
8	0.5%
9	0.03%
10	0.05%

Figure 3.2: Number of digits vs. found

3.3.4 DummyCrack

The DummyCrack module is responsible for evaluating a dictionary. The goal is to count the numbers of hits a dictionary can achive against a clear text password list. A lot of cracking tools are able to calculate how many passwords can be found against a certain encryption or hash mechanism. In our case we work on clear text passwords and do not try to crack MD5s or other hash values. While on encrypted passwords it is important to measure the number of password tries our goal is to measure the number of hits after a certain amount of tries.

The algorithm behind is divided into two steps:

1. Read the password file, and
2. count the hits.

In the first step the password file is read into the class EvalCounter. EvalCounter is an enhanced hash table. It includes a function “increase”, which is responsible for counting the occurrence of a password. The increase function checks in a first step, if a given password is already present in the hash table, if not, the password is added with occurrence one.

For counting the hits, the DummyCrack module expects to get the guessed passwords via standard in (STDIN). This decision was made to reduce disk space usage. The module takes a password from STDIN and checks if the EvalCounter contains this password. In this case a counter for the hits is increased by the number of occurrences in the password file. To avoid duplicated counting, the password is removed from the hash table.

For evaluating the values an intermediate output is generated as soon as a certain threshold of tries is reached. The threshold starts with one and is increased by the factor of ten. As soon as the end of STDIN is reached DummyCrack stops and prints an aggregation of the results.

3.3.5 GenSample

The last module to describe is the algorithm to generate samples from a target set. The idea is to load the data into memory and divide the set of data into two parts, a sample

set and a test set. To generate sample the program gets two parameters. The name of the file from where the samples have to be generated and a factor x. The number of lines are in the dataset (l) is counted and divided by $x + 1$. Bringing it to:

$$n = \frac{l}{x+1}$$

By this the input parameter is is treated as a factor $1 : x$.

The algorithm chooses n different random numbers between 0 and $l - 1$. These numbers define the lines which are to be chosen by the algorithm as samples. At the end the algorithm writes two files. One containing the samples (marked lines), the other containing the not selected lines, the test set.

3.4 Cracking by search engine

Not directly related to composite password we looked at a different way how to crack a hash value other by using a dictionary attack. On the Internet are different websites which uses rainbowtables and databases to get the original word from a hash value. To get a feeling how fast a hash value can be cracked, we implemented an algorithm which makes request to different services to request the original word of a MD5 hash value. To avoid asking the same value more than one time, the algorithm stores the result of every request into a database. If a hash value could not be resolved the value is stored in a special table, marking this value as not found in the services.

As an input we used leaked passwords from the Internet. A lot of these data is stored as hash values. Making it hard to being decrypted by an attacker. By this approach we were able to crack 2/3 of the passwords.

3.5 To 1337 or not to 1337

Leet-speak (1337-speak) was first used during the bulletin board age in the 1980's. During that time the Internet was still in his childhood. Nearly no private user could afford a direct connection to the Internet. Instead it was very popular to use bulletin boards systems (BBS). User connected to a BBS through a modem and a dial up line presenting a login screen. As soon as he has passed this screen he could access the bulletin board, which was mostly what we know as a forum in our days. A lot of these BBS were connected to other BBSes, they exchanged to content of their boards on a daily sometimes even an hourly basis. Additionally to the forum some board allowed the user to chat to other users currently logged in. Sometimes in these groups there were talks about unwanted topics going on, like hacking for example. To get rid of such talk the operators used black lists of unwanted topics, an operator would have to approve certain message containing these black listed words. Very soon the users found their way around this by making intentional spelling mistakes or manipulating the words. They exchanged letter by numbers and signs which looked similar to the letters they would

use in this position. As an example the letter 'a' was replaced by '4' which looks similar to an capital 'A'.

In our days this exchange is still popular in the group of hackers which, especially when it goes to user-names. On big forum site or on free-mail servers short or popular user-names are often gone very fast. The user-name 'Norbert' is a good example for this. It is a common first name in Germany. To write the name in leet-speak there are a lot of possibilities like:

- n0rbert,
- norb3rt, or
- nor8ert

Just to name some of the possibilities. While the human brain can easily translate these letters back to the original meaning it is a lot harder for a computer. By looking at the following (incomplete) list of translations we see that some letters like 'r' and 'z' translate to the same leet letter. This makes it hard to identify the real word which was intentionally behind the phrase. In contrary to the computer the human mind forgives spelling mistakes in a much more easy way, therefore the human being 'sees' what the right word would be.

This makes it especially hard to analyze passwords. First and foremost if the special signs were not intended to be used as leet-speak. During the analysis of the RockYou list we found passwords like:

- danny!
- f3l!X1509J
- peps!!

While in the first example the '!' is used only as an exclamation mark, the second uses it as a replacement for 'i' and in the third we can only assume that the first '!' was meant to be an 'i' and the second was meant as exclamation mark. With respect to this example we came up with the fact that we had to test at least 2 possible replacements for each possible leet character:

- The meaning of the letter being replaced through the leet speak filter, and
- the original meaning of the letter.

As long as there are only a limited number of such signs in a string it is quite easy to brute force all possible exchanges to see if a valid word would be the outcome. But as soon as we have a longer word where there is a double meaning for nearly every letter, the number of possible words goes up exponentially. While we as humans would assume that 'pepsi!' would have been the right candidate, the computer does not have such a language feeling and would have to look for all 4 possibilities. Depending on the dictionary the computer could identify the following possible matches:

- peps!! - four letter word plus two special signs[1]
- pepsi! - five letter word plus one special sign
- peps!i - four letter word followed by a special sign and a one letter word
- pepsii - five letter word followed by a one letter word

As soon as this happens the analysis of the passwords simply take to much time and has to be stopped. During the experiments we did choose a maximum of 1000 possible outcomes. As soon as the limit was reached we skipped the word for analysis in regards to leet-speak.

[1] Peps is a financial term for a pension contribution plan

4 Experiments

As with every research the thesis has to be validated. In this chapter we will start with a close look on the data which will be used in the experiments.

4.1 Datasets

For evaluating the algorithm it would be nice to have real live data. The issue with real live data is, that such data could contain user data which have to be protected. Even when such data has already been leaked to the Internet there are certain laws which prohibit to store such data. To circumvent this issue the datasets used in this work are from a source which has anonymized the data. All datasets have been downloaded from Skull Security [Sku]. They contain only the passwords which have been leaked, no login names and no email addresses. By using this source I strongly believe that this research does no harm to the user which have been affected.

4.1.1 RockYou

The RockYou dataset is a set of leaked passwords from the company RockYou, which is a developing company for games focused on social networks, founded in 2005. In 2007 they started to integrate their games into Facebook. In December 2009i, attacker succeeded to use SQL injections to extract user data from the internal database.

The login mechanism of RockYou had several flaws:

- the minimum size of a password was 5 characters only,
- no special sign were allowed,
- passwords had been stored in clear text, and
- the password was mailed to the user.

While these passwords are not very interesting for a password research, the bad thing got even worser. RockYou had offered their users to post the games and results to the social networks. For this purpose they told, that they need the credentials of the systems, mentioning that “Login info will never be saved”. The leak proved a different truth. The database contained tables with partner data including data to social network sites like Myspace, Facebook, and others. Later the data was published on various sources. In anonymized form the passwords are still available at Skull Security [Sku].

After the leak it took several days until the users were informed. Most users got the information about the leaks from news sites on the Internet. At the end of December 2009 a class-action lawsuit was opened against RockYou.

month	number of emails
2011-10[1]	43104
2011-11	78902
2011-12	119245
2012-01	170828

Figure 4.1: Number of passwords leaked

4.1.2 Myspace

The passwords on the Myspace list are most likely leaked through a phishing site. In 2006, hyperlinks had been send through AOL instant messages leeding to a bogus website, spoofing the myspace.com login page. As soon as the user entered their credentials, the site set a cookie to prevent another visit to the user. The user was forwarded afterwards to the real myspace.com website.

At the end, 37144 passwords had been leaked to the public. It is unknown which passwords really were in use on Myspace. Passwords like "fuck you, stupid, trying to get peoples passwords! we're not that stupid!!!!!" clearly show that at least some of the victims encountered that they dealt with a bogus site.

4.1.3 Others

Beside these datasets there are a lot of other passwords leaked. To get a feeling about the amount of passwords being leaked on the Internet, I monitored the Pastebin site [Pasb] for several months, using a script from Xavier Garcia, which had been published via his blog [Xav]. We modified the original algorithm to enable a small rating system together with a threshold. By using these modifications the tool stored files, from on Pastebin to the local harddisk. The main criteria to download a file was that a email address has to be in the file, indicated by a regular expression for email addresses. Afterwards it was counted how many email addresses from yahoo, google and Hotmail were included. If a minimum of 100 addresses were included the file was stored and later checked manually.

The script monitored Pastebin for a period of 3 and a half month. After, manually removing false positives, the email addresses have been extracted and sorted. Duplicates have been removed and the addresses have been counted. As soon as the numbers were collected, the user data was deleted. The investigation started on October 10th 2011 and ended on January 22nd 2012. Not all leaks contain email addresses, by knowing this the following numbers are the minimum of passwords leaked each month.

4.2 Running the attack

Due to the amount of data in the RockYou dataset, it turned out that the Java parameter "-Xmx3000m" had to be used, increasing the memory reserved for the application to 3

[1] We started in the middle of this month

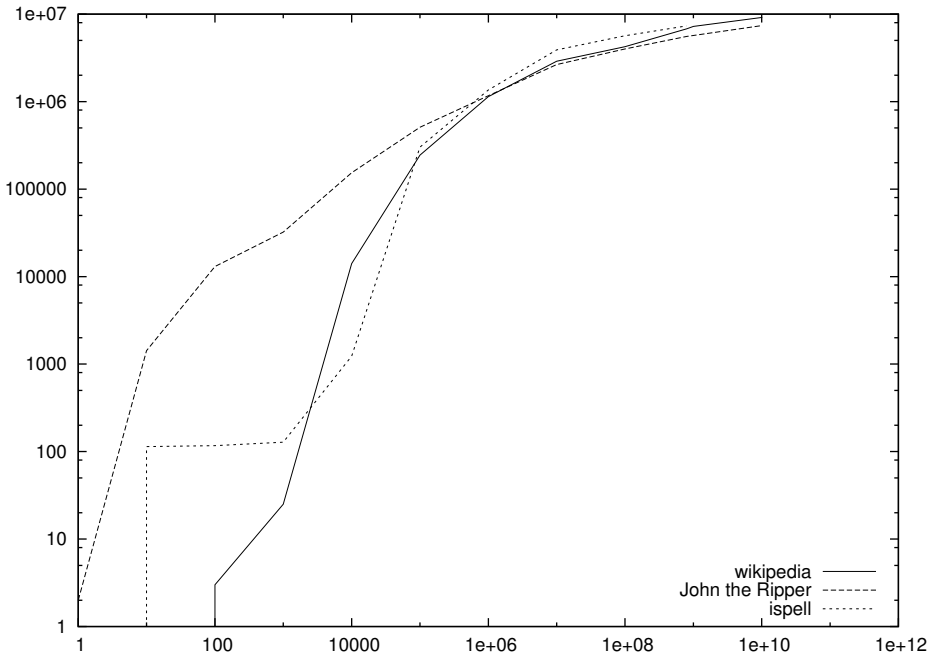

Figure 4.2: The 1:1 distribution logarithmic scale

giga byte.

To gather the results we divided the RockYou list into a sample and a testset. With the Gensample algorithm we divided the list into two parts with the quotients 1:1 and 1:100 and tested the algorithm using the ispell and the Wikipedia basic dictionary. To measure the quality of the results we included the results from the JtR incremental attack, the ispell and the Wikipedia dictionary.

4.2.1 The 1:1 distribution

The first set we analyzed for the experiments was the one to one distribution. The Gensample algorithm divided the RockYou list in 16.301.694 sample passwords and the same amount of passwords in the testset. The wordbook evaluating algorithm analyzed the samples and created the basis for the further work

Figure 4.2 shows the number of passwords we were able to crack with the JtR incremental list, the Wikipedia dictionary, and the ispell dictionary. The Ispell and Wikipedia dictionary were used after a full run, inclusive the creation of the intermediate dictionary and the Weir algorithm. Figure 4.2 shows that, in the beginning, the JtR incremental list gives much faster results. While we got only after 100 password tries the first ≈ 100 hits with the ispell version, the Wikipedia brought just 3 hits. Due to the much bigger Wikipedia dictionary it takes much longer to get a good number of hits.

Figure 4.3 shows the same dictionaries as in Figure 4.2 but this time the number of cracked passwords is shown on a non logarithmicscale. It shows that the ispell dictionary

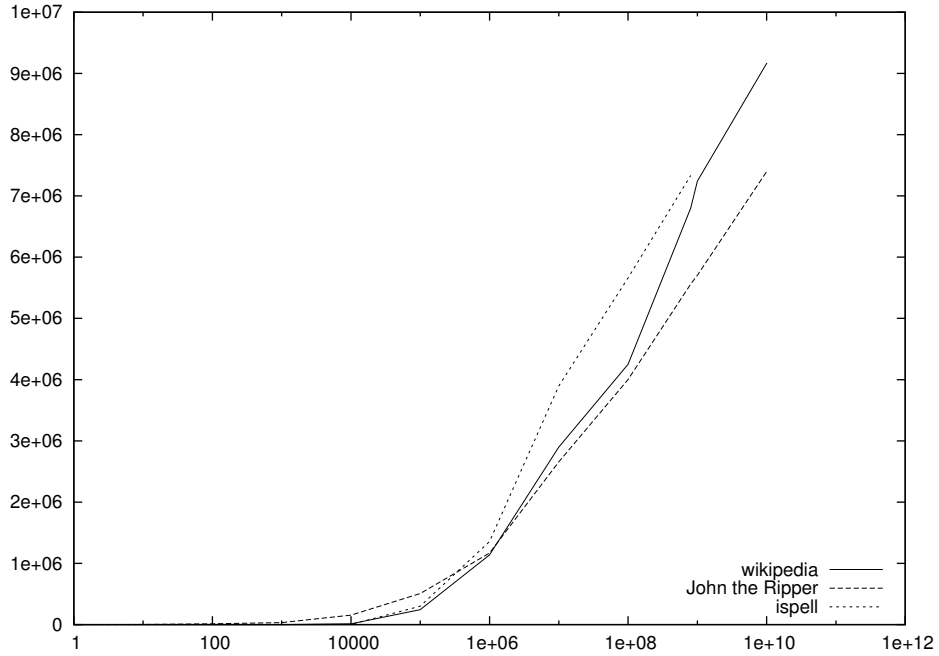

Figure 4.3: The 1:1 distribution non logarithmic scale

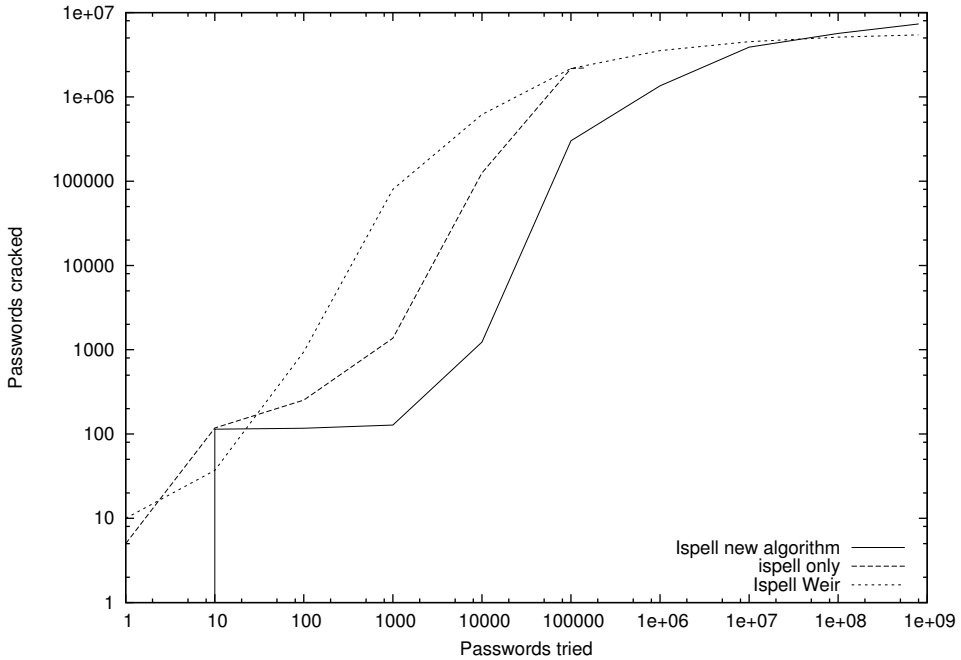

Figure 4.4: The 1:1 distribution Ispell

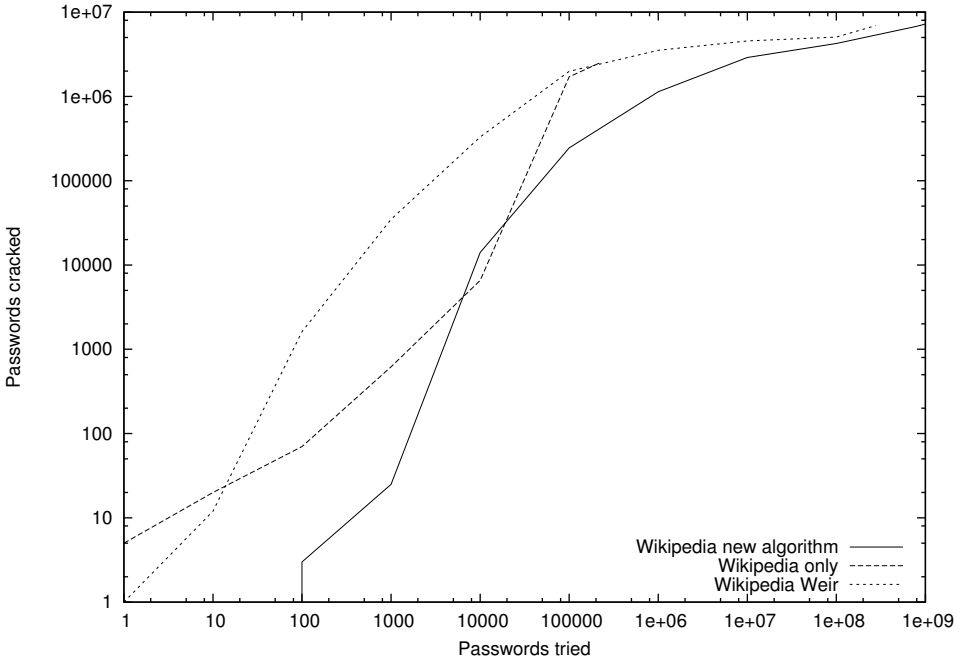

Figure 4.5: The 1:1 distribution Wikipedia

brings nearly the same amount of password hits as the JtR incremental. But the ispell dictionary creates these hits with roughly 100 times smaller amount of tries.

Figure 4.4 compares the amount of passwords we are able to crack with three different approaches. We used the ispell dictionary only, the ispell dictionary with enhanced by Weir and the ispell dictionary with a full run. Our new approach has in the beginning a worse performance than the original dictionary and even the dictionary enhanced by Weir.

For a more detailed approach we are taking a look at the numbers. All together we tried to crack 16.301.694 passwords. With the dictionary only attack we tested 142.238 passwords and found 2.204.812 of these (13.5%). For the Weir approach we tested 795.525.940 passwords and found 5.436.255 (33.3%). Using our approach we got 7.333.935 after 801.280.616 tested passwords which covers 44.9% of all passwords. This is an enhancement of 34% compared to the approach of Weir.

The drawback of our approach is that it takes some time to be effective. This could be reduced by taking the probability of words into consideration. This drawback becomes a problem only in case that it takes a long time to run a single test. At the moment the most time consuming operation is the step to apply the patterns of the Weir algorithm to the intermediate dictionary. The original 142.238 passwords from the ispell dictionary have been increased to 238.117.401 words in the intermediate dictionary. The Weir algorithm filters these words based on the numbers taken from the analyzing algorithm and enhanced these with the patterns.

The biggest increase of the words could be found on 9 character words. The original

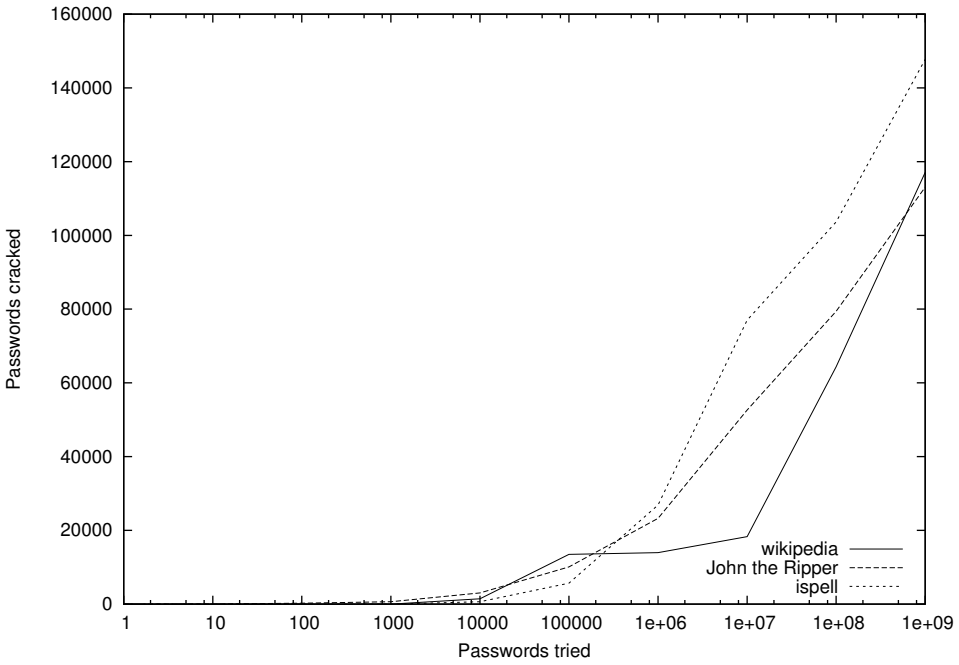

Figure 4.6: The 1:100 distribution non logarithmic scale

dictionary contained 44.225 words of 9 character. The intermediate dictionary on 9 character words raised this number to 133.418.581. Taking the probability of the patterns with 9 characters into consideration these patterns could not get effective on the attack.

Contrary to the results with the ispell dictionary, the Wikipedia version did not reproduce these results. The reason for this is that the original Wikipedia dictionary had 218.036 words which is basically 53% more. For the Wikipedia dictionary we had to limit the number of patterns used to get the numbers in a reasonable time. As a result pattern which would come effective and bring the advantage of composite passwords had to be left out.

4.2.2 The 1:100 distribution

The second analysis was taken on an one to hundred distribution. The sample set has 32.280.583 passwords and the test set had 322.805 passwords. In this case we got the same results as on the one to one distribution. After 1.000.000 tries to crack the passwords the Ispell cracking dictionary shows it advantages by having 91% more passwords cracked. The advantage is reduced to 26% after 1.000.000.000 tries.

4.2.3 Cross validation experiment

To evaluate the performance of our approach we also measured the number of passwords we could crack when learning the patterns of RockYou and try to crack a password file

Dictionary	**Passwords cracked**
Ispell	26.046
Wikipedia	29.793
RockYou	31.351
JtR incremental	27.638

Figure 4.7: Cracking Rootkit

from a different source. To measure these numbers we used the patterns we learned from the one to one distribution and used JtR to crack the passwords of the Rootkit leak. Using our approach with the Ispell dictionary we were able to crack 26.046 out of 84.403 passwords. Table 4.7 show the number of the other sources we used to crack the passwords. An interesting effect is that the RockYou leak could crack 31.351 passwords with 32.000.000 tries. This shows that the quality of the dictionary to crack passwords is the most important factor when cracking passwords. While we are able to improve a given dictionary, this improvement has its limitations.

5 Mitigation

Up to this point we talked only about attacking and how attacker can target the users. But there are also defense mechanism for users, programmers and administrators.

5.1 For users

Users are the weakest points in the chain. As researchers we can only try to teach the users. Some of them will listen while others simply don't care. For those users who care I will show up some easy to use techniques which help them to get different passwords on different sites. Passwords which are hard to guess, some of them even hard to remember.

An easy to remember rule for passwords is, that they should be treated like underware.

- Change them often,
- don't let them lay around,
- don't share them with friends,
- the longer the better, and
- keep them mysterious.

5.1.1 Password Card

The first technique is a password card, see Figure 5.1. There are some good sites on the Internet which generates the password card for you [Pasa]. The idea behind the password cards are that it contains all the passwords you need during your daily work. A passcard usually has the size of a creditcard. It is a pice of paper, which the user should encapsulate in plastic. While it is bad to write down a password, it is even worser to choose a weak password.

Lets take a look at such a passcard in Figure 5.1[1]. This passcard is divided into a set of symbol and rows with colors. To choose a password for a site the user chooses a symbol, which he thinks might represent the site for which he wants to use the password. As an example, the star or the note could represent YouTube. Next he chooses a color he associates with the site. The user needs only to remember the symbol and the color he had chosen. Lets assume the user had chosen the note sign and yellow (line 5) for YouTube. He would lookup this in the card and continue to the right. A valid and

[1] The passcard published here should not be used. It is just an example, to show how a passcard is working.

Figure 5.1: A sample password card

good password would then be: "QG8HdP5nVf". This password is 10 signs long, totally random and very hard to guess. Optional to enhance the security the user should add some special sign and easy to remember information.

Obviously it is very important to keep this card on person, and to take care that no marks are added to the card, in case a thief gets his hands to it. If the card is lost, generate a new one and change the passwords to the ones on the new.

The number at the bottom of the card is a identification string to this card. With this number the user can recreate the card from the website where he had taken the card from. One can put the number at the bottom of the card in a safe place, like a cash box in case the card gets lost.

While this is not bullet proof security this system is a compromise between strong passwords and security.

5.1.2 Length vs. number of characters

A common question is "Is it better to choose long passwords or is it better to use complex one?" Lets have first a look on the mathematical site. The formula for the size of the password space is number of characters ($|\Sigma|$) of the used alphabet to the power of the password length (n) or in other words:

$$|\Sigma|^n$$

Σ can be divided into 4 groups:

- Lower case alphabet – 26 characters,
- upper case alphabet – 26 characters,
- numbers – 10 digits, and

- around 34 other characters which can be reached on an American keyboard.

When we take only upper and lower case characters, all together 52. At the length 12 we get all together $52^{12} = 390.877.006.486.250.192.896 \approx 2^{68}$ possible passwords.

For the next number we take a full set, but only a 8 character length. This brings us to the following calculation: $96^8 = 7.213.895.789.838.336$ which is 54000 times smaller. Making it more important to choose a long password instead of a complex one.

On the other hand the non complex passwords are usually chosen within a set of language words. Taking just words reduces the complexity a lot. Taking this into consideration the user should use a long password, which does not appear in a dictionary, adding some numbers and special signs in between makes a password perfect.

5.1.3 Software

Software can be very helpful to manage and generate passwords. The big advantage of software is that the user has to remember only one password. Choosing a good password for this can protect all other passwords. The disadvantage of cause is a single point of failure. If a Trojan horse has infected the system which manages the passwords, the password file and the master password can be lost. But this is usually also effective for all passwords typed in by the user.

Generate passwords on Linux

Unix system come with a special device for random data. This mechanism can be used to create passwords with tools available on nearly all installations:

```
1 dd if=/dev/urandom count=1 2> /dev/null |\
2 uuencode -m - |\
3 sed -ne 2p |\
4 cut -c-14
```

Listing 5.1: Easy to use script to generate passwords

The first line used the dd command to generate some random data. dd copies 1 block (count=1) from /dev/random (if=/dev/random) to standard out (STDOUT). The last part drops additional output of dd to /dev/null, a device which deletes all inputs data.

Line two uuencode converts given data into a human readable format. In this case the -m parameter requests to use base64 encoding. Base64 encoding is a so called three to four encoding. It takes three bytes of input and converts it to 4 byte of output. the dash at the end tell the command that the data to encode is coming from standard in (STDIN). Output is written to STDOUT.

Line three reduces the input from STDIN by printing only the second line. sed is originally a tool to manipulate stream based on regular expressions. The tool manipulates the data from STDIN and forwards it to STDOUT. In this case the parameter tells the tool to output line two.

Line four reduces the input to 14 characters (-c-14). To generate password with a greater size this parameter can be changed to any number below 60. The user can even

parameterize the script to use a given input as the parameter, by changing this from -c-14 to -c-$1.

Keepass

Keepass is an example for a password management software, its current version is 2.18 as of March 13, 2012. It stores passwords in an encrypted database. When the user generates a new database he has to choose a master password. This password, hashed using SHA-256, is used as the key for the encryption algorithm. As encryption algorithm the user can choose between AES and Twofish. To prevent leaking of information, like user names and notes, the database is encrypted as a whole. The core features of Keepass are:

- Open source – while in the crypto community it is common to publish algorithm to the community, commercial software, in general does not publish it sources, making it impossible to validate the algorithms. Keepass as an open source software makes it sources available under the GNU general public license V2
- Strong cryptography – Keepass uses AES, Twofish and SHA256 to protect the passwords. The database is encrypted as a whole.
- Portability – Keepass can be downloaded as a zip file and as a version with installer. The zip file can be extracted to a USB stick, making it direct available on any windows system that provide an USB interface. Ports for Linux, Mac OS X, and even smartphones exists.
- Easy database transfer – The database is a file, which can be easily copied to another computer without special software.
- Strong random password generator – Keepass can generate passwords according to user input rules. By this the user can enter the restrictions a site gives to passwords and a strong password can be generated according to this rules.
- Support for user names, dates and notes – Together with the site the user can store a user name, creation, modification dates, and additional notes. All these data can be searched from within Keepass. All this data is encrypted inside the database, avoiding to be leaked if the data falls into the false hands.

Using this tool a user can have all his passwords with him all the time. Backups of the passwords are easy to maintain, by simple copying the file.

Using password management tools has also some disadvantages. By using the portability feature, the user might bring his passwords at a risk. Using it in an environment where the user should not trust the computers, like in Internet cafes the password file can be stolen, the master passphrase can be compromised by a keylogger. By using some additional notes an attacker could identify the sites where the passwords belong to.

The password manager might become a single point of failure for the user. If no backups are present a crash of the users file system deletes all passwords. Often the user

relies on his password database and by this does not keep additional notes for which sites he had generated his passwords, making him even forget where he had logins.

If a computer with a password manager is lost, an attacker could try to guess the password. In case it is a weak password an attacker can get access to all passwords and take over the accounts managed by the password manager.

Next to password management tools like Keepass there are online password managers. They provide nearly the same functionality, but have two additional flaws. First the user has to trust the password management site, in a way that they store the passwords in a secure way, that they do not give informations about the users passwords, and that they will not be hacked. The user will never be able to prove that the passwords are saved in a secure way. Even if the site would publish its sources the user could never be sure that the published source is the software running on the site. A user could use such a site for passwords, which are not important for him, like logins to sites where the login shall only personalize the information. For sites where password security is important, like logins for his job, the user should never use online password managers.

Firefox

Most of the passwords a regular user uses are passwords for websites in our days. Firefox, one of the most common browsers includes a mechanism to recognize and store passwords together with the user name. To protect the passwords from being leaked or simply copied by an intruder or thief, Firefox supports the usage of a master password. As soon as a master password is set, Firefox uses this as a key to a 3DES encryption. The user has to provide the master password once per program session. As always it is important to choose a strong master password. If no master password is set, Firefox stores all passwords in base64 encoding. The master password mechanism is good enough for the usage of accounts, which can not create any direct harm to him. As soon as it goes to important data it is recommended to use other stronger mechanisms to handle your passwords.

For the not so important sites we recommend the usage of the script from Section 5.1.3 together with a master password. For important systems like home banking, email and shopping accounts we would recommend the use of a good password manager.

5.2 For developers

A lot of programmers are creating software where they think only the good guys will use their software. This often leads to attacks like SQL injections and blind SQL injections, a technique to extract whole databases over an insecure website. One of the most important lesson is, not to trust a user. Specially to validate each and every user input. This does not only cover the data a user can enter in a form, but also all other data the user is sending to an web application and which is processed in any way, even a not validated cookie can be used for an SQL injection.

While this is one of the basic rules, a developer can not nessecarily assume that all developers are following that rule, therefore it is very important to protect important

data, like passwords inside the database.

5.2.1 Hashes

One of the most effective ways to protect passwords inside a database is to use a strong hash function. A strong hash function simply buys the users some time to change passwords he is reusing on other sites. As elaborated in the the Section 4.1.3, there are thousands of leaked passwords every month. In general there are the following hash algorithms available to use:

- MD5 – one of the weakest hash functions is specified in RFC1321, was invented in 1991. The first flaw was found in 1996. At this point cryptographer started to recommend the usage of other functions. In 2004 a flaw was found that made the usage of MD5 questionable. Nevertheless this flaw was about collision resistance.
- SHA-1 is currently the most widely used hash function. It was first published in 1995. It generates a 160 bit hash code from a password. There exist theoretical collision attacks against SHA1.
- SHA-2 is the next candidate in the SHA family. First published in 2001 it supports output sizes of 256 and 512 Bit.
- SHA-3 is the latest candidate of the SHA family. To select this candidate a competition was held, and the winner will be announced in 2012.

While MD5 is still better than no hashing, programmers should use at least SHA1 to hash passwords in a secure way. Using hash functions makes it impossible to use the passwords instantly. The attacker first needs to guess the passwords, giving the users some time after the leak to change their passwords on sites where they used the same. To achive this, hash algorithms use a one way function, which can be calculated only in one direction. By this a password can not be calculated back from a hash value. If hash functions are used an attacker has only the option to guess a password and hope that the guessed word is used as a password.

5.2.2 Salt

Salting a password is a technique were the password is extended by an additional string (salt) which differs from user to user. The salt is stored together with the password inside the database in clear text.

Whenever a user chooses a new password, the salt is added to the password before hashing. By using the salt it is not important how the salt actually looks like, it is only important that it changes from user to user. A good salt might for example be the username.

Cracking tools usually support the usage of salts as well. But using a salt the cracker has to calculate the hash value individually for every user, for every word he tries to guess. It also prevents the usage of rainbowtables. Especially when using the username

as a salt, the attacker will not gain any advantaged by using a rainbowtable to attack a set of leaked passwords. The attacker would have to create his rainbowtable for each salt which is in the leak. This removes the advantage of pre calculated hash rainbowtables since the attacker would have to take every possible salt into consideration.

5.2.3 3d password storage

Securing passwords in a database is one of the key problems in password usage. Using hash functions with salts makes it hard for an attacker to guess passwords, but it is still possible. We need another way to handle this problem if we want to increase security. One way is to secure the hashed and salted password with an additional encryption, where a further secret is involved.

We call this a three dimensional password storage. The first dimension is the hash of a password. Same password are hashed to the same result, making it easy for an attacker to identify users who have the same password. The second dimension is the salt, which is changed from user to user. The attacker is no longer able identify users which have the same password. The third dimension becomes an additional encryption or hash with a secret value. An attacker first need to break this encryption before he can try to guess a password.

The main problem in this approach is that this secret needs a good protection. If an attacker succeeds in getting his hands on the password, he would also succeed in getting this secret if it is stored on the computers involved. Therefore we recommend to use an additional hardware module. This module needs the following features:

- Encryption – the module needs to either encrypt or hash the input data with a fixed key or initialization vector (IV).
- Secure storage – the key or IV used by the device needs to be securely stored. Meaning that the key may be set by an operator in the beginning, but may under no circumstances be revealed to the system.
- Fast – especially on huge system where a lot of passwords have to be checked per second, the result needs to come fast.
- Configurable – it is important that an operator can set the key/IV used. On redundant systems it would be important that every machine has such a hardware module, making a fall back in case of hardware problems easy.
- Easy to use – the usage and integration into software needs to be easy.
- Backup – the device would need to backup the secret to an external data storage.

One way to implement the device would be to use an USB token, where it is not possible to extract the key. This device can be initially programmed and filled with the secret. The secret can be stored on an USB stick, stored in a safe and be used to fill

another token on a secure computer. It is important that the secret is not kept on any PC, since an attacker could try to attack this system.

When the users sets his password, the password will be hashed and salted. The hash value will be send to the Token. The token will secure the hash and send it back to the software. Now the secured hash can be stored inside the database together with salt. This brings a new dimension in password security.

An attacker who wants to crack a password, needs first to crack the new security layer. By using a strong encryption like 3DES or SHA512 the attacker would need either to crack the encryption password or find the special used initialization vector. The attacker will need to know at least one password, to calculate the hash value which has been secured. With this knowledge he would have the possibility to brute force the secret. By using 3DES he would have to brute force 168 bit. With current hardware this is not feasable.

5.2.4 Online applications

As already mentioned in Section 2.3.1 developers of online applications are able to fight online guessing attacks very well. One of the biggest key factors is that the application can count the number of faulty login tries. This number should be limited on user and on IP basis. In case that to many false tries arrive the IP address and the user account should be suspended for a certain amount of tries.

5.3 For administrators

Beside users and developers there is a third group of people which have to deal with passwords. Administrators especially in big organizations usually have to deal with a huge number of passwords, since nearly every application needs its own credentials.

5.3.1 Test systems

Test systems, which are used for various tests during software development, or to test implementations before they go to a production system, tend to be treated with less security. Usually a lot of people have access to these test systems. Most of them either do not care about security or have their priority on getting things done. In the later case they use even unsecure protocols if they are available if it is more convenient for them.

In an article about “25 Infosec Gurus Admit to their Mistakes and What They Learned from Them” [Tri], Chris Wysopal described the case that developer could use telnet on a test system when SSH was not available. Unfortunately telnet does not have any encryption. The communication is done in clear text. An attacker succeed to sniff the traffic and got the credentials of the user, which affected the reputation of the attacked site.

Also, test systems sometimes have so called generic users for testing. These users are often shared by a group of people, sometimes even worse they have generic passwords which can be easily guessed. While most people argue, that these systems are test systems

and that no harm can be done with them, test system sometimes have read access to productive systems, making them able to be used to leak confidential information.

5.3.2 Single sign on

Users have to deal with a huge amount of passwords. A way to help your users is to use a single sign on mechanism. In single sign on systems the user has to authenticate himself only against one application, which takes care that all other applications requiring credentials accept the user without entering his data again. The advantages are that the users need to remember only one set of credentials instead of several. On the other hand if these credentials are leaked the attacker has access to all applications the user would have access to.

Microsoft Live ID is one example for such a single sign on mechanism. Users who have a login for this service can access the services of MSN, Xbox Live, Zune Marketplace and several other services by login in once during a session. As soon as the user is logged in he can access these services if he has an account on them. There is also a module for Windows XP and later versions to link the account on the PC to the Live ID. By this he can access the services as soon as he is logged in to his PC.

5.3.3 Two factor authentication

Another approach to enhance the security of company system is to use two factor authentication. While in the office one of this factors can be the IP address of the office PC or the port where the computer is connected to, there could even be a number of allowed offices for this user.

When the user is outside his office the administrator can provide their users with special devices or smart cards which will be the second factor beside the password. A good and cheap solution for this could be the yubikey (see Section 6.2), a device that works like a USB-keyboard and generates one time passwords.

Modern laptops are sometimes already equiped with smart card readers or finger print readers. While an attacker can easily get his hand on the fingerprints without the knowledge of a victim, it takes some effort to copy a smart card. All these arrangements can provide additional authentication factors to secure login processes and to make it nearly impossible for an attacker to access a company.

5.3.4 Password policies

Policies are rules how a password has to look like when it is chosen. The administrator usually can choose the rules out of a predefined set. The options he has depend highly on the system he is using. Widely used rules are:

- Minimum length of x characters,
- include numbers,
- include capital and non capital letters,

- include special letters,
- not contain names, the company name, and
- change password every y days.

At a first look this seems to be a good option to enhance the password security. Unfortunately this is not always the case. User tend to be very creative in choosing passwords which are easy to remember and often therefore being weak. For example the word "Passw0rd!" matches all criteria from above. It includes small and capital letters, a number and even a special sign. But this password can be guessed easily and therefore gives a elusory security.

If the password policies are communicated to the users when changing the password, an attacker can use this to his advantage. He can create a manipulation algorithm to change the words he is using inside a dictionary attack to match only words which fit to the rules. By this the security which was intended draws back on him and actually reduces the search space of an attacker. For example, if the password policy requires a minimum of 8 letters the attacker can ignore every word with less then 8 characters, or more generally he can ignore every word that does not match the entire policy. In [WACS10] Matt Weir analyzes the effectiveness of using the NIST entropy to measure the security of a password, taking password policies into consideration.

A better option on policies would be to use the so called "guessing entropy". The guessing entropy measures how hard it is to guess a password. By this a random password like "pVKkvK6V" has a much higher entropy as "Passw0rd!" also words are in general not forbidden but they have to be combined with enough other data to be safe. An example for such a password of words would be "correctbatteryhorsestable". Even that our approach tries to crack these passwords the number of passwords to be tested is so huge that the passwords would likely not be crack in a reasonable time.

6 Related work

During this research we had a look at a lot different approaches on password cracking. In this part we are going to discuss the work of Matt Weir and some alternatives to passwords. Furthermore we are going to discuss special algorithms which are used to crack passwords, followed by a look at side channel attacks.

6.1 Matt Weirs work

Matt Weir et al. described in his paper [WAMG09] a way to manipulate a dictionary to increases the success rate on password guessing. For this purpose the algorithm learned patterns from a given sample set. These pattern described the way passwords are looking. He divided a password in groups of letters, numbers and special signs for each password. For each group the pattern contained the size of this group. When a password would have to be guessed he took the most likely pattern and replaced the groups with words, numbers and signs. With his approach he managed to crack between 29% and 129% more passwords compared to a classic guessing approach with the same number of guesses.

6.2 Password alternatives

Password are the most popular but also one of the most vulnerable authentication mechanisms. Therefore a lot of people are discussing the alternatives. Unfortunately the alternatives are not always user friendly. In this section we will have a look at some of the alternatives to passwords.

Pico

Pico is an idea from Frank Stajano [Sta11]. Pico will be a device which manages user credentials for different sites. The device will communicate with a computer by radio. If a users wants to login to a remote system the user will be presented with a QR code. The QR code is scanned by the user. The QR code identifies the site and the device identifies by this the needed credentials. A plugin/driver sends the credentials encrypted to the remote site enabling the login for the user. To protect the users identity the username and password are chosen randomly when a user decides to sign up to a new service.

YubiKey

YubiKey is a small USB device sold by YubiCo [Yub]. It identifies it self to a computer as a USB keyboard, making it available to any computer without the need of an additional

driver. The purpose of the device is to provide a one time password (OTP). One Time passwords are only valid for one use. While a standard password can be captured by a Trojan horse or a virus and used afterwards by an attacker, the OTP is only valid for a single use. To achive this goal the device generates a 128 Bit code at each time of use. The code contains the following components:

- a hidden identity field,
- a counter which is reseted at each power up,
- a counter stored on the device,
- a time based non-predictable counter,
- a random seed, and
- a checksum.

This code is encrypted with AES-128. The device sends a 12 character identity together with an encoding of the 128 Bit message to the computer.

To use this device on a system, the user attaches the YubiKey to a computer. He goes to the application where he wants to use the device, selects the input field for the YubiKey, and presses the button on the device. Usually YubiKey is used in a two factor authentication, meaning that the user still uses a login Id and a password. When the login information has been sent to the application, there are two ways to handle it.

For small organizations and single user usage, YubiCo provides a server which can validate the correctness of the data. To verify the OTP a developer can include an API which is available for the different languages including but not limited to C, Java, PHP and others. Special API modules are additionally provided for PAM, the Unix authentication mechanism. For big organizations there are hardware security modules and radius servers available. These devices enable the organization to be independent from the YubiCo servers, providing a even higher level of security.

6.3 Special techniques for guessing passwords

There are two special techniques known to improve the speed of cracking hashed passwords. Both of them have some advantages and disadvantages.

6.3.1 Rainbowtables

Rainbowtables are a technique to crack hashed passwords [NS05]. For hashed passwords there is no way to calculate the source data out of a hash value. A good first approach seems to be to precompute hash values from passwords and store them in a table. Due to the huge amount of data this approach is resulting in, we need a way to reduce the amount of data. At this point rainbowtables come in.

A rainbowtable is mainly a set of password hash pairs where the amount data is dramatically reduced. to achive this goal we define a set P which contains all possible passwords, we are considering for the table. For our purpose we limit our self to one specific hash function h. This hash function will map a password p to the a set of hash values H. Furthermore we need a reduction function r which is able to map a given hash k back into the P.

The idea behind rainbowtables is that we are make the following transformation:

$$p_0 \rightarrow k_0 = h(p_0) \rightarrow p_1 = r(k_0) \rightarrow ... \rightarrow k_n = h(p_n)$$

with

$$p_0, ..., p_n \in P$$

and

$$k_n, ..., k_n \in H$$

p_0 and k_n are stored inside the database. We call this transformation a chain. Using these chains we reduce the amount of data nearly by the factor n. Due to the fact that a hash value usually has a higher amount of bits than the passwords, we will encounter that in some cases chains with different starting passwords will end with the same hash value.

After this precomputation we are now able to crack passwords. For this we will use the following algorithm:

```
take a given hash value k
k' = k
while(k' is not in table)
{
  calculate k' = h(r(k'))
}
For all p which match k' inside the table
{
  For i from 0 to n
  {
    k2 = h(p)
    if k2 = k
    {
      return p
    }
    p = r(k2)
  }
}
return not found
```

Listing 6.1: Cracking with rainbowtables

With this algorithm we are recreating the chain inside the memory. The result will be a password which results in the given hash value.

6.3.2 Special hardware

The second technique is to use a graphic card to speed up the cracking process. Modern graphic cards are designed to speed up the rendering process for computer games. They

are specially designed to make arithmetic operations on a huge set of data very fast. In 2006 NVIDIA came up with the first graphic card with the Compute Unified Device Architecture (CUDA).

The idea of CUDA was, to have general purpose computing on a graphics processing Unit (GPU). A GPU is basically a special processor which is able to perform mathematical complex tasks in a very short time. These tasks are mainly but not limited to vector operations, matrix manipulation and rendering. For this the GPU has several computations units which work in parallel. For maximum speed each of these units need to perform the same operation. As a result of this, programs running on a GPU need to special optimized. The developer needs to think which operations can be done in parallel to get maximum performance. Using these technique the algorithm can be 100 times faster, then on a regular PC. CUDA is specially designed for NVIDIA graphic cards. The language used is similar to C. Since version 4.1 NVIDIA has opened the source code for its CUDA compiler [NVI]. The goal is to make CUDA available for other platforms.

The chair Embeded security from the Ruhr University of Bochum created a device called COPACOBANA (Cost-optimized Parallel Code-Breaker). The device contains up to twenty FPGA modules, each with six low cost FPGAs. A FPGA, short for field programmable gateway, is a integrated circuit. These circuits can be configured to perform special tasks with a special programming language called hardware description language (HDL). After the configuration is finished the FPGA acts like a special purpose circuit. Unless computer programs these FPGAs do not have a code running, instead they change their inner logic to build the functionality in hardware. By this it is possible to get a special chip to compute DES and other cryptographic algorithms in a very fast way. A PC connected to the FPGAs controls the input and the results.

6.4 Side channels

Talking about side channel attacks against passwords is always about the system to be attacked and not the passwords itself. The attack reveals the password or at least to get a way around them.

6.4.1 FireWire on Windows

The prerequirement for a firewire attack is that a firewire interface is attached to the computer or in case of laptops that a firewire driver is present. The firewire interface is specified to support direct memory access (DMA). This system was initially invented to allow a device to access the memory without putting load on the processor. The DMA has to be initiated by the processor.

An attacker connects to the victims computer via firewire. He configured his computer to act as a device which supports firewire. The victims computer automatically loads the driver (if necessary) and gives permission to direct memory access. The attacker starts a software like 'Inception' [Car]. This software starts to search the memory of the target machine for the signature of the login procedure in the lower 4GB memory. As soon as the signature is found, the jump to the relevant program code for rejecting a password is

overwritten by NOP, the assembly code for NoOPeration. The attacker can now login to the computer with any password.

If no firewire interface is attached to the computer, there is another possibility to make this attack work. If a PC Card interface is present an attacker can insert a PC Card which provides a firewire interface.

Possible mitigations against this attack are to physically disable firewire and other interface which are accessible from outside the computer and provide DMA or to disable firewire on operating system level.

While this attack was working on Linux as well, some time ago the Linux kernel has been patched to disable DMA by default.

6.4.2 Cold Boot Attack

Cold boot attacks are designed to circumvent disk encryption software. They require a computer where the victim already entered the password for the disk encryption. The attacker needs physical access to the system. The attacks starts by opening the computer and finding the memory chips inside. The attacker uses a cooling spray to reduce the temperature of the chips. If temperature of memory chips is dropping under a certain degree, they keep the data stored inside even if the power supply is taken away [Sko02]. The attacker now extracts the chips and inserts them into a computer he can control.

The next step to the attacker is start a software on his computer where he just added the frozen Memory chips. Now he can search for the signature of the AES key schedule and extract the AES key. By this attack the password itself is not revealed. But by getting to the AES key the attacker can decrypt the harddisk and has access to its content.

6.4.3 Evil maid

Similar to the cold boot attack evil maid [Joa] is targeting disk encryption systems. It was first presented by Joanna Rutkowska from Invisible lab. In this scenario the attacker has access to the computer of the victim. He starts the computer from an USB stick or a CD. After starting the evil maid software overwrites the bootloader with a modified version. As soon as the victim starts the computer he is presented by the bootloader screen he is expecting. He enters his password. The modified bootloader saves the password at an empty place on the harddisk and starts the original routines.

The next step for the attacker is to get again access to the computer of the victim. He reads the password previously written and recreates the original boot loader. Afterwards the attacker can decrypt the harddisk with the password of the user.

7 Conclusion

The biggest challenge on cracking passwords is to get a highly sofisticated dictionary. An attcker who has only a small dictionary will be able to enhance this with our approach. On small dictionary we were able to show a significant improvement on cracking the paswords. Nevertheless if the dictionary is already in a very good shape our approach took only effect at the end. If an attacker is limited in time for his attack he will have a bigger success with a specially build dictionary. If time is not the problem our approach can improve the number of cracked password.

As we showed in Section 4.2.3 learning patterns on RockYou and cracking the passwords from the Rootkit source using already leaked passwords showed the best result compared to the number of words tested.

During this research we evaluated how an attacker can use techniques to improve the guessing of passwords by combining words together. While we were able to show a significant improvement on a small dictionary this could not be reproduced on a more sophisticated dictionary.

7.1 Future work

During the work on this thesis, some topics came up where the time was missing for an indeep investigations.

We did take a look at combined passwords without taking the gramatics into consideration. Another approach could be to look at Markov chains to combine passwords. Markov chains are basing on Markov models.

As already described in Section 5.2.3 we thought about a new way to store passwords more secure. Encrypting or further hashing with a fixed password or initialization vector, which can not be extracted from a black box seems to enhance the security of passwords. This topic by itself is highly related to password security but would have driven the work into another direction. We described only the basic idea and there are still some questions open:

- In a comporate environment there is a huge amount of passwords, which have to be validated in a short periode of time. How can a feaseble performance be achived, without disturbing the users.
- Does it make a difference with respect to performance and security to use a hash or an encryption function?
- Can a redundant service be build with this model?

- Can the idea be build into hardware?

While looking for password data for this work, we stumbled about pastebin.org. Pastebin is a website which allows user to publish text files anonymously. We found that a minimum of 100.000 passwords are leaked each month. Some of them contained hashed passwords, some of them even clear text passwords. We tried to identify the sources of these leaks. For hashed passwords the leak most likely was taken from vulnerabilities of some websites. Also some of the clear text passwords came from this sources.

At some point in time, we found user password combinations which were enhanced by additional data like the name and version of the browser or the name of the PC. Taking a look at the HTTP header of a regular request, the name of the computer is usually not transfered to a target side. One possible reason why they are inside the leak could be malware on the computer from where the user tried to login. Other possible sources for leaks in general could be rogue proxy servers and rogue tor exit nodes. To evaluate and investigate the possible sources a extended preparation setup would be needed. We would have to try to find which malware caused the leak. We would need to setup accounts on different mail provider and try to identify the rogue servers.

List of Figures

List of Listings

Bibliography

[And08] Ross Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems*. Wiley, 2008.

[Ars] Ars Technica. Constitutional showdown voided: Feds decrypt laptop without defendants help. `http://arstechnica.com/tech-policy/news/2012/02/constitutional-showdown-voided-feds-decrypt-laptop-without-defendants-help.ars`, as of March 13, 2012.

[Car] Carsten Maartmann-Moe. Inception – break and enter. `http://www.breaknenter.org/projects/inception/`, as of March 13, 2012.

[Cir] Cirt. Cirt: default password list. `http://cirt.net/passwords`, as of March 13, 2012.

[Dar] Darknet. Password cracking wordlists and tools for brute forcing. `http://www.darknet.org.uk/2008/02/password-cracking-wordlists-and-tools-for-brute-forcing/`, as of March 13, 2012.

[Elc] Elcomsoft. Homepage Elcomsoft. `http://elcomsoft.com/`, as of March 13, 2012.

[FH07] Dinei Florencio and Cormac Herley. A large-scale study of web password habits. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 657–666, New York, NY, USA, 2007. ACM.

[Heia] Heise. Backdoor in TRENDnet IP cameras. `http://www.h-online.com/security/news/item/Backdoor-in-TRENDnet-IP-cameras-1428896.html`, as of March 13, 2012.

[Heib] Heise. Video conferencing systems as spying tools. `http://www.h-online.com/security/news/item/Video-conferencing-systems-as-spying-tools-1421346.html`, as of March 13, 2012.

[Joa] Joanna Rutkowska. Why do I miss Microsoft BitLocker? `http://theinvisiblethings.blogspot.com/2009/01/why-do-i-miss-microsoft-bitlocker.html`, as of March 13, 2012.

[KDM03] Steve Wozniak Kevin D. Mitnick, William L. Simon. *The art of deception*. John Wiley & Sons, 2003.

[Mar06] A.A. Markov. Rasprostranenie zakona bol'shih chisel na velichiny, zavisyaschie drug ot druga. *zvestiya Fiziko-matematicheskogo obschestva pri Kazanskom universitete, 2-ya seriya, tom 15*, page 135–156, 1906.

[Mar71] A.A. Markov. Extension of the limit theorems of probability theory to a sum of variables connected in a chain. *reprinted in Appendix B of: R. Howard. Dynamic Probabilistic Systems*, 1, 1971.

[Mat10] Matt Weir. *Using Probabilistic Techniques to Aid in Password Cracking Attacks*. PhD thesis, Florida State University, 2010.

[Mil94] G A Miller. The magical number seven, plus or minus two: some limits on our capacity for processing information. 1956. *Psychological Review*, 101(2):343–52, 1994.

[NS05] Arvind Narayanan and Vitaly Shmatikov. Fast dictionary attacks on passwords using time-space tradeoff. In *Proceedings of the 12th ACM conference on Computer and communications security*, CCS '05, pages 364–372, New York, NY, USA, 2005. ACM.

[NVI] NVIDIA. New cuda toolkit unleashed. `http://developer.nvidia.com/content/new-cuda-toolkit-unleashed`, as of March 13, 2012.

[Oxf] Oxford. FTP: wordlist. `ftp://ftp.ox.ac.uk/pub/wordlists/`, as of March 13, 2012.

[Pasa] PasswordCard.org. Your Password Card. `http://www.passwordcard.org/en`, as of March 13, 2012.

[Pasb] Pastebin. #1 paste tool since 2002. `http://www.pastebin.com`, as of March 13, 2012.

[Sec] Sectools. ecTools.Org: Top 125 Network Security Tools. `http://sectools.org/tag/crackers/`, as of March 13, 2012.

[Sho] Shodan. Expose online devices. `http://www.shodanhq.com/`, as of March 13, 2012.

[Sko02] Sergei Skorobogatov. Low temperature data remanence in static RAM. Technical Report UCAM-CL-TR-536, University of Cambridge, Computer Laboratory, June 2002.

[Sku] Skull Security. Passwords. `http://www.skullsecurity.org/wiki/index.php/Passwords`, as of March 13, 2012.

[Sof] Softpedia. FBI Unable to Decrypt Brazilian Bankers Data. `http://news.softpedia.com/news/FBI-Unable-to-Decrypt-Brazilian-Banker-s-Data-145640.shtml`, as of March 13, 2012.

[Sta11] Frank Stajano. Pico: No more passwords! *Proceedings of Security Protocols Workshop*, 2011.

[Tri] Tripwire. 25 Infosec Gurus Admit to their Mistakes and What They Learned from Them. `http://www.tripwire.com/state-of-security/it-security-data-protection/25-information-security-blunders/`, as of March 13, 2012.

[Uni] University of Leipzig. Deutscher Wortschatz. `http://wortschatz.uni-leipzig.de/`, as of March 13, 2012.

[WACS10] Matt Weir, Sudhir Aggarwal, Michael Collins, and Henry Stern. Testing metrics for password creation policies by attacking large sets of revealed passwords. In *Proceedings of the 17th ACM conference on Computer and communications security*, CCS '10, pages 162–175, New York, NY, USA, 2010. ACM.

[WAMG09] Matt Weir, Sudhir Aggarwal, Breno de Medeiros, and Bill Glodek. Password cracking using probabilistic context-free grammars. In *Proceedings of the 2009 30th IEEE Symposium on Security and Privacy*, pages 391–405, Washington, DC, USA, 2009. IEEE Computer Society.

[Wika] Wikipedia. AaaaaAAaaaAAAaaAAAAaAAAAA - A Reckless Disregard for Gravity. `https://en.wikipedia.org/wiki/AaaaaAAaaaAAAaaAAAAaAAAAA!!!_%E2%80%93_A_Reckless_Disregard_for_Gravity`, as of March 13, 2012.

[Wikb] Wikipedia. Wikipedia database download. `http://en.wikipedia.org/wiki/Wikipedia:Database_download`, as of March 13, 2012.

[Xav] Xavier Garcia. When a shell is not enough. `http://www.shellguardians.com/2011/07/monitoring-pastebin-leaks.html`, as of March 13, 2012.

[xkc] xkcd. Password Strength. `https://xkcd.com/936/`, as of March 13, 2012.

[Yub] YubiCo. Homepage YubiCo. `http://yubico.com/`, as of March 13, 2012.