UNIVERSIDADE DA BEIRA INTERIOR
Engenharia

Password Habits and Cracking Toolkit

Ricardo Xavier Paiva dos Santos

Dissertação para a obtenção do grau de Mestre em
Engenharia Informática
(2º Ciclo de Estudos)

Orientador: Prof. Doutor Pedro R. M. Inácio

Covilhã, Outubro de 2015

Dedication

"... Dedicated to my beloved Parents and Sister ..."

Acknowledgements

The first person to whom I would like to thank, for the realization of this dissertation, is to Professor Doutor Pedro R. M. Inácio, my supervisor, for all the tireless and precious support as also the opportunity to work and learn with him.

I would like to thank to Pedro Tavares who helped me at the beginning of this project and Diogo Fernandes, for all the valuable advices for achieving the toolkit PassCrackGUI as well other projects. Another gratification is to Miguel Neto for all the talks, in which there was great value on achieving the PassCrackGUI and related works.

The last and not the least, and because the dissertation is not the road traveled but the culmination of the work done over the past five years in college, i want to dedicate also to my fallow friend Hugo Paulino for all the support and friendship.

Resumo

As palavras-passe desempenham, hoje em dia, um papel importante em sistemas informação. Estas estão muitas vezes na base de mecanismos de controlo de acesso e constituem frequentemente o primeiro factor *something you know* de mecanismos de autenticação. São chaves para computadores, sistemas de software, informação confidêncial e até para edifícios, e a sua adoção generalizada torna a sua descoberta um dos principais objetivos da fase inicial de ataques informáticos e uma área de investigação muito interessante. Por um lado, dado que as palavras-passe são sequências de caracteres com as quais valores fornecidos por utilizadores têm de ser comparados, a sua representação tem de ser guardada em sistemas computacionais; por outro, dada a sua natureza sensível, estas têm de ser guardadas de uma forma segura. Ao invés de guardar as palavras-passe em texto limpo, é comum e preferível guardar transformações destas sequências de caracteres, obtidas através de funções com propriedades muito especificas, tais como funções de cifra ou resumo criptográficas. Existem vários métodos conhecidos e documentados hoje em dia para a execução desta tarefa, descritos na literatura da especialidade e considerados seguros, embora estas não sejam sempre corretamente utilizadas. Assim, a obtenção de uma palavras-passe a partir da representação constitui normalmente uma tarefa computacionalmente inviável. O compromentimento de palavras-passe (do inglês *password cracking*) é então tentado através da submissão repetida de diversas palavras já conhecidas (usando dicionários ou compendios) ou padrões à função de transformação, comparando o seu resultado com a representação capturada, até que uma correspondência seja encontrada ou as possibilidades se esgotem. Assim, a segurança dos mecanismos usados para a obtenção das representações está dependente do quão previsíveis as palavras-passe são.

Esta dissertação aborda temas relacionados com hábitos de construção de palavras-passe e ferramentas de *password cracking*. Muitas ferramentas especializadas de *cracking* estão disponíveis nos dia de hoje, sendo muitas delas gratuidas ou código aberto, desenhadas apenas para interação em linha de comandos. Uma das principais contribuições deste trabalho foi o desenvolvimento de uma interface gráfica para diversas ferramentas de *cracking* (como o *Hashcat*, *John the Ripper* e *RainbowCrack*), reunindo as suas funcionalidades mais interessantes de uma forma concisa e inteligente. A ferramenta desenvolvida, designada por *PassCRackGUI*, foi usada com o intuito de descobrir palavras-passe em diversas bases de dados contendo representações, e que vazaram para a Internet em 2014 e 2015. Este estudo foi feito com a intenção de analisar o quão expostas as respetivas palavras-passe estão e também de perceber os hábitos dos utilizadores na construção destas sequências de caracteres. Para um melhor estudo deste último tópico, foi preparado e entregue um questionário a 64 participantes. A análise dos resultados deste questionário constitui outra contribuição deste trabalho.

PassCrackGUI é o principal resultado deste programa de mestrado. É totalmente funcional, fá-

cil de usar e está disponível gratuitamente como um projeto *open source*. Foi desenvolvido em *Java* e testado nos sistemas operativos *Linux*, *Windows* e *Mac OS*. Quando usado na tentativa de *cracking* das bases de dados vazadas, foi possível recuperar 36% de 4233 representações de palavras-passe, apenas utilizando dicionários e simples regras num computador portátil vulgar. Parte do problema reside nos mecanismos adotados para a obtenção das representações, já ultrapassados na maioria dos casos; enquanto que a existência de palavras-passe fracas também contribuiu para este número (e.g., um significante número de palavras-passe eram constituídas por 4 dígitos apenas). Os resultados do questionário estão em conformidade com outros trabalhos nesta área, nomeadamente em termos de esteriótipos. Por exemplo, as respostas sugerem que os homens usam palavras-passe com maior diversidade e comprimento do que as mulheres. Ainda assim, vários aspectos contraditórios nas respostas levam à conclusão que os participantes parecem estar a alegar usar palavras-passe mais fortes do que usam realmente.

Palavras-chave

Código de Autenticação da Origem da Informação Baseada em *Hashing*, *Cracking*, Crunch, Função de Derivação de Chaves de Cifra a Partir de Palavras-Passe, Função Resumo, Hashcat, HMAC, Interface de Utilizador, John the Ripper, PassCrackGUI, Palavra-passe, PBKDF

Resumo Alargado

Introdução

Este capítulo, escrito em Língua Portuguesa, pretende explicar sucintamente toda esta dissertação de uma forma mais alargada e completa que o Resumo. Para proceder a uma sintetização mais organizada, este capítulo foi dividido por secções em concordância com as existentes no corpo da dissertação.

Primeiramente, são abordados os objetivos pretendidos com este trabalho, bem como o seu enquadramento e a respetiva descrição do problema. De seguida serão apresentadas as principais contribuições e é dado o contexto do respetivo tema. Evolui-se para a descrição da ferramenta desenvolvida e dos testes feitos, bem como para a discussão dos resultados obtidos no contexto da sua aplicação no *cracking* de bases de dados reais com palavras-passe. Perto do final, são apresentadas algumas conclusões sobre o trabalho desenvolvido e referidas algumas linhas de trabalho futuro.

Descrição do Problema, Enquadramento e Objetivos

Cada vez mais existem utilizadores a usufruírem da Internet. A sua utilização e dos seus serviços, tornaram-se mesmo uma dependência para muitas pessoas. A utilização de vários serviços está normalmente associada a um registo (um nome de identificação -- *username* -- e uma palavra-passe), dados cuja proteção é essencial. Contudo, a fraca noção da importância desta informação e dos perigos a que pode estar exposta por parte do utilizador, bem como o por vezes fraco conhecimento do funcionamento dos serviços em que se regista, pode levar a que este seja descuidado na escolha ou manuseamento na sua palavra-passe. Mais, a por vezes fraca perceção e conhecimento dos programadores na área da segurança leva também à implementação de mecanismos fracos de armazenamento ou transmissão de palavra-passe, aumentando também o perigo de exposição ou comprometimento. A verdade é que não faltam formas alternativas de autenticação nos dias de hoje, métodos seguros de armazenamento de palavras-passe, e até ferramentas de comprometimento ou tentativa de encontrar palavras-passe, conhecidas na gíria da especialidade como ferramentas de *cracking* de palavras-passe.

A utilização de ferramentas de *cracking* pode ser uma forma de sensibilizar as pessoas para este perigo. O contacto com esta realidade pode ajudar a consciencializar alguns utilizadores e programadores sobre os perigos da utilização de fracas palavras-passe ou de métodos fracos ou deficientes de armazenamento de palavras-passe em computadores, respetivamente. Contudo, a utilização deste tipo de ferramentas requer normalmente um conhecimento prévio nesta área. Uma evidência deste facto é que a maior parte destas ferramentas apenas permitem interação em modo linha de comandos, e as interfaces daquelas que as disponibilizam são difíceis de compreender e utilizar.

Tomando o contexto referido antes como motivação, foram definidos os objetivos de desenvolver uma interface de utilizador, bem como toda a lógica de suporte, para várias ferramentas de *cracking* de palavras-passe, simultaneamente integradora e simples de utilizar (no sentido de pessoas com menos conhecimentos serem capazes de a usar). Para além deste objetivo, foi ainda definido que se faria um estudo sobre os hábitos de construção de palavras-passe passe de utilizadores Portugueses, nomeadamente através da realização e análise de um questionário. Este trabalho está portanto enquadrado na área da segurança informática, em tópicos de autenticação e de controlo de acesso. Para se alcançarem os objetivos deste trabalho, este foi estruturado em diversas partes:

- Elaboração de um estudo do trabalho já desenvolvido sobre o tema, para contextualização;
- Pesquisa de ferramentas de *cracking* de palavras-passe disponíveis, bem como a sua análise com vista à sua possível incorporação na interface a desenvolver;
- Realização de um estudo para a elaboração do aspeto gráfico da interface. Desenvolvimento da interface gráfica e do código de suporte; testes e refinamentos;
- Teste da ferramenta desenvolvida com bases de dados que vazaram para a Internet e análise dos resultados. Criação e disseminação de um questionário sobre hábitos de palavras-passe, e análise dos seus resultados;
- Escrita da dissertação e de dois artigos científicos de conferência.

Principais Contribuições

A principal contribuição deste trabalho foi o desenvolvimento de uma interface gráfica para o uso de ferramentas especializadas em *password cracking*. A *PassCrackGUI*, como de resto essa ferramenta foi designada no contexto deste trabalho, permite uma utilização, intuitiva e integrada, de ferramentas como *John the Ripper*, *Hashcat*, *RainbowCrack*, entre outras.

Com o desenvolvimento do *PassCrackGUI*, e através da sua utilização, foi possível elaborar um estudo sobre os hábitos de construção de palavras-passe. Este estudo, foi elaborado de duas formas distintas: uma através de bases de dados comprometidas, onde foram analisas as palavras-passe descobertas através da *PassCrackGUI*; e a outra, através de um questionário feito a 64 pessoas. Os resultados do estudo, e sua análise, constituem outra contribuição deste trabalho de mestrado.

Em consequência dos dois pontos referidos anteriormente, foram elaborados dois artigos científicos para publicação: o primeiro, intitulado *"PassCrackGUI — A Graphical User Interface for Password Cracking Tools"*, foi submetido e apresentado na conferência *Conftele (2015)* [Ric15b]; o segundo, com o título *Analysis of Password Habits and Leaked Databases*, foi submetido e ap-

resentado na conferência *INForum (2015)* [Ric15a].

Trabalho Relacionado e Contexto

O Capítulo 2 da dissertação discute o tema da segurança das palavras-passe, e inclui uma descrição de funções e métodos normalmente utilizados na sua proteção, reforçando um pouco melhor a motivação para o trabalho. No decorrer dos anos, desde a invenção das palavras-passe, as soluções para autenticação de utilizadores passaram, quase sempre, por melhorar a segurança no seu armazenamento, embora a literatura seja rica em métodos de autenticação alternativos, mas aparentemente mais difíceis de usar ou implementar. Mesmo com a utilizações de funções de *hash*, e de se associar um *salt*, para salvaguarda de representações seguras das palavras-passe, os *crackers* foram desenvolvendo mecanismos que permitiam a sua descoberta. Dois dos factores que mais ajudaram o desenvolvimento do *cracking* de palavras-passe foram a evolução do poder de processamento nos computadores e o uso de palavras-passe repetidas ou fáceis de adivinhar por parte dos utilizadores. O capítulo também elabora neste assunto.

A partir de metade do capítulo são descritas algumas das ferramentas mais populares de *cracking* de palavras-passe, bem como alguns utilitários com elas relacionados (e.g., para geração de dicionários ou de listas de palavras a partir de máscaras de padrões). A discussão inclui ferramentas *online* (que são por exemplo usadas para tentativas de *cracking* em tempo real a serviços) e *offine* (que são, por exemplo, aplicadas a representações de palavras-passe vazadas). As ferramentas que a PassCrackGUI integra são aqui descritas com mais detalhe. No meio do capítulo são referidos os incidentes mais significativos em termos de bases de dados com palavras-passe que vazaram para a Internet nos últimos. anos.

Interface Integrada para Ferramentas de *Cracking*

O capítulo 3 da dissertação aborda o funcionamento da interface gráfica desenvolvida e explica com mais detalhe cada uma das funcionalidades que fornece. Através de diagramas (de casos de uso e de actividades), e num contexto de engenharia de software, são explicadas as principais funcionalidades do *PassCrackGUI*. A utilização de diversas imagens ao longo do capítulo ajuda a detalhar cada funcionalidade da ferramenta, bem como a localizá-las na interface desenvolvida. Dão-se provas da simples metodologia utilizada na sua construção, que permite uma rápida aprendizagem do seu funcionamento.

É neste capítulo que também são descritos, com detalhe, alguns dos parâmetros da ferramenta, explicitando melhor os requisitos necessários à sua execução ou utilização. A ferramenta implementada é multi-plataforma, e a execução em sistemas operativos *Windows* e *Linux* é igualmente ilustrada e explicada através de capturas de ecrã. A *PassCrackGUI* permite a diversas pessoas terem acesso a várias ferramentas de *password cracking* de código aberto de uma forma simples e integrada, sendo esta uma das maiores contribuições desta dissertação.

Teste e Análise de Resultados

O capítulo 4 da dissertação apresenta os resultados dos estudos feitos sobre hábitos de construções de palavras-passe. Os resultados estão divididos por duas grandes secções. A primeira diz respeito à análise de uma série de bases de dados contendo palavras-passe de instituições ou organizações Portuguesas que vazaram para a Internet entre 2014 e 2015, e que à data da escrita desta dissertação ainda estavam disponíveis para descarregar. A segunda resume a análise feita através de um questionário desenhado para auferir alguns hábitos em termos de construção destes importantes pedaços de informação. Foi interessante notar que alguns resultados, obtidos de uma e outra forma, eram discordantes. Por exemplo, enquanto nas palavras-passe descobertas através da aplicação do PassCrackGUI às bases de dados vazadas, a diversidade de caracteres utilizados era pequena, as palavras-passe sugeridas nos questionários eram muito mais ricas, em termos de diversidade e de comprimento.

Através destes resultados, e como evidenciado em casos de estudo existentes na literatura, foi possível concluir que os utilizadores (humanos que são) alegam, normalmente, usarem palavras-passe mais fortes do que provavelmente usam. Outra possível abordagem que pode ser considerada, consiste na amostra das pessoas que responderam ao questionário, quer pela faixa etária, quer por outro sexo. Na verdade, os resultados sugerem que os homens usam palavras-passe mais fortes ou, alternativamente, que mentem mais relativamente a este aspeto.

Outro importante facto interessante notado é que a maior parte das bases de dados comprometidas continham representações obtidas através de mecanismos ultrapassados e considerados inseguros em 2015. Uma das funções de *hash* mais utilizadas era a Message Digest 5 (MD5). Isto talvez seja um reflexo de que a segurança dos sistemas (do qual vazaram) se reflete em todas as suas componentes.

Conclusões e Trabalho Futuro

No decorrer do capítulo 5 são apresentadas as conclusões bem como algumas linhas de trabalho futuros. É dito que os objetivos principais do trabalho de dissertação foram atingidos com sucesso. A interface gráfica de utilizador construída as mais populares ferramentas de *cracking* de palavras-passe *offline*, bem como o seu código, foi testada em diversos sistemas operativos e disponibilizada gratuitamente na forma de código aberto. Foi desenvolvida em Java, precisamente para favorecer a portabilidade. A PassCrackGUI integra ainda algumas comodidades adicionais que permitem ao utilizador identificar a função de derivação da representação ou gerar listas de palavras a testar. O estudo que estava definido como objetivo foi conseguido através da tentativa de obter as palavras-passe a partir de representações de bases de dados vazadas para a Internet e também a partir de um questionário. A primeira abordagem permitiu também testar a ferramenta e a interface gráfica desenvolvida. As experiências permitiram constatar que alguns dos maiores fatores a contribuir para a fraqueza do uso das palavras-passe

são a má escolha das credenciais e, do lado dos programadores, a falha em adotar mecanismos mais seguros para as armazenar.

Como trabalho futuro, aponta-se a possibilidade de desenvolver ferramentas de geração de listas de palavras-passe a partir de páginas pessoais ou em redes sociais dos alvos. Para além deste algoritmo, considerou-se o desenvolvimento de uma versão profissional da ferramenta, que poderia ter como principal objetivo o auxilio às autoridades nas investigações criminais.

Abstract

Passwords comprise important pieces of information nowadays. They are on the basis of many access control systems and are often the first, something-you-know factor of authentication mechanisms. They comprise keys to computer systems, confidential information or even physical facilities, and their widespread adoption makes of their discovery one of the main objectives of the initial phase of computer attacks and an interesting research topic. On the one hand, since passwords are sequences of characters with which the input of users have to be compared to, their representations have to be stored in computer systems; on the other, given their sensitive nature, they have to be stored in a secure manner. Rather than the passwords themselves, it is common and preferable to save transformations of these sequences of characters, which should be obtained using functions with stringent properties such as the ones of cryptographically secure hash or encryption functions. There are many known methods available and documented nowadays for such task, scrutinized in the literature and considered secure, though they are not always correctly employed. Obtaining a password from a representation is thus, normally, a computationally unfeasible task. Cracking a password often refers to the procedure of submitting several known passwords (using dictionaries or compendiums) or patterns (using brute force attacks) to the transformation procedure and compare the result with a representation, until a match is obtained, if ever. As such, the security of the mechanism used to obtain the representations is also dependent of how guessable the passwords are.

This dissertation addresses the topics of habits for construction of passwords and tools for cracking them. Several specialized tools for cracking are available nowadays, most of them free or open source, designed for command line interaction only. One of the main contributions of this work comprised the development of a Graphical User Interface (GUI) for several cracking tools (namely Hashcat, John the Ripper and RainbowCrack), congregating their most interesting features in an integrated and meaningful manner. The developed toolkit, named PassCrackGUI, was then used in the cracking attempt of several Databases (DBs) with password representations that leaked to the Internet in 2014 and 2015 with the intention of analyzing how vulnerable they were to the procedure, and also the contemporary habits of people in terms of construction of passwords. Also aiming to better study the topic mentioned in last, a questionnaire was prepared and delivered to 64 participants. This analysis of password habits constitutes another contribution of this work.

PassCrackGUI is a main output of this Master of Science (M.Sc.) program. It is fully functional, easy to use and made freely available as an open-source project. It was written in Java and tested in Linux, Windows and Mac Operating Systems (OSs). When using it to crack the leaked DBs, it was possible to recover 36% of the 4233 password representations using only dictionaries and simple rules on a common laptop. Part of the problem lies in the adopted mechanisms

for obtaining the representations, which were outdated in most of the cases; while very weak passwords also contributed for this number (e.g., a significant number of 4 digits long passwords was found in one of the DBs). The results from the survey corroborate other works in the area, namely in terms of stereotypes. For example, the answers suggest that men use longer and more diverse (in terms of character sets) passwords than women. Nonetheless, several contracting aspects lead to the conclusion that the participants may be claiming to construct stronger passwords than they really use.

Keywords

Cracking, Crunch, Hash Function, Hashcat, HMAC,Hash Based Message Authentication Code, John the Ripper, Interface, PassCrackGUI, Passwords, PBKDF,Password-Based Key Derivation Function, RainbowCrack

Contents

List of Figures

List of Tables

Acronyms

AAA Authentication Authorization and Accounting

ACM Association for Computing Machinery

AFP Apple Filing Protocol

CEO Chief executive officer

CLI Client Line Interface

CPU Central Processing Unit

CVS Concurrent Versions System

DB Database

DES Data Encryption Standard

DOS Disk Operating System

FIDO Fast IDentity Online

FMS Fluhrer, Mantin and Shamir

FTP File Transfer Protocol

GPU Graphics Processing Unit

GUI Graphical User Interface

HMAC Hash Based Message Authentication Code

HTTP Hyper Text Transfer Protocol

HTTPS Secure Hyper Text Transfer Protocol

ICQ I Seek You

IDE Integraged Development Environment

IMAP Internet Message Access Protocol

IRC Internet Relay Chat

JDK Java Development Kit

JTR John the Ripper

LDAP Lightweight Directory Access Protocol

MAC Message Authentication Code

MD4 Message Digest 4

MD5 Message Digest 5

MIT Massachusetts Institute of Technology

MS-SQL Microsoft SQL

M.Sc. Master of Science

NCP Network Control Protocol

NFS Network File System

NIST National Institute for Standards and Technology

NNTP Network News Transfer Protocol

OS Operating System

PACK Password Analysis and Cracking Kit

PBKDF Password-Based Key Derivation Function

PBKDF2 Password-Based Key Derivation Function 2

PC Personal Computer

PHC Password Hash Competition

PIN Personal Identification Number

POP3 Post Office Protocol 3

PTW Pychkine-Tews-Weinmann

RDP Remote Desktop Protocol

Rlogin Remote Login

RSH Remote Shell

SAM Security Account Manager

SAP Session Announcement Protocol

SHA Secure Hash Algorithm

SHA-1 Secure Hash Algorithm 1

SHA-3 Secure Hash Algorithm 3

SIP Session Initiation Protocol

SMB Server Message Block

SMTP Simple Mail Transfer Protocol

SNMP Simple Network Management Protocol

SOCKS5 Socket Secure 5

SQL Structured Query Language

SSH Secure SHell

UBI Universidade da Beira Interior

USB Universal Serial Bus

VNC Virtual Network Computing

XMPP Extensible Messaging and Presence Protocol

XSS Cross-Site Scripting

WEP Wired Equivalent Privacy

WPA-PSK Wi-Fi Protected Access Pre-shared Key

Abbreviations

AUTH Authentication

VRFY Verify

Chapter 1

Introduction

This dissertation was written in the scope of a master's degree program in Computer Science and Engineering from the University of Beira Interior. This first chapter motivates its subject and delimits its scope. It also presents the problem statement and objectives, as well as the main phases in which this project was divided and the resulting contributions.

1.1 Motivation and Scope

The combination of a username and a password is the most widely used method for access control and authentication in computer systems nowadays. Its widespread adoption is mostly explained by the fact that methods built upon this combination are relatively easy to implement and actually use, requiring only input coming from the keyboard. When they were created, back in the 1960s, their text-based nature was suitable for the purposes of authenticating to the systems of that time and, due to the hardware limitations, the associated mechanism was secure for a password with a suitable length (e.g., more than 7 or 8 characters). Unfortunately, the passwords, and their associated mechanisms should not have lived as long as they did, as even their creator, Fernando Corbato [Ead14], admitted recently, since they did not aged as well as technology evolved. This problem was aggravated by the widespread adoption of password based mechanisms for authentication in Internet sites in the 1900s and 2000s. Nowadays, there a struggle to reinforce the authentication mechanisms with additional factors, as a result of several breaches of data and successful compromises of accounts, only to prove that passwords are perhaps not suitable anymore or need more attention from their holders.

Since they comprise critical pieces of information, passwords need to be securely stored in systems. Given the way of functioning of the authentication or access control mechanism, they need to be known by the user and stored on the target system, at least in an equivalent (secure) form. There are examples of systems that still store this information in plaintext [Hac16] but, normally, safe representations of passwords are produced by cryptographically secure key derivation functions or encryption algorithms before storage. Along with the development of the techniques to store passwords, several tools to crack them were also implemented. Since the key derivation functions are one way functions, these cracking tools adopt the approach of trial and error, by repeatedly calculating representations of known or enumerated inputs through those algorithms until a collision is found. Since humans tend to reuse passwords, use combinations of words with numbers, etc., these techniques are actually quite effective. As

such, part of the problem is in the way these passwords are constructed, while the other may be on the way these passwords are stored or in the security of the system that holds them. Because of their role in computer systems, they are typically a tempting target for attackers, motivating this and other works on this specific area.

The main area of knowledge of this master's program is Computer Science and Engineering. This work is focused on studying habits in terms of construction of passwords and on the development of tools for cracking them. Its scope is confined to the computer security field, namely to topics concerning user authentication, access control and system security. Under the 2012 version of the Association for Computing Machinery (ACM) Computing Classification System, a de facto standard for computer science, the scope of the master's program, reflected in this dissertation, falls within the categories named:

- **Security and privacy∼Hash functions and message authentication codes**
- **Security and privacy∼Cryptanalysis and other attacks**
- *Security and privacy∼Authentication*
- *Security and privacy∼Operating systems security*
- Security and privacy∼Key management

1.2 Problem Statement and Objectives

One of the main problems addressed by this work is the lack of means to interact with password cracking tools in a user-friendly and integrated manner. There are several highly sophisticated tools for password cracking nowadays, most of them offering only a Client Line Interface (CLI). Graphical User Interface (GUI) for these tools are usually developed as separate projects, with most of them focusing one of them only. The CLI and some of the GUI are difficult to use, typically because they are developed within a specialized community, and do not take advantage of features provided by other related utilities or scripts. On the other hand, it is possible to find scripts for automatizing cracking tools or congregating some of their functionalities, but not in a graphical manner.

Another problem addressed in this work is the potential lack of quality in terms of construction of passwords and the easiness with which entire databases with such information leak to the Internet. This is related also with a lack of awareness in the computer security field.

The objectives of this master's program are as follows:

1. To study freely available password cracking tools and related utilities and construct a

toolkit with a GUI. This toolkit should congregate features of some of the best tools (e.g., by using them in background) in an integrated front-end, that is simultaneously user-friendly and complete;

2. Study the habits in terms of construction of passwords. This should be performed resorting to user-surveys and to the analyzes of real databases containing passwords that leaked to the Internet from Portuguese sites, taking advantage of the developed toolkit and taking advantage of the momentum created by its development.

1.3 Adopted Approach for Solving the Problem

In order to achieve the aforementioned objectives, the work described herein was divided into five main parts, briefly described as follows:

1. The 1st part consisted in getting acquainted with the problem and with the notions involving this master's program. It also included reading specialized literature on the state-of-the-art, analyzing a similar work on password habits and related projects. This part also included exchanging some thought with other persons involved in this project or working on this area. During this phase, it was decided that a survey with the intuit to study password construction habits was to be developed along this work;

2. The 2nd part was focused on researching freely available passwords cracking tools and selecting the most suitable ones for the construction of the GUI. This phase also included a broader study of the identified tools, namely John the Ripper, Hashcat, Crunch, RainbowCrack and Hash-identifier, which were later integrated in the toolkit;

3. The toolkit was developed in the 3rd part, following the research of the previous phase. This part comprised also the initial testing and recursive fine-tunning of the GUI. In order to gather some feedback about the interface and functionalities provided, a quick prototype was presented to people in the laboratory to gather some feedback before producing the final version, described herein. The several tests performed at the final stage of this phase enabled confirming that all functionalities were working.

4. The 4th part comprised testing the tool with leaked databases containing passwords and analyzing the results of the experiments. This phase also included the delivery and compilation of results of a questionnaire on password habits.

5. The final part of this work comprised the writing of the dissertation and of two conference papers.

1.4 Main Contributions

The main contributions resulting from this master's program are twofold:

1. The first contribution concerns the development and delivery of a fully functional GUI for contemporary mainstream cracking tools, such as Hashcat, John the Ripper and RainbowCrack. To the best of the knowledge of the author, at the time of writing this dissertation, there was no other GUI congregating the same set of tools and offering the same amount of features. The tool was developed in Java, tested in different Operating Systems (OSs) and its source code is released under an Massachusetts Institute of Technology (MIT) license. It can be downloaded from the github repository via `https://github.com/RicardoXPSantos/PassCrackGUI`. Apart from the interaction with the aforementioned tools, it includes additional features to fine-tune the cracking commands in a console-like interface and facilities to generate patterns or to identify the type of hash in a given input. The GUI is described in chapter 3 and it is the main subject of the paper *PassCrackGUI -- A Graphical User Interface for Password Cracking Tools*, presented and included in the proceedings of the *10th Conference on Telecommunications* (ConfTele2015), held in Aveiro, Portugal, in September 17 and 18, 2015.

2. The second contribution comprised the study of the habits, in terms of construction of passwords, of Portuguese users. This study included the tentative to crack passwords contained in databases that leaked onto the Internet, and also the setting up, delivery and processing of the results of a questionnaire on that subject. The main conclusions of this study are included in chapter 4 and were the main subject of the paper titled *Analysis of Password Habits and Leaked Databases*, presented and included in the proceedings of the *7º Simpósio de Informática* (INForum 2015), which took place in Covilhã, Portugal, in September 7 and 9, 2015.

1.5 Dissertation Overview

This dissertation is divided into 5 main chapters and an appendix. Their content is succinctly described below:

- Chapter 1 -- **Introduction** -- motivates the subject at hands, delimits the scope of this work, identifies the problem to be addressed and enunciates the objectives to be pursued. It also presents its main contributions and the adopted approach for achieving those objectives;

- Chapter 2 -- **Related Work and Background** -- lists and briefly discusses related works on the topic of password habits. It elaborates on the techniques that are typically used to store passwords on computer systems, to then explores the subject of *password cracking*. This is done by briefly describing specialized tools for online or offline password cracking,

as well as other relevant tools within this context;

- Chapter 3 -- **Integrated Front-End for Cracking Tools** -- is focused on the explanation of the most important aspects of the GUI developed within the scope of this work. The contents of this chapter include the identification of the main requirements, use case and activity diagrams, as well as the explanation of the PassCrackGUI interface. One of the sections addresses the supported cracking tools, while others explain the features that enable their simpler usage. Part of the features are described in a format similar to a user manual, towards the end of the chapter;

- Chapter 4 -- **Tests and Password Habits** -- contains the results of several studies performed along this master's programme to understand the habits of Portuguese users when it comes to devising passwords. It also describes the datasets used for some of those analyses and characterizes the population of end-users to which a questionnaire was delivered;

- Chapter 5 -- **Conclusions and Future Work** -- contains the main conclusions of this dissertation, and possible future research and work directions;

- Appendix A -- **Survey on Password Habits** -- contains the set of questions that compose the survey discussed in chapter 4. The questions are in Portuguese because the intention of the survey was to help analyzing password habits of Portuguese users.

Chapter 2

Related Work and Background

2.1 Introduction

This chapter discusses important concepts of the specific field in which this work is included. Section 2.2 discusses how passwords are typically stored in computer systems, security wise, while Section 2.3 provides an opposite perspective by talking about password cracking. Towards the end (in Section 2.4), several tools for password cracking are presented. A table on important leaks of databases containing password representations since 2012 is also included herein.

2.2 Storage of Password Representations in Computers

The combination of a username (or an equivalent identifier) with a passwords is a widely used form of authentication and access control in computer systems. At the time of its creation, the password, which was purely text based, comprised a perfect something you know factor for the aforementioned objective, since it was easy to use and implement in the available systems. Eventually, not even its inventor was expecting that methods based on passwords would make it so far, in spite of its advantages: it is simultaneously easy to use and the underlying methods are easy to implement.

Due to its nature, the usage of passwords for authentication requires this critical piece of information, an equivalent representation or some means to verify its value, to be stored in the authenticator system. Apart from that, it should only exist in the mind of the human using it. This immediately brings up three different problems: (i) if it is in the mind of a human, it will probably make some sense, otherwise he would forget it more easily; (ii) if it is stored in the authenticator, it needs to be done in a safe manner; (iii) even if only a representation is stored in the authenticator, the original form of the password was probably known by this system during registration.

By now, it is common sense that passwords should not be stored in plaintext, though leaks of files and Databases (DBs) with passwords in that form happen once in a while. Normally, what is stored is a representation obtained via a one-way function. Ideally, this one-way function possesses the properties of a cryptographic hashing algorithm, such as collision resistance. By doing so, one is assuring that two passwords will result in two potentially different representations and that the compromise of a representation does not mean an immediate compromise of the

password. The obtained password representation may not be reversible but, for the purpose of authentication, this does not pose a problem, since the verification procedure can always calculate the representation starting from the password, and then compare the two values. It normally never needs to calculate the password starting from the representation.

Passwords comprise often the entry points to computer systems or services, and their importance has been thus motivating the development of many tools to discover them starting from the knowledge of their representation on the target, or stating from no knowledge at all, apart from the kind of feedback of the authentication mechanism to successful or unsuccessful attempts. This procedure is known in the computer security area as *password cracking*. When this dissertation was written, there were many password cracking tools available, most of them are free to download and open-source. Since the algorithms used to protect the passwords are computationally hard to invert, cracking is normally done via brute-force or resorting to dictionary in the same direction used to construct the representation. In other terms, several representations are calculated for potential passwords until one match is found.

Using only one-way functions over functions is not sufficient to protect the passwords, since many started calculating and storing representations in large associative maps, aiming to later explore the tradeoff between efficiency and storage. In order to avoid pre-computation of password representations, most mechanisms now combine passwords with salts before hashing. Salts are random values generated transparently to the user, and stored in plaintext with the representations. Their size and randomness determines the security of the procedure also. Additionally, slow algorithms or added rounds are used to make the calculation slower and cumbersome to repeat [BBBB10, Cod10], which effectively renders pre-computation useless. Most tools are now used against leaked representations and after the salt is known (no pre-computed table is available).

The weakest link became the strength of the passwords, or how easy they are to guess, or even if they were cataloged in the past. Many registration procedures try to address this problem during password generation. For example, many on-line platforms or sites, try to help the user create his password with suggestions and rules (e.g., forcing to use several types of characters, like lower and upper-case, and special symbols [Che12]). This idea is becoming mainstream, along with multi-factor authentication, but it may become cumbersome to users. Jason Hong described a bad experience regarding this matter in an article published in the Association for Computing Machinery (ACM) magazine [Hon15]. According to him, creating a password for the United States of America government website was a real headache. The site was asking for a word with at least eight characters, at least one upper-case, one lower-case, one special character and a digit. There could not be alphabetic sequences in the password and no character could be repeated more than twice. The password had to be changed every 60 day, and could not be equal to any of the previous 10 passwords. Even when he tried to generate the password

randomly, it faced failure in some criteria.

Other interesting approaches have been proposed more recently. Kamouflage [BBBB10] is one of those approaches. The system proposes the usage of honeywords (see next subsection), a concept similar to that of honeypots but for passwords. The idea is to have fake passwords in the DBs so that the attacker is confused about which one is the true. These words cannot be random, because the attacker would most certainly distinguish them from the true ones, which really reflects one of the problems of passwords.

2.2.1 Hash Function and Key Derivation Functions

Although the need to create strong passwords is critical, it is also constantly necessary to develop and apply suitable methods to store this data on information systems. Additionally, it is also important to perhaps adopt other (stronger) authentication methods, e.g., based in public key cryptography. In the meanwhile, the usage of multi-factor authentication may help handling the problem.

As mentioned before, storing passwords in information systems is typically done resorting to one-way functions such as pure hashing functions or the so called key derivation functions. Sometimes, one may also find mechanisms that encrypt the password or that use this password to encrypt a fixed string (e.g., Windows used this method in the past). One characteristic that is typically crucial for these functions is that they need to be slow so as to make live or pre-computation of representations harder. In the area, algorithms such as `bcrypt` and `scrypt` are considered good options for that purpose [D1ß, Cod10], because they are slow and the number of iterations can be adjusted to make them more slow or occupy more memory. Interestingly, `scrypt` enables adjusting not only the processing time but also the memory footprint, which is important when considering Graphics Processing Unit (GPU) optimization for cracking attempts.

The functions used in this context need to exhibit a few peculiar properties: (i) they need to be one-way; (ii) collision resistant; and (iii) slow. These properties are typical of cryptographic hashing functions. Thus, it is possible to find several systems or applications storing passwords using such functions directly, or elaborated procedures that integrate them. Typical functions used in this context are Message Digest 5 (MD5) and Secure Hash Algorithm (SHA)-1, which output 128 and 160 bits, respectively. Other options are provided by the SHA-2 family of functions, from which it is possible to use SHA-384 and SHA-512 (outputting 384 bits and 512 bit hashes, respectively). It is also common to use key derivation constructions such as the Password-Based Key Derivation Function 2 (PBKDF2) to produce the representations or Hash Based Message Authentication Codes (HMACs), calculated with one of the previously mentioned hash functions. More recently, the family of hash functions SHA-3 was standardized by National Institute for Standards and Technology (NIST) after a competition, but its adoption is still ongoing. It is possible to adjust the algorithms to output 224, 256, 384 and 512 bit digests. Notice that many

of the methods used to store passwords are actually primarly used for other objectives, namely for calculating digital signatures and Message Authentication Codes (MACs).

The importance of this specific subject is such that a competition for finding new methods for improving the state-of-the-art in terms of these algorithms has been created. It is named Password Hash Competition (PHC) [ph15], from which Argon has resulted in July, 2015 [Ale15]. The three main topics inspiring this competition was the usage of weak protection mechanisms in the Internet, few number of choices and the need to keep advancing the state-of-the-art.

Nonetheless, the usage of secure password storage methods would be avoided altogether if different authentication means were used. In December 2014, Phillip Dunkelberger, Chief executive officer (CEO) and President of Nok Nok Labs [Lab15], gave a talk about the efforts within the FIDO Alliance (Fast IDentity Online (FIDO) Alliance is non-profit organization with over 150 companies involved) [All15]. He argues that simple characteristics or objects we use, like fingerprints sensors, web cameras, speak at the microphone or even the heartbeat, can be use for authentication. However, he also added that FIDO is looking to keep it simple, like a Universal Serial Bus (USB) key for example. He further stated that, if within the two-factor authentication context, the *Something you Have* is pretty match cover, the *Something you know* factor could be again reduced to something simple like a secure Personal Identification Number (PIN) [Rap14].

2.2.2 Honeywords

The term honeyword was first introduced by Bojinov et al. [BBBB10]. It is defined as a set of fake words that are stored with user records, namely with the true password representations of users. These words are generated via suitable means to be similar to real passwords and trick attackers that compromise the system into thinking that they are the true ones. Before being stored, the same key derivation or hash function is applied to these words also.

The novelty of this system is that leaks involving the honeywords (and potentially the passwords also) are detectable due to the following. If an attacker is successful in exfiltrating the DB, he will end up with a set containing legitimate and fake passwords. After cracking some of them, and depending on the ratio between the two, he may probably end up trying to log into the system with one of the honeywords, which should issue an alert and be subject to further processing, as elaborated by Juels and Rivest [JR13].

2.2.3 Storage of Passwords in Windows, Linux and Unix Systems

The typical input of cracking tools is the password representation value and eventually the salt. It is often possible to fed them with lists comprised of such entries, one per line. To provide this explanation with a practical example on where to find this information in OSs, this section contains a brief description on how some versions of Windows, Linux and Unix systems store representations of passwords.

In the case of older versions of Windows, the accounts information is usually stored in a Security Account Manager (SAM) file, which is stored in the `%SystemRoot%/system32/config/SAM` directory of OSs from the Windows family, starting from version NT [Gri02]. The file contains entries with a format similar to

```
username:ID:LM hash:NTLM hash::: ,
```

where the `username` is the account name of a given user, the `ID` is an identifier for the account, and the remaining two fields are hash values for the password [Pie08]. These values are obtained through proprietary methods, but one involves using the password to encrypt a fixed string using Data Encryption Standard (DES), while the other uses Message Digest 4 (MD4). These methods were used in Windows NT, 2000 and XP.

In the case of Linux or Unix-like OSs, the storage of password related information is made in the `\etc\shadow` file, while the account information *per se* is stored in the `\etc\passwd` file. Each account is represented by a line in these files, each line with a format similar to the following:

```
login name:encrypted password:date of last password change:
minimum password age:maximum password age:password warning period:
password inactivity period:account expiration date:reserved field
```

The value representing the password in the file is obtained using one of several potential hashing or encryption functions. The format of the representation is given by `$id$salt$encrypted`, where the `id` refers to the algorithm used. In modern systems, it is typically equal to 5 or 6, meaning that SHA-256 or SHA-512 were used, respectively. An example of such field is included next:

`$6$3WDiHsU4$5pjHyGZibmC61Y1W7XnwZitG1/5WU1kNbgZuwqQ23yn0cvTP8a5ASRwp/Khmu6yrDbNmKcFKfPRd49tDsOJ781`

2.3 Leakage and Cracking of Passwords in the Past

As reported earlier in this chapter, passwords are not normally stored in clear text, but a secure representation instead. The password cracking is the process of discovering a clear text of a given representation via trial and error [Mar08]. Typically, the process involves calculating the representation of several potential candidates using the same hashing algorithm until a match is found. Notice that a given clear text does not necessary need to be the password; it suffices for its representation (according to the hashing function) to be the same. If cryptographic hash functions are correctly being used, the clear text is most probably the password. Nonetheless, if the hashing procedure does not use all characters of the password to calculate the representation, several different collisions may be found. Either way, finding the true password

or a collision will grant access to the attacker, since authentication is normally performed by comparing the representations, and not the passwords.

Because of their way of functioning, cracking tools need to be very optimized and fast. In order to fully take advantage of the available technology, many of them support multi-threading processing not only over multiple Central Processing Units (CPUs), but over GPUs as well [Van15]. To make them more efficient, it is typically possible to prepare lists of words and patterns *a priori*, which may narrow the search to a comfortable number of options. Unfortunately, nowadays, it is inclusively possible to find these lists on the Internet, and they originate from the leakage of and cracking production DBs containing real password representations. Table 4.1 lists some of the major publicly known leaks since 2012 [MER+15]. When such previously prepared lists are used, the cracking method is using a *dictionary*. If all patterns (up to a given length) are to be tested, then the utilized method is *brute-force* [WAdBMG09]. There is active research in this topic also, namely on methods for constructing wordlists for example based on keyboard configuration and patterns [HAF15, LWH07, SBHM09]. The Markov chains are also widely used in this area [MYLL14], since many times the next key press is dependent on the previous one.

2.4 Tools for Password Cracking

Password cracking is a very popular topic in some computer security communities and has been motiving the development of many tools with that specific purpose. Most of them are based on the same underlying method of trial and error, but many are specific for a given target application or usage of passwords. For example, some of these tools are intended for circumventing or cracking networked systems and protocols with authentication, while others are specialized in local accounts, databases, file types or applications. It is thus normal to categorize these tools as online or offline (password cracking tools). This section is divided into two sections, in accordance with that categorization.

2.4.1 Online Password Cracking Tools

Online password cracking tools are specially designed to circumvent authentication or crack passwords in remote systems. Tools in this category typically support the protocols (or a subset of the associated messages) to communicate with the target systems.

Hydra [Hau14, Sha14] is one of the most powerful and complete online cracking tools available. When writing this dissertation, its most recent version was 8.1, and the amount of supported protocols and applications was significant, effectively giving an idea of the complexity (and motivations behind) of these tools. It supported the following protocols, functions and applications: Asterisk, Apple Filing Protocol (AFP), Cisco Authentication Authorization and Accounting (AAA), Cisco auth, Cisco enable, Concurrent Versions System (CVS), Firebird, File Transfer Protocol (FTP), Hyper Text Transfer Protocol (HTTP) `FORM-GET`, HTTP `FORM-POST`, HTTP `GET`,

HTTP `HEAD` HTTP `PROXY`, Secure Hyper Text Transfer Protocol (HTTPS) variants for all the previous HTTP counterparts, I Seek You (ICQ), Internet Message Access Protocol (IMAP), Internet Relay Chat (IRC), Lightweight Directory Access Protocol (LDAP), Microsoft SQL (MS-SQL), MySQL, Network Control Protocol (NCP), Network News Transfer Protocol (NNTP), Oracle Listener, Oracle SID, PC-Anywhere, Personal Computer (PC) Network File System (NFS), Post Office Protocol 3 (POP3) , PostgreSQL, Remote Desktop Protocol (RDP), Rexec, Remote Login (Rlogin), Remote Shell (RSH), Session Announcement Protocol (SAP), Session Initiation Protocol (SIP), Server Message Block (SMB), Simple Mail Transfer Protocol (SMTP), SMTP Enum, Simple Network Management Protocol (SNMP) (the first 3 versions), Socket Secure 5 (SOCKS5), Secure SHell (SSH), SSH key, Subversion, Teamspeak, Telnet, VMware Authentication, Virtual Network Computing (VNC) and Extensible Messaging and Presence Protocol (XMPP) (it actually supports more than 50 protocols). It supports several login mechanisms for each protocol. Because it is written in Java, this tool is cross-platform and can be run in Linux or Unix-like systems, Mac OS/X and Windows. It is even possible to find the mobile applications of Hydra. The tool is open-source, and can be thus be easily expanded to provide further support for more protocols and applications. It is possible to use Hydra from the command line, but also to use the HydraGTK GUI to interact with this tool.

Medusa [Joe12] constitutes another good example of a fast online password cracking tool. It is specially suitable for brute-forcing a remote authentication system and it resorts to parallelism to achieve faster results. It was developed with the intention to address some gaps of Hydra. As such, it is similar to the previously presented tool in several aspects, but with improved stability, better code organization and the integration multi-threading (it enables, for example, targeting several systems at the same time). Contrarily to the previous tool, a GUI is not available for Medusa, though it also supports a large panoply of protocols such as AFP, CVS, FTP, HTTP, IMAP, MS-SQL, MySQL, NetWare NCP, NNTP, PcAnywhere, POP3, PostgreSQL, Rexec, Rlogin, RSH, SMB, SMTP-Authentication (AUTH), SMTP-Verify (VRFY), SNMP, SSHv2, Subversion, Telnet, VMware Authentication, VNC, Generic Wrapper, and Web Form.

The number of online cracking tools is large [Sha14] and the description of all of them would fall out of the scope of this work. Nonetheless, this section would not be complete without the mention to **Aircrack-NG** [Air15], which is basically a set of tools that allows auditing the security of Wired Equivalent Privacy (WEP) and Wi-Fi Protected Access Pre-shared Key (WPA-PSK). Amongst other features (such as frame sniffing and recording, virtual network cards, traffic injection), it allows the recovering of passwords used in Wi-Fi networks, sometimes via packet injection (for reseting associations) and sometimes via exploitation of the security design flaws of the protocols or mechanisms. As such, its ways of functioning is a little different from other cracking tools, hence the value of mentioning it, apart from its popularity. There is a GUI for this set of tools, though the CLI is also very well implemented and easy to use. Aircrack-ng is open-

source and can be run in Linux or Unix-like systems. There are binaries for Windows also. It is normally utilized from the command line and requires some effort from the user to understand its functioning, even when the GUI is used. This happens because of the steps involved, which require setting up the network card in promiscuous mode, collect traffic and execute attacks.

2.4.2 Offline Passwords Cracking Tools

When writing this dissertation, three of the most popular offline password cracking tools were Hashcat, John the Ripper and Rainbowcrack. Starting with **Hashcat** [Has15], it can be said that its development was motivated by the need for a cracking tool with multi-threading capabilities by design. When it was developed, there were already some available tools with parallel processing support, but only as an add-on or via patch (e.g., John the Ripper). The 0.30 version of this program, considered to be its public release with this name, was published in 2003. It was preceded by two projects, one of which known as *attomcrack* (version 0.01) and another one known as *Dr.Hash* (version 0.29). Even though *attomcrack* was already using multi-threading by design, it was a very simple tool supporting only dictionary attacks. At version 0.49 (year of 2015), Hashcat supports brute-force and dictionary attacks, and includes the facilities to take advantage of GPUs (though the GPU has to support the feature), which is an absolute requirement for password cracking.

Along with **Hashcat**, **John the Ripper** [Ope15a] is a very popular offline password cracking tool. Its first stable release was in 1996 and, as such, it came a long way since then. In mid-2015, its most recent free version was 1.8.0. Many of its development iterations made this tool converge to optimized code and, as such, it is considered to be a very fast application. Its name is a reminder of the dark humor of this community, and one of its purposes is to detect weak passwords in Unix systems, though it evolved from a tool for Disk Operating System (DOS). It supports both dictionary or brute-force attacks and it is available in free and paid (so called *Pro*) versions. *Johnny* [Ope15b] is a well organized GUI for John the Ripper, which enables graphical access to many of its features.

Rainbowcrack [Pro15] is another cracking tool, whose popularity is mostly due to the fact of being able to use rainbow tables for finding collisions. It was proposed by Philippe Oechslin in 2003 [Oec03] and, at the time of writing of this dissertation, its most recent version was 1.6.1, It provides functionalities for producing the rainbow tables, sorting and looking up values. Its code is highly optimized to run on 64bit machines and it includes support for several GPUs. The tool is multi platform (it runs on Windows and Linux OSs) and it comes with a GUI.

The three tools described with more detail in this section are the ones supported by the toolkit developed within the scope of this work. Though there are individual GUIs for some of them and their source code is available, the proposed toolkit was designed and developed from scratch.

2.4.3 Other Password Cracking Related Tools

The area of password cracking is very dynamic, with many different tools being developed for all sorts of purposes. For example, it is possible to find scripts, such as *Hash ID* [r..11], whose objective is to list the names of the potential hash or derivation functions used to produce a given password representation. This representation is provided as an input to the script. *Hash ID* was previously developed by *Blackploit* [Bla15a] and integrated in the Kali Linux (1.0.9 version) OS, as one of the utilities for password cracking. Because it comprises a very useful resource in this context, it was decided to integrate it in the toolkit to help users to identify the function that needs to be configured before any cracking attempt.

Another tool that was integrated into the toolkit developed along this work is *Crunch* [bof14]. This small program, written in the C language, can be used to generate wordlists and dictionaries, which may latter be fed to the cracking tools. It accepts parameters such as the minimum and maximum size of the words one wants to generate, the characters and masks. It then generates all possible combinations according to those parameters. This tool was integrated in the GUI and it can be used to create dictionaries in a simple manner.

2.4.4 Available Toolkits

As previously mentioned, some tools have GUIs, but these are custom made for each one of them. Additionally, some of these GUIs are not necessarily user friendly, with their developers assuming that they will be used by experts or technically skilled persons. Nonetheless, within this context, it is possible to point out, for example, the toolkit called Password Analysis and Cracking Kit (PACK) [Pet13]. PACK is a very complete software for analyzing password lists. It does not provide a GUI and does not integrate cracking tools (despite its name), but it can used to generate input files for Hashcat. The set of utilities included in PACK enable studying files containing previously cracked passwords or dictionaries, set up filters for these files, masks, detect patters and other characteristics that can be effectively used to then increase efficiency of Hashcat by providing a more directed and tidy input file with potential passwords or patterns.

2.5 Conclusions

This chapter discusses important concepts surrounding the area of password cracking, and provides some of the background for the work carried in this master's program. It starts from an explanation on how passwords are typically stored in information systems, to then motivate the problem being address through the presentation of several leaks that led to exposed passwords. Unfortunately, these leaks have been feeding other cracking procedures, as many passwords are repeated between services and humans. Towards the end of the chapter, some of the most popular online and offline password cracking tools were described. Some of these tools were integrated or are supported by the toolkit described in the following chapter.

The discussion showed that some tools can only be used via a CLI, while others have their own GUI. To the best of the knowledge of the author, there is no other GUI facing up the same three cracking utilities selected in this work. Furthermore, cross-platform seems to be an attractive characteristic of GUIs, since the tools can also be often used in different platforms too, specifically Windows and Linux. These guidelines were taken into consideration during the next phases of this master's program, as reported in the following chapter. For example, the developed toolkit was written in Java and tested in different OSs.

Entity	Year	Data Lost (number of records)
Anthem	2015	8 0000 000
JP Morgan Chase	2014	7 6000 000
Gmail	2014	5 000 000
Home Depot	2014	56 000 000
Mozilla	2014	76 000
Community Health Services	2014	4 500 000
New York Taxis	2014	52 000
Dominios Pizzas (France)	2014	600 000
MacRumours.com	2014	860 000
LexisNexis	2014	1 000 000
AOL	2014	2 400 000
Korea Credit Bureau	2014	20 000 000
Target	2014	70 000 000
Ebay	2014	145 000 000
Adobe	2014	152 000 000
Neiman Marcus	2014	1 100 000
Advocate Medical Group	2013	4 000 000
SnapChat	2013	4 700 000
Florida Courts	2013	100 000
Florida Department of Juvenile Justice	2013	100 000
Central Hudson Gas & Electric	2013	110 000
Kirkwood Community College	2013	125 000
Citigroup	2013	150 000
Washington State court system	2013	160 000
Nintendo	2013	240 000
Twitter	2013	250 000
Apple	2013	275 000
Scribd	2013	500 000
Drupal	2013	1 000 000
Dun & Bradstreet	2013	1 000 000
Kroll Background America	2013	1 000 000
Kissinger Cables	2013	1 700 000
Ubuntu	2013	2 000 000
Vodafone	2013	2 000 000
Facebook	2013	6 000 000
Yahoo Japan	2013	22 000 000
Evernote	2013	50 000 000
Living Social	2013	50 000 000
Emory Healthcare	2012	15 000
Formspring	2012	420 000
Yahoo Voices	2012	450 000
Medicaid	2012	780 000
California Department of Child Support Services	2012	800 000
New York State Electric & Gas	2012	1 800 000
Three Iranian banks	2012	3 000 000
South Carolina Government	2012	6 400 000
Office of the Texas Attorney General	2012	650 0000
Global Payments	2012	7 000 000
Gamigo	2012	8 000 000
LinkedIn, eHarmony, Last.fm	2012	8 000 000
KT Corp.	2012	8 700 000
Greek government	2012	9 000 000
Zappos	2012	2 4000 000
Massive American business hack	2012	160 000 000
Blizzard	2012	14 000 000

Table 2.1: Known leaked password databases since 2012 [MER+15].

Chapter 3

Integrated Front-End for Cracking Tools

3.1 Introduction

This chapter describes the toolkit developed within the scope of the research presented in this dissertation. Section 3.2 elaborates on the software engineering elements of the user interface, as well as on the software dependencies and requirements of PassCrackGUI. The functionality of the toolkit in terms of cracking and features is detailed in Section 3.3, while an utlization guide and manual is included in Section 3.4. The main takeaways of the chapter are enumerated in Section 3.5.

3.2 The Integrated Front-End

PassCrackGUI is a frontend GUI for well-known cracking tools. The user interface and the supporting logic was developed in Java using the Integraged Development Environment (IDE) NetBeans v8.0.2. Figure 3.1 contains a screenshot of the second iteration of the interface, whose layout was inspired in other tools [Com09, Ope13, Ora14a, Ora15a, Ora14b]. This improved iteration undergone several modifications addressing design and structural flaws from the previous version. Recall that one of the objectives of PassCrackGUI is to be user-friendly, allowing its use by a wide range of users with different skill sets. The development of the interface had this in mind, keeping the interface itself simple and organizing features in a straightforward way. The improved version of the interface also introduced a newer and friendly color map. The choice of grayscale and black and white mirrors the spirit of penetration testing.

The layout is organized in three major vertical parts. The user workflow is oriented from the left to the right, which comes as natural as reading. This means that a password cracking attempt will start with the selection of options in the left most pane, evolving to the central one and finalizing at the right, with the fine-tunning of commands and exhibition of results in consoles environments.

3.2.1 Dependencies and Requirements

The toolkit was developed in Java for very specific password cracking tools. In order to be fully functional, those tools must be readily available in the system running PassCrackGUI. The external dependencies and requirements of PassCrackGUI can be summarized as follows:

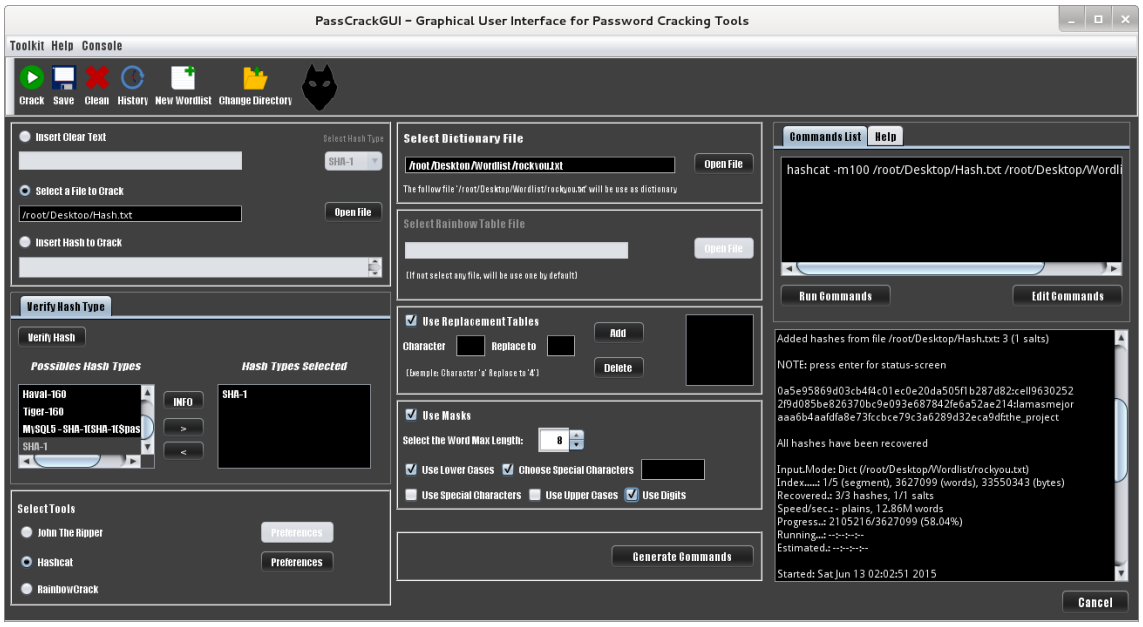

Figure 3.1: Screenshot of the GUI of PassCrackGUI. The interface requires a minimum resolution of 1320x665 pixels.

- **Java**: PassCrackGUI was written using version 1.8.0_25 of the Java Development Kit (JDK), which can be downloaded from [Ora15b];
- **Cracking Tools**: the toolkit uses well-known cracking tools in the background. They are John The Ripper [Ope15a], Hashcat [Has15] and RainbowCrack [Pro15];
- **Crunch**: PassCrackGUI additionally uses Crunch [bof14], a worldlist generator written in C programming language and available for download at [bof14];
- **Hash ID**: this script is used for identifying hash types copied into the interface by the user within a given cracking session. It is available for download at [r..11].

Additionally, as hinted in the caption of Figure 3.1, the toolkit uses a fixed sized window, requiring a minimum resolution of 1320x665 pixels.

3.2.2 Use Cases

This subsection presentes two use cases from the software engineering point-of-view, which allows users to understand the workflow as it runs. The first Use Case, which is included in Figure 3.2, illustrates the main scenario where the user interacts the main features of PassCrackGUI. Summarily, the toolkit is able to either crack passwords or generate wordlists, using the tools mentioned in the previous subsection. For the former feature, the user is able to select which tools and algorithms he or she wishes to use in the background, as well as provide input data. It is noteworthy to mention that PassCrackGUI puts together the commands passed onto the background tools on the interface itself, allowing the user to change them at will. For the

latter feature, the generated wordlists can be used in the current or future cracking sessions, and the interface provides procedures and dialogs for saving the created lists into a directory of the choice of the user.

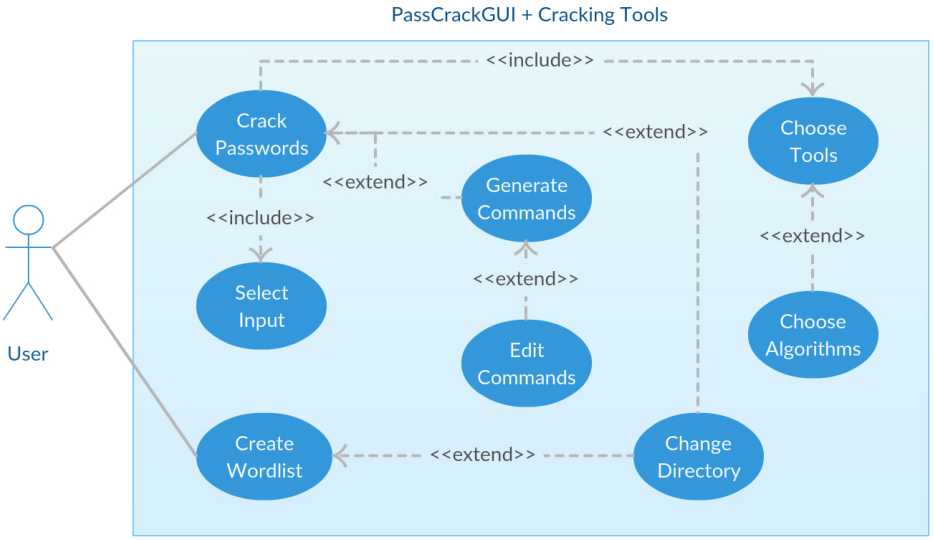

Figure 3.2: Main Use Case diagram for PassCrackGUI.

PassCrackGUI offers multiple choices of input types, meaning that it allows the user to choose which option suits better for a given cracking session. The use case for this feature is illustrated in Figure 3.3, which depicts the choices of inserting cleartext, a single hash value or a file with an arbitrary number of hash values. The first option is useful for testing own passwords, where PassCrackGUI computes the hash value of the cleartext (the user can choose from MD5, Secure Hash Algorithm 1 (SHA-1) or SHA-256) and then instructs the cracking tools to break it. For operations related with hash provided data, PassCrackGUI attempts to find out the underlying algorithm used to generate them using Hash ID.

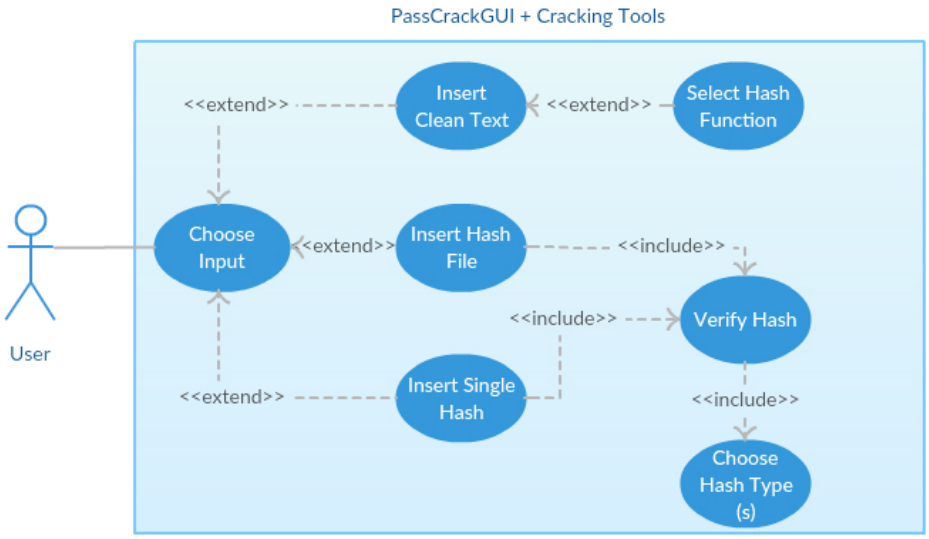

Figure 3.3: *Input option* Use Case diagram.

In terms of cracking capability, the toolkit allows to select a subset or the entire set of tools for execution accordingly with the chosen dictionary or brute-force attacks. For a dictionary attack configuration, the user is prompted to input the required dictionary file and may utilize the so-called replacement tables, which can be used to produce further password combinations starting from the existing ones. On the other hand, brute-force attacks take advantage of permutation masks that allow a more efficient cracking of the search space. Moreover, the toolkit provides a useful feature for specific cracking sessions, where the user is able to select sets of characters.

PassCrackGUI further contemplates a history feature that allows to save the configurations of cracking sessions. This is done explicitly, requiring the user to save and restore the sessions. For example, if the user has the necessity to choose the same tools for multiple attacks in different toolkit utilization, it is possible to save those choices. Saving can be made using `Save` button. Besides that, the three supported cracking tools can be either selected manually and chosen all in simultaneously. After selecting Rainbowcrack, it is also necessary to provide a rainbow table. The other two tools (John the Ripper and Hashcat) will spawn the normal Dictionary and Brute-Force options. If the Dictionary attack is chosen, the user needs to load a dictionary file via the appropriate dialogs. In the case of a brute-force, there is the option to define masks and set the maximum length, which is mandatory.

3.2.3 Activity Diagram

The activity diagram of PassCrackGUI, which is shown in Figure 3.4, complements the use cases explained in the previous subsection by providing a high-level view of the toolkit execution throughout time, in terms of the selection of cracking tools and their options. It emphasizes that the workflow of the toolkit was designed in such a way that it enables a gradual selection of the options, from the most general to the more specific ones.

3.3 Implemented Functionalities

This section elaborates with more detail on the functionality and features of PassCrackGUI. It is divided into two other subsections: *Cracking Features* and *Interface Features*.

3.3.1 Cracking Features

It should be mentioned that the cracking capability of PassCrackGUI is provided by the tools running in the background. The choice of these tools took into account the popularity within the community and easiness of integration into such a toolkit with regard to dictionary and brute-force attacks. Additional features such as the generation of wordlists and hash identification come in handy for cracking sessions. The following discussions go into detail with regard to the aforementioned features.

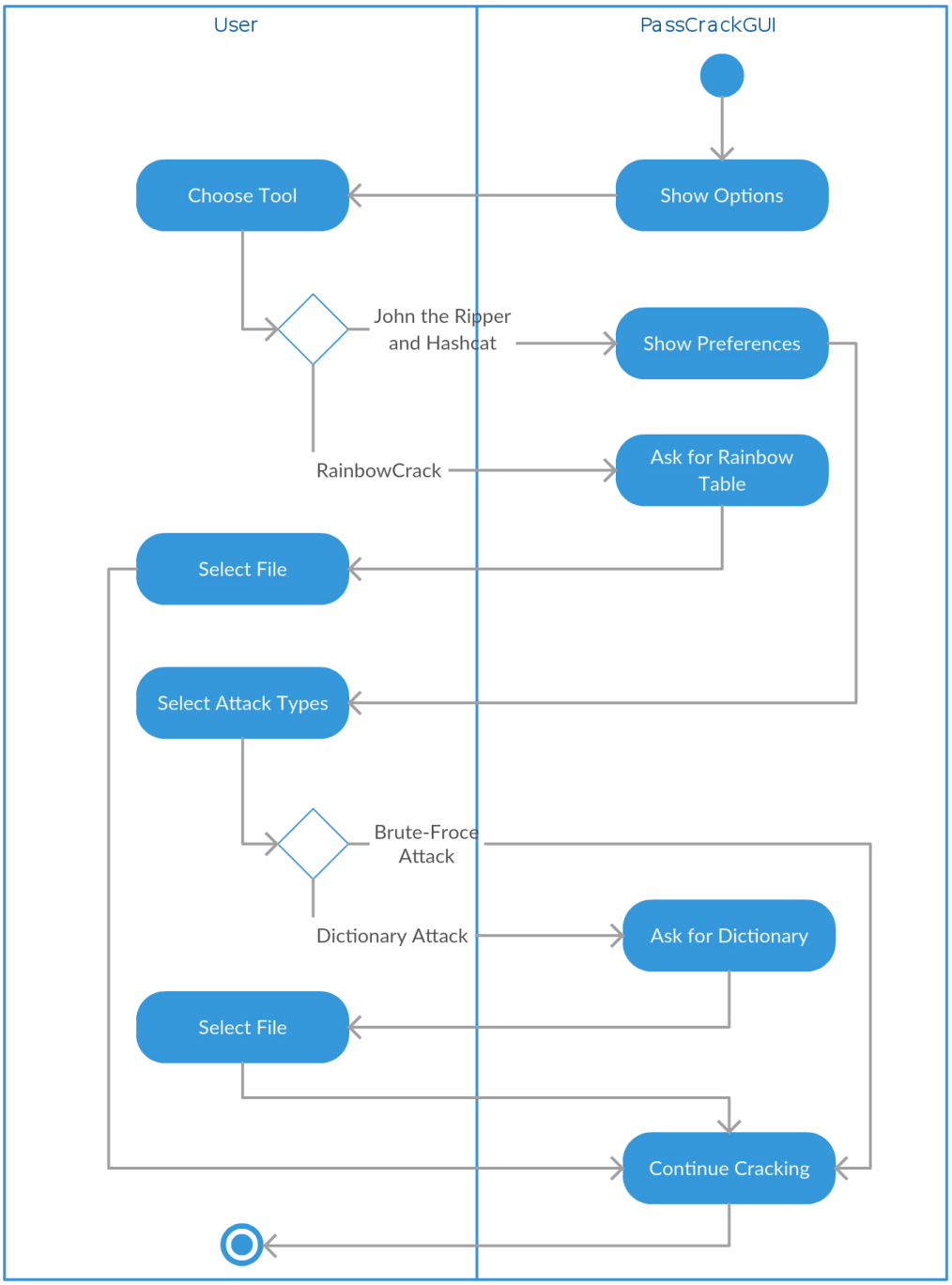

Figure 3.4: Activity diagram for the *Selecting Tools* Use Case.

The integration of *Hash ID* allows for the identification of the hash function of the data inputed to the toolkit. It is strictly mandatory to select at least one key derivation or hash function algorithm for it to run, and this functionality makes that choice simpler, because it outputs the list of the most probable algorithms according with the length and characters of the provided hash. This feature is executed by pressing the button `Verify Hash`. In order to seamlessly include this feature in the GUI, the Python script had to be modified (its output) to cope with the displaying requirements.

Both John the Ripper and Hashcat support dictionary attacks, using pre-computed data files. Such dictionary files serve as shortcuts for cracking as they usually contain common combinations of characters retrieved from real-world systems. In fact, the generation of worldlists takes into consideration several patterns seen in such real-world systems. A successful cracking sessions ends immediately when a cracking tool finds a hash on such a dictionary. The generation of worldlists in PassCrackGUI requires the specification of character sets and minimum and maximum sizes of each string to be generated. This specific featured is powered mostly by Crunch. Hashcat provides a wordlist generation capability that allows to transform and derivate a new wordlist from another one, performing simple substitutions and transformations by precisely taking into consideration the patterns previously mentioned (e.g., swapping letters with similar-looking numbers). This feature is known as replacement tables and is now also available for use with John the Ripper via PassCrackGUI.

Besides the typical brute-force attacks, PassCrackGUI also provides means to use masks for specialized cracking guesses. The usage of masks allows the user to filter out or replace digits, lowercase and uppercase letters, and special characters. Hashcat actually allows to specify which characters to use. PassCrackGUI takes all this into consideration, activating and deactivating selectable options according with the configuration laid out by the user.

The toolkit provides also the so-called *Time-Trade Memory Attack*, which is associated with RainbowCrack. It requires the pre-computation of data files with rainbow tables, or obtaining them with the right format elsewhere. For example, it is possible to download rainbow tables from the Internet, some of them are free, while others are paid.

3.3.2 Interface Features

The toolkit provides useful features in terms of software usability, making use of window dialogs to instruct the user and shortcuts for triggering actions. For example, to ease the routine of selecting files, the GUI provides the possibility of selecting them throughout a graphical wizard. One of those features allows to change the colors of the output consoles, which usually varies according with personal user preference. It is possible to select both the background color and the text color. The installation path of the supported cracking tools (see Section 3.3) can also be configured for the toolkit, namely by editing its main configuration file. The interface provides a graphical way to change these settings also. Another feature allows the user to reset the currently chosen options to their default values. Furthermore, the toolkit is able to save the configuration of a given cracking session. Later on, the user can restore such configuration and re-run cracking sessions. These sessions can be named according to user preference also, but PassCrackGUI automatically saves all of them with a predefined serial name, storing also the name of the attacks, the date and the time of the session. The history can be searched from within the interface.

The output console on the right hand side of the PassCrackGUI interface depicts the commands put together by the toolkit. These commands can be generated or edited by the user with a simple click, allowing modification of the commands without restriction. Such feature may be useful for more experienced users or for those who wish to know the background commands of the cracking tools.

It should be mentioned that the interface adapts dynamically to the inputs of the user. For example, after selecting one of the cracking tools, a different set of choices is displayed in the intermediate pane, favoring user-friendliness. This way, the user does not need to know the options or limitations of the different tools. Several shortcuts (mentioned in the final part of this chapter) can be used to control some aspects of PassCrackGUI so as to improve the workflow.

3.4 Toolkit Installation and Utilization

This section discusses the installation of the toolkit and its usage. The discussion is thus divided into two main subsections.

3.4.1 Installing and Executing the Toolkit

Installing the toolkit includes downloading the package containing the source code, uncompressing and compiling it. Obviously, the dependencies described in Section 3.2.1 should be satisfied a priori. Installing in Windows or Linux requires nonetheless the files to be organized differently. In Windows, all Java files must be included in a folder named `toolkit`, while all other files (`path`, `Hash_ID_v1.1.py`, and `png` files) should be in the folder containing this `toolkit` folder, as depicted in figure 3.5.

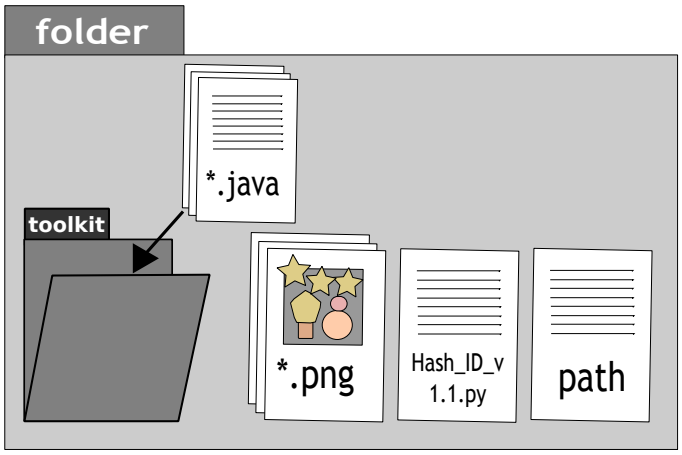

Figure 3.5: Organization of the files for compiling the PassCrackGUI on Windows.

In the case of Linux based OSs, all files must be saved on a folder named `toolkit`, as shown in figure 3.6.

Once the files are organized, the toolkit can be compiled by navigating into `toolkit` folder using

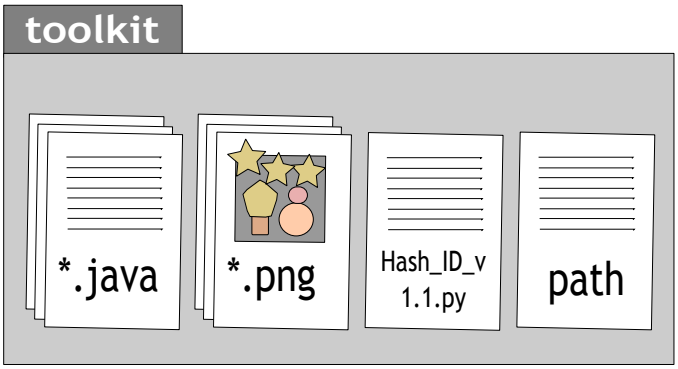

Figure 3.6: Organization of the files for compiling the PassCrackGUI on Linux.

a console or terminal and issuing a command similar to the following one:

```
javac *.java
```

The user must then exit the folder and run the toolkit as shown below:

```
cd ..
java toolkit.Interface
```

3.4.2 Toolkit Utilization

This subsection is meant to describe the utilization of PassCrackGUI, serving as a usage manual for the toolkit. The instructions reviewed herein focus on a few options and complement the discussions made throughout this chapter with regard to the toolkit features, providing illustrations along the way. The utilization guidelines can be summarized as follows:

- **Inserting Cleartext**: The dialog shown for inputting data can be seen in Figure 3.7. This dialog represents the first step in the utilization of the toolkit and allows to select the input data type. The first one allows the insertion of cleartext through a textbox. Unless the user chooses a specific algorithm on the right hand side to compute the correspondent hash value of the input text, the default one is used. The algorithms MD5, SHA-1 and SHA-256 are available. That correspondent hash value is shown below in the window for better clarification.

- **Inserting Hash Values**: PassCrackGUI handles a single or multiple hash values when the user respectively inputs a single hash value or a file to analysis (a file may contain multiple hashes from different hash functions). The former case is illustrated in Figure 3.8. In either case, a dialog to verify the inserted hash type is spawned. In this window, shown in Figure 3.9, the user is presented with two inner windows that allow to select or exclude the cryptographic hash or key derivation functions. At least one algorithm must be selected. This feature is accessed via the `Verify Hash` button.

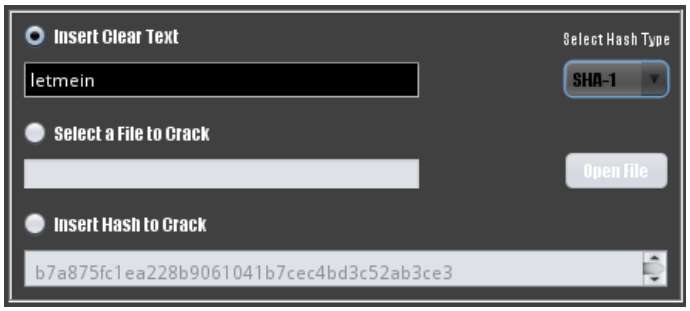

Figure 3.7: The part of PassCrackGUI for clean password insertion.

Figure 3.8: Input of values to crack.

- **Selecting Tools**: After selecting the input, the user should proceed for the configuration of the cracking tools in the inferior left side of the interface. He may choose between John the Ripper, Hashcat and RainbowCrack, as illustrated in Figure 3.10. The selection of any of the two first tool will trigger changes in the middle pane of the interface, which will update to show the cracking options of each of the tools.

  Figures 3.11 and 3.12 respectively show the middle pane as it appears when reviewing the cracking options for John the Ripper and Hashcat. A default attack is chosen for John the Ripper and Hashcat if none is user-specified. E.g., for John the Ripper, an attack defined in the tool itself, which combines different attacks including dictionary, some defined rules and, in last, brute-force 3.11 is executed.

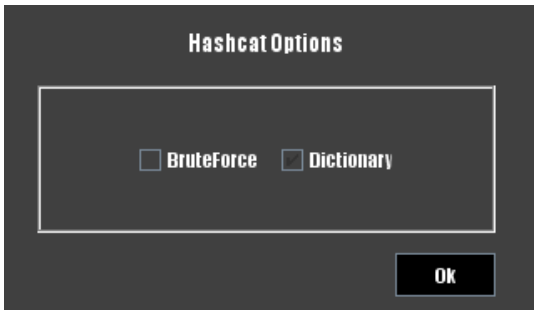

Figure 3.12: Selection of attacks for Hashcat.

There is only one algorithm available for RainbowCrack (rainbow table attack), which is not shown here.

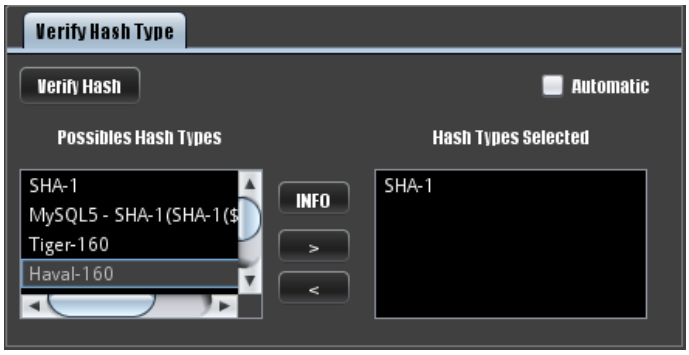

Figure 3.9: Verification of possibles types of hashes.

Figure 3.10: Choice of available tools.

- **Selecting Dictionaries**: If dictionary attacks are selected, the user is prompted to input one dictionary file. If none is specified in the dialog shown in Figure 3.13, PassCrackGUI uses a default one (the one provided with the free version of John the Ripper).

Figure 3.13: Choosing a dictionary file dialog.

- **Selecting Rainbow Tables**: For a RainbowCrack session, a rainbow table file must be inputted in a dialog similar to the one of selecting a dictionary file, as shown in Figure 3.14.

Figure 3.14: Choosing rainbow table.

- **Using Replacement Tables**: The use of replacement tables is associated with dictionary attacks. The configuration of such tables is done by means of the interface shown in Figure 3.15. A summary of the rules of the replacement table is shown on the right part of the window. The interface enables adding and removing simple rules in an intuitive manner.

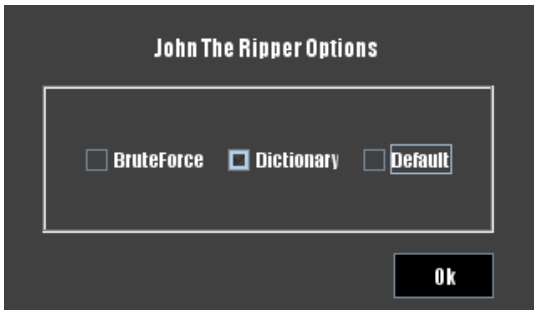

Figure 3.11: Selection of attacks for John the Ripper.

Figure 3.15: Setting up replacement tables interface.

- **Using Masks**: Using masks in brute-force sessions triggers the dialog in Figure 3.16. Since brute-force attacks may take a long time to run, this feature was inserting to keep the amount of possible combinations manageable. The *Chose Special Characters* option is only available for the brute-force attack of the Hashcat tool.

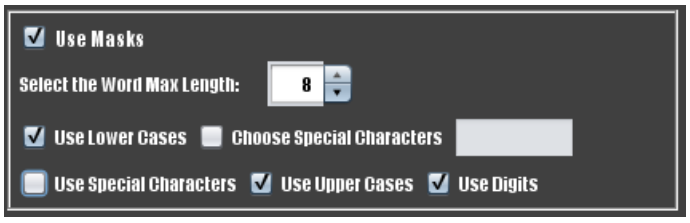

Figure 3.16: Utilization of masks in brute-force attacks.

- **Managing Commands**: As previously mentioned, PassCrackGUI puts together the commands passed onto the background tools and makes them editable to the user. For that end, the user must generate the commands through the `Generate Commands` button. Afterwards, editing commands is done in the window shown in Figure 3.17, after pressing `Edit Commands`, directly in the provided console. The warning seen in Figure 3.18 is shown to users in order to make them aware of the fact that editing commands may end up in failed executions, unless they know what they are doing.

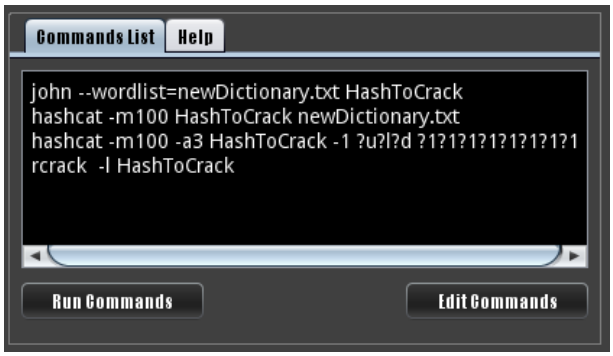

Figure 3.17: List of commands generated in a console environment.

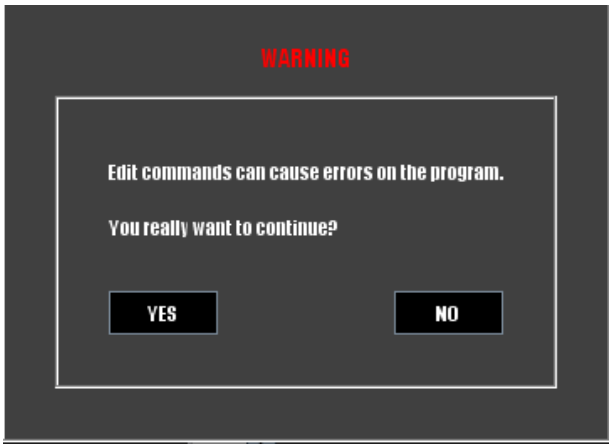

Figure 3.18: Warning seen after pressing the `Edit Commands` button on PassCrackGUI.

- **Console Output**: A sample of the console output can be observed in Figure 3.19. This console environment component is included in the right most pane of PassCrackGUI. It is always visible, and it enables obtaining the outputs of the tools as they were running in the CLI. Nonetheless, some logic was added so that the final cracking values are summarized at the end of the execution of the command, which was not provided by the tools.

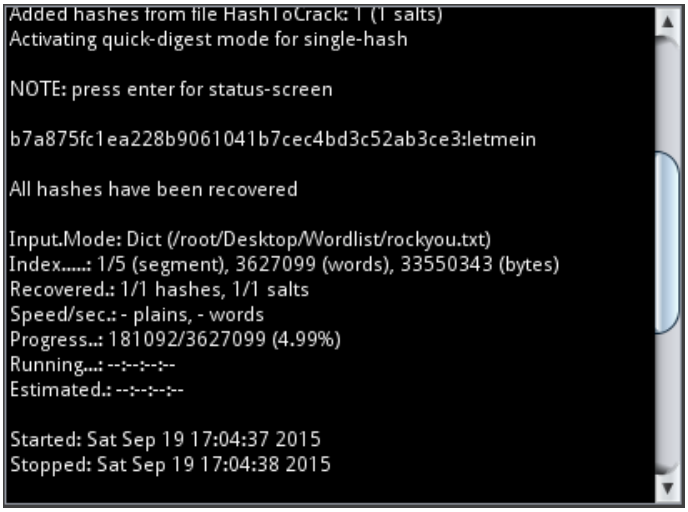

Figure 3.19: Showing outputs of cracking attempts.

- **Toolbar**: PassCrackGUI depicts a simple toolbar with buttons to organize some of the features it provides. The toolbar is shown in Figure 3.20. The options shown there are associated with, and provide a quick means to access, the main features of the interface. For example, the stored records of previous sessions can be accessed via the `History` button and returning to the main cracking interface is done via `Crack`.

Figure 3.20: PassCrackGUI main toolbar: fast access to `Crack`, `Save`, `Clean`, `History`, `Wordlist` and `Changing Directories`.

- **Changing Directory**: The path of the dependencies can be changed according with the directory selection shown in Figure 3.21. The paths introduced therein must be valid in order to save the changes successfully.

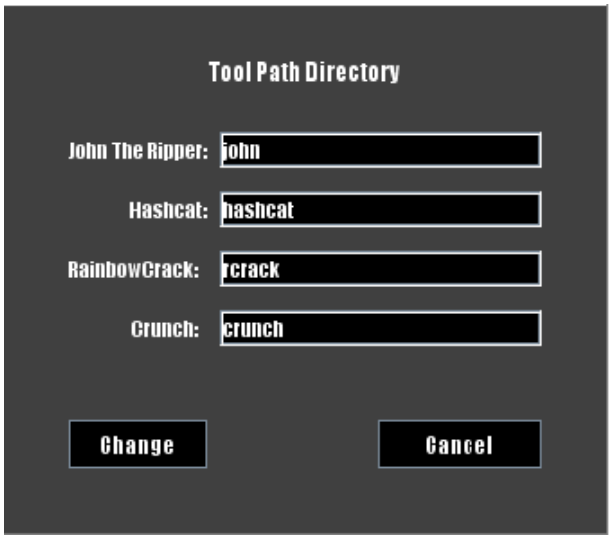

Figure 3.21: The dialog for changing directories of the underlying cracking tools.

- **Shorcuts**: PassCrackGUI defines three keyboard shortcuts to avoid the usage of the mouse for some of the features it provides. The first shortcut triggers the execution of the toolkit:

CTRL+R

Exiting the toolkit can be done with the following shortcut:

CTRL+ALT+E

The last shortcut can be used for editing the colors of the interface:

CTRL+E

3.5 Chapter Summary

The motivation for the development of PassCrackGUI comes from the need to standardize and normalize cracking features from multiple tools in a user friendly manner. This chapter described in detail the development and functionalities of the toolkit, elaborating on the cracking features and usability functionality of the software. It showed that the interface is composed of three main areas, with the work flow oriented from the left to the right. The user starts with the selection of the cracking tools and inputs to crack, evolves to the fine-tunning of several aspects of the algorithms, generates the commands and launches the compromise attempts. The logic supporting the frontend takes care of spawning the tools with the right options and parameters. Nonetheless, a console environment, in which the generated commands are displayed, was embedded in the interface, which allows for the modification of the commands, assuring that any additional features of the tools can also be used. This design decision makes the tool more versatile and increases its longevity. This way, specialized users may adjust the commands at their will, and new options that are added to the underlying tools may still be accessible from within the main GUI in the future.

PassCrackGUI was developed in Java to favor portability. Along the chapter it was mentioned that it was tested in Windows and Linux. Installing the toolkit is simple, though it requires analyzing and assuring its dependencies, as well as following the slightly different installation procedures, also mentioned herein.

Chapter 4

Tests and Password Habits

4.1 Introduction

This chapter describes several analyses concerning password habits conducted in the scope of this master's programme. The next section (section 4.2) discusses the results of cracking attempts to databases containing password representations that leaked to the Internet. A survey on password habits involving people from the academic and industrial worlds is explained in section 4.3, and the most interesting results and some conclusions of these studies are then wrapped up in section 4.4.

4.2 Cracking Leaked Databases With PassCrackGUI

One of the paths that were followed to obtain an idea of the passwords constitutions was to use real databases with passwords representations. If the databases contained password representations (instead of plain passwords), this path would also enable studying how easy it would be for a malicious entity to crack them. Additionally, this exercise would enable testing and eventually fine-tuning PassCrackGUI.

This part of the work started with the search for Portuguese databases containing password representations that leaked to the Internet. It was not hard to find them, as some well-successful *Anonymous* operations, for example, have been devoted to leaking such DBs in the past. All DBs used in this work were obtained from sites like PastBin and GhostBin and they were still available for download at the time of writing of this dissertation. They leaked onto the Internet in two different years: 2013 and 2015.

Beyond the password representations, most of the DBs included many details concerning their users, but only the usernames and password related information (hashes) were used in the scope of this work. The data was processed mostly automatically via scripts and PassCrackGUI, except for minor fine-tuning of the parameters of the toolkit. No login attempts were performed upon successful cracks. The obtained passwords were processed accordingly to produce the results depicted below.

For this part of the work, a total of 12 databases were selected. Table 4.1 contains some information regarding the characterization of the dataset, including the amount of leaked data and the hash function used for protecting the passwords. The different DBs will be hereinafter

referred to as D1, D2, ..., D12.

	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12
# pass. rep.	60	1575	1485	630	7	153	3	3	37	63	21	196
algorithm	MD5	MD5	phpass	MD5	MySQL	MD5	MD5	MD5	MD5	MD5	MD5	MD5

Table 4.1: Table containing information of the password representations in the DBs used in the scope of this work, namely the algorithm used to obtain the representations and the total number of entries in each DBs.

To provide an immediate idea of the time complexity associated with the cracking process and with the different hash functions, a simple experiment was devised. The experiment consisted in calculating hashes with different functions in a laptop computer, to obtain the approximate number of hashes per second it was capable to produce, ending up with table 4.2. The computer used in this experiment is the same that was used for all other calculations mentioned herein. It was a Toshiba laptop from 2012, with a multi-core processor, with 2 cores and 2 threads per core (with 3 MB of cache memory) running at 2.4 GHz each. The hash functions used in this experiment include, but are not limited to, the ones of the leaked DBs. From the analysis, it can be concluded that most systems are using what may be considered nowadays as extremely fast hash functions (e.g., MySQL and MD5). The slowest was phpass, which was used in one of the DBs. Fast hash functions or Password-Based Key Derivation Functions (PBKDFs) ease the cracking process and should be avoided. These results were achieved by running Hashcat via PassCrackGUI while utilizing the brute-force attack model.

Hash Function	Approximate number of calculated hashes per second
MySQL	34 000 000
MD5	24 000 000
SHA-1	15 000 000
SHA-256	7 600 000
SHA-512	2 500 000
phpass	20 000

Table 4.2: Approximate number of hashes generated by the computer utilized in the scope of this work.

The 12 DBs contained a total number of 4233 passwords. After various attempts to obtain strings that lead to the representations, a total number of 1539 passwords were successfully cracked (Figure 4.1). All attacks were performed through PassCrackGUI executing algorithms of the available tools, such as dictionary and brute-force attacks. In addition to the utilization of these two algorithms, several features associated with them, such as replacement tables in the dictionary attack and the use of masks in brute-force, where also utilized. Although the vast majority of the passwords were cracked utilizing the dictionary attack, the use of masks on the brute-force attack was crucial to discovered some passwords. The dictionaries used in these experiments were the following: (i) four files containing Portuguese words (some of the files had words with accentuation, while others did not); (ii) the popular and publicly available RockYou dictionary [Ron11]; and (iii) the Hashkiller dictionary [Bla15b].

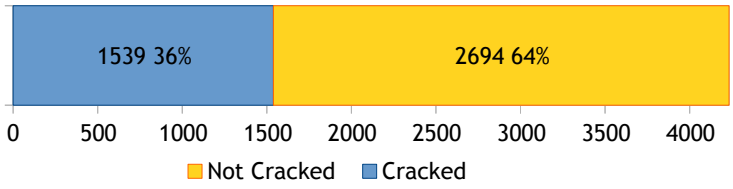

Figure 4.1: Number (and percentage) of cracked and not cracked passwords representations.

The subset of passwords that were cracked from the available data was statistically processed and the results are included below. As such, they are not representative of the entire collections, since only about 36 percent were discovered, though they provide a good picture of the security strength. In security, one weak link is most of the times enough for compromising systems.

One of the first statistics of interest for this data is the length of the passwords, which are illustrated in Figure 4.2 using an histogram. It is possible to emphasize 2 aspects at this point: (i) the biggest passwords in the dataset were 15 characters long and (ii), there seems to exist an abnormal percentage of passwords with length 4.

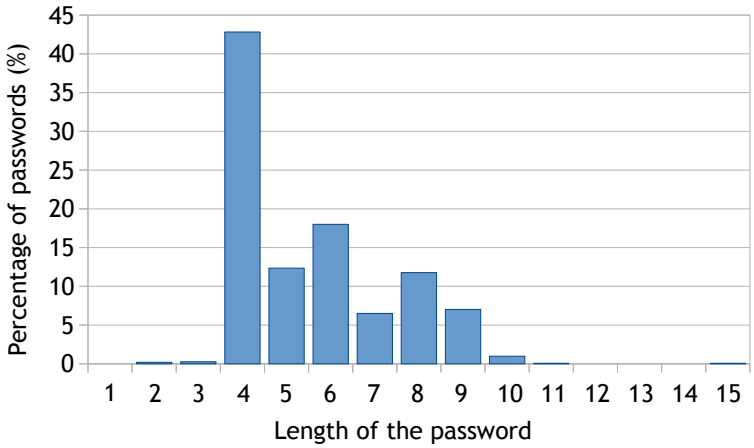

Figure 4.2: Histogram of the lengths of the cracked passwords (including the DBs with PINs).

After analyzing the different characters used in the creation of passwords, it was notice that there was an abnormally high incidence of passwords constituted by digits only, many of these containing only 4 characters. In other words, many users utilize 4 digits words (which can represent years or PINs) as means of authentication. One of the DBs (D2) was particularly profuse on this type of credentials. Figure 4.3 emphasis this fact by showing the number of passwords that use the different sets of characters used in this type of data (digits, lower and upper case, and special characters). The analysis of this figure shows that many cracked passwords are only constituted either by digits, lower case characters and by a combination of digits and lower case characters.

The finding of the high incidence of PINs on one of the DBs was unexpected and somehow

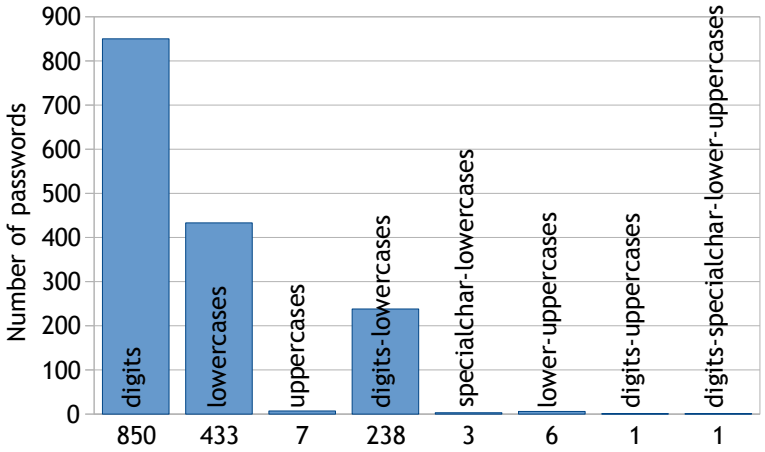

Figure 4.3: Different types of character sets utilized in the passwords from the leaked DBs.

disappointing, given the context. Nonetheless, given that it could be considered more of an artifact, it was decided to include some charts without the cracked passwords for D2. Figure 4.4 shows the histogram of the lengths of the cracked passwords (excluding PINs) and Figure 4.5 illustrates the constitution of the cracked by showing the type of characters used in each one of the positions (in terms of percentage). After removing the PINs, a new mode is highlighted in the histogram of Figure 4.4. Users seem to prefer passwords with 5 to 9 characters.

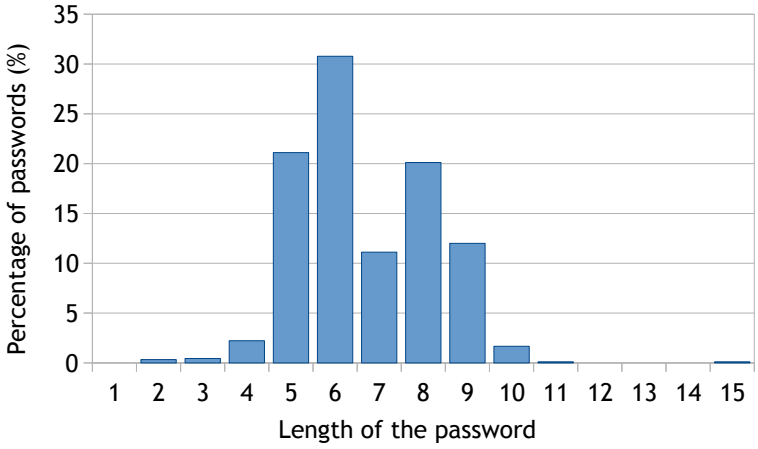

Figure 4.4: Histogram of the lengths of the cracked passwords (excluding the DBs with PINs).

Figure 4.5 shows which type of characters appear on the different positions of the cracked passwords up to the length of 15 (maximum length found in the cracked credentials). Nonetheless, a better understanding of this chart requires it to be analyzed in conjunction with the one of Figure 4.4, concerning the password lengths. For example, 95% of the passwords were smaller than 9 characters and, as such, the chart should be read while keeping that value in mind. One of the observations taken from Figure 4.5 is that passwords are often composed by digit and lower cases. Upper cases and special characters are rarely used.

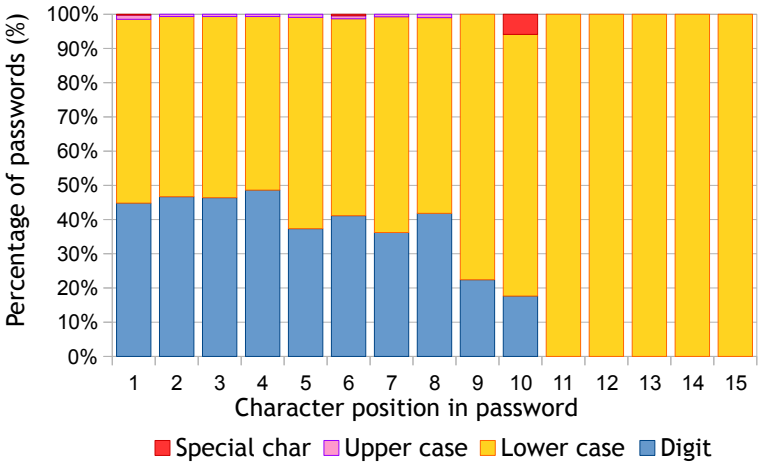

Figure 4.5: Constitution of the cracked passwords in terms of digits, lower case, upper case and special characters (excluding the DBs with PINs).

4.3 Password Habits

The statistics and charts contained in this section were obtained via a questionnaire. In order to preserve the privacy of all respondents, details that could identify them uniquely, such as name, were not asked, and the most personal questions were related with the age, gender and professional area. The information that was gathered is also presented in the form of aggregated statistics for the entire population, and not individually, for the same reason. The questionnaire was comprised by a total number of 21 questions, with some of them only visible depending on previous answers. It was written in Portuguese, given the target audience, and it was fully transcribed to appendix A. A total number of 64 persons participated in this survey.

The first 3 charts included concern the characterization of the respondents (population) and they also to the first 3 questions of the questionnaire. The first chart, in figure 4.6, shows the distribution of the ages by ranges. There were no respondents older than 48 years and it is possible to verify that more than 75% had ages comprised between 15 and 25 years.

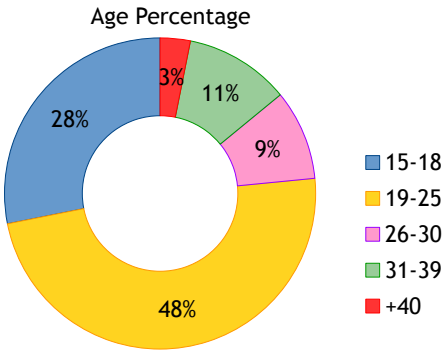

Figure 4.6: Characterization of the population: distribution in terms of age ranges.

The second question in the questionnaire asked for the gender of the respondent. There were

more females (36 women) than males (28 man) participating in the survey, as shown in figure 4.7. Women were generally more willing to participate in the survey than men. Nonetheless, the discrepancy is not that significant and it was considered that the participants provide a good representation of the population.

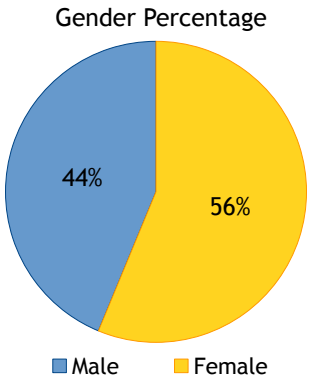

Figure 4.7: Characterization of the population: distribution in terms of gender.

The chart in figure 4.8 provides an idea of the distribution of the sample in terms of professional area (it contains the absolute frequencies of participants per professional or personal background). In terms of representativity, most of the participants were from the the *Health*, *Informatics and Engineering* (other than *Informatics*) areas, followed by the *Education*, *Chemistry*, *Economics* and *Arts* areas. Besides those, there were other persons with different professional or personal background such as high school students.

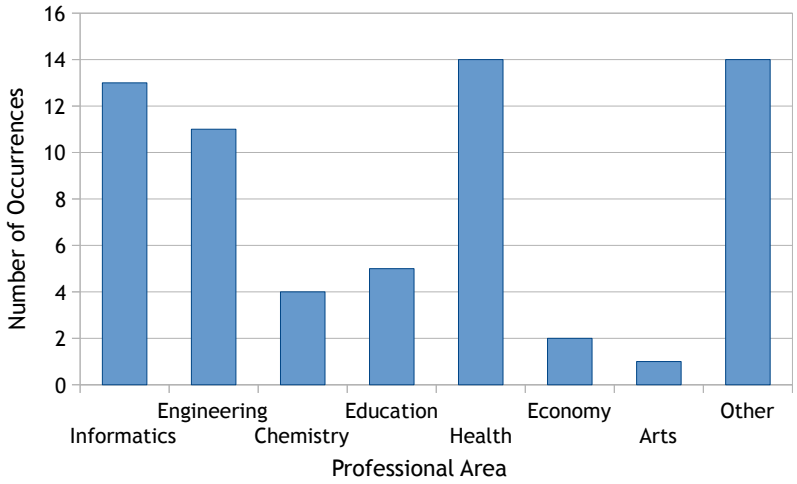

Figure 4.8: Characterization of the population: distribution in terms of professional area or personal background.

A subsequent question of the survey aimed at discovering how many different passwords each participant of the survey was using for authentication. Figure 4.9 summarizes the answears to this question and clearly shows that most of them (40 persons) were using from two to four

different passwords. Three respondents stated that they were using a single password only.

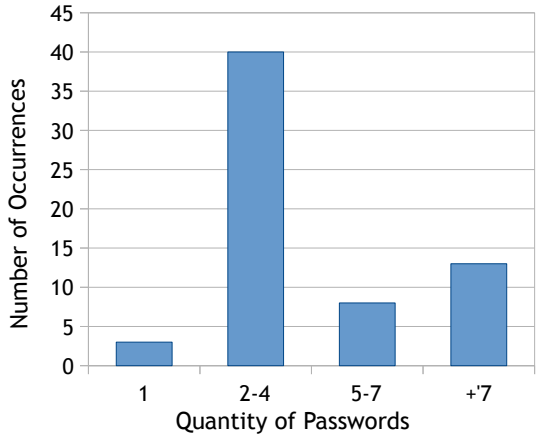

Figure 4.9: Quantity of passwords that the participants of the survey claim to utilize.

On the length passwords subject, 81,25% of the participants (52 persons) claimed that they were using at least 8 characters, with half of them stating that their passwords were at least 10 characters long. The answers to this question are compiled in Figure 4.10. There are still some persons that use a length between 5 and 7 characters or in some cases, even less.

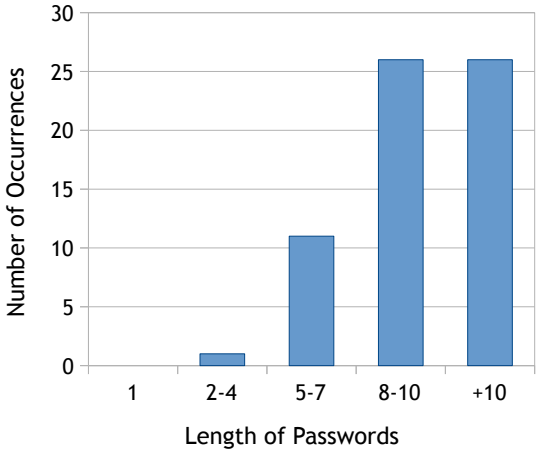

Figure 4.10: Length of the passwords that the participants of the survey claim to be using.

Another interesting statistic derived from the answers of the survey concerns the composition of the passwords in terms of characters. Approximately 90% of the the participants stated that their passwords contained digits; approximately 65% claimed to be utilizing upper cases; and 50% stated they were using special characters also. Approximately 19% of the participants said that they were using a password management software. Notice that the questionnaire included, in last, a field for the participant to create a password similar to the one he or she uses (question 16) and another question asking to indicate which type of construction is more familiar, after

presenting some alternatives (question 15). The chart in figure 4.11 was constructed with bases on the answers to question 16. It provides a graphical perspective over the different sets of characters utilized in the supplied passwords. As can be observable the most representative set includes all different types of characters. Then follows the use of lower cases and digits and lower cases, upper cases and digits.

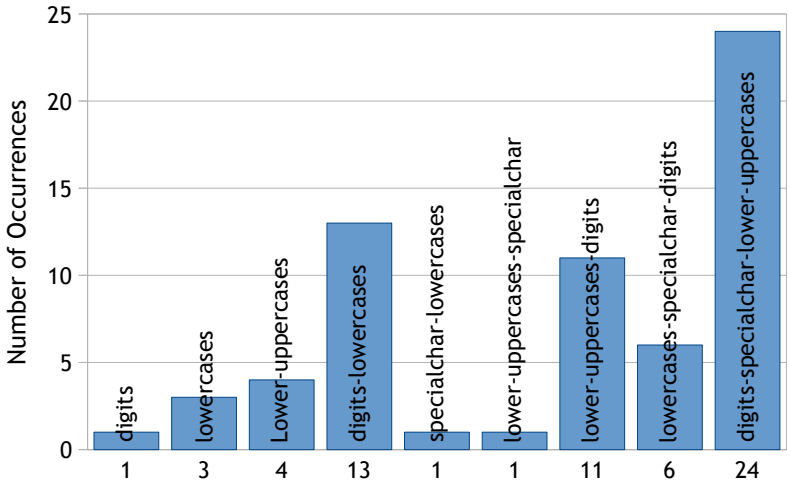

Figure 4.11: Number of different character sets used on the passwords provided by the survey respondents.

Figure 4.12 revisits the subject of the lengths of the passwords, but now for the ones provided in the aforementioned text field. It is possible to observe that most of the provided passwords have 8 to 20 characters. Unfortunately, someone typed a password containing 65 characters: a wake up call to one of the problems of dealing with humans.

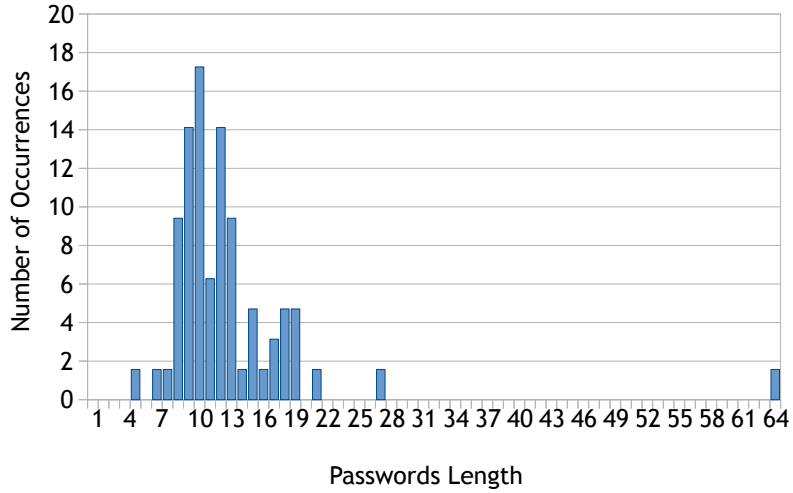

Figure 4.12: Number of characters of the passwords entered by the survey participants in the text box of question 16.

Figure 4.13 provides a perspective over which type of characters appear at each position of the

provided passwords. Though is it not easy to draw many conclusions from the chart, there seems to be a tendency on the usage of special characters towards the end of the passwords (notice that, starting from position 29, the results are derived from a single password). Moreover, people seem to use more upper cases at the beginning of the passwords than at the end, which corroborates the structured reasoning that makes such secrets weaker.

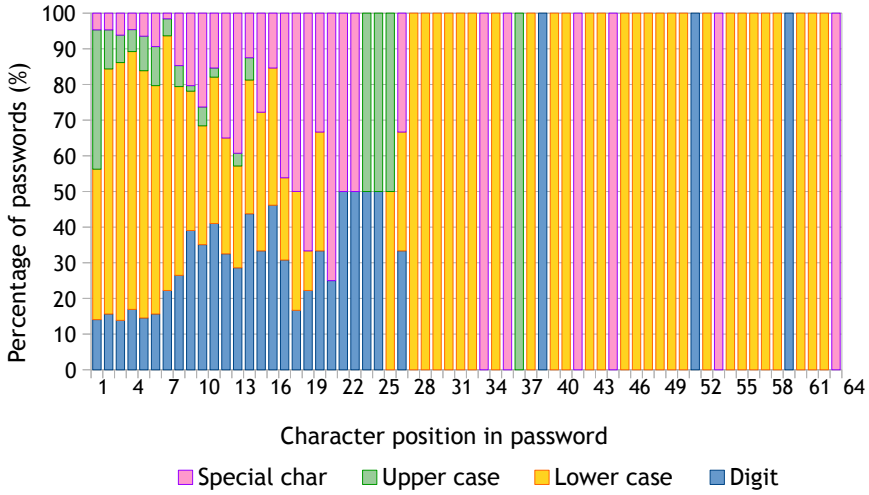

Figure 4.13: Percentage of different sets of characters utilized on the different positions of the passwords entered by the survey participants in the text box of question 16.

The chart in Figure 4.14 was created to obtain an idea on the differences between the passwords entered by the survey participants and what they claimed in questions targeting the character sets. The representation is semi-artificial in the sense that the maximum password length (i.e., 13) is not real and corresponds to the average number of characters of the passwords entered in the text field of question 16. The questions concerning the constitution of passwords were only asking were the participant was typically using each different type of character. The options were: at the beginning; at the middle; at the end. An additional option was asking if the characters were used in an alternated manner. The graphical representation with 13 characters was thus constructed by assuming that the first four positions correspond to the beginning of the password, while the last four correspond to its end; being the remaining five the middle. The decision to make this graphical representation semi-artificial was to favor the comparison with the charts of Figures 4.5 and 4.13. The results seem to corroborate the higher usage of upper chars at the beginning of the password but the usage of special chars anywhere is contradicting with the results depicted on Figure 4.13.

To better quantify contradictory aspects of the answers, Table 4.3 was included. It directly compares the answers to the questions regarding the usage of the several sets characters with the passwords provided in the last field of the survey. For example, it is possible to verify that 32 people answered affirmatively to the usage of special characters, but only 27 use them when asked for a password. On the other hand, from the 32 persons who claimed to not utilize special

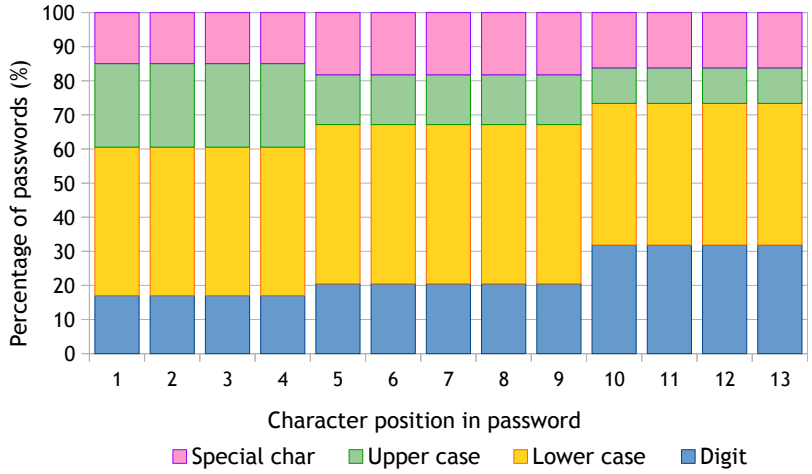

Figure 4.14: Percentage of different sects of characters sets utilized on the different positions of the passwords according to claims of the participants of the survey.

characters, 5 of them did.

Answer to question ⟶	YES		NO	
Actual usage in password ⟶	YES	NO	YES	NO
Used Special Characters?	27	5	5	27
Used Using Digits?	53	5	2	4
Used Using Upper Cases?	37	5	2	20
Used Using Dates?	14	6	3	41
Making Sense?	35	6	3	20

Table 4.3: Relation between what the survey respondents claim to use on their passwords and what they actually use when asked to create a new one.

Table 4.4 ends this section by compiling some statistics that may aid in the identification of stereotypes, similarly to what was done in [MKV+13]. The table contains the average length of the passwords inserted by the participants of the survey, as well as the percentages that characterize their constitution, divided by gender (Male, Female) and by professional areas. Notice that some of the areas depicted in Figure 4.8 are not included in the table because there were not enough participants for those areas, when comparing with the total number of respondents (all below 10% were not included). The total number of participants in each category is also included in the next to last column. Interestingly, women seem to use shorter and less balanced (in terms of sets of characters) passwords than men. People with Informatics background provided longer passwords and used more digits.

4.4 Discussion of the Results

The discussion of the results contained in this chapter need to be contextualized. For example, the results concerning the constitution and length of passwords presented in section 4.2 are limited to the ones that were effectively cracked. On the other hand, the analogous analysis and graphical representations on section 4.3 are constrained by the human factor (humans are

	Password Constitution						
Different Sets	Length (Average)	Digits	Lower Case	Upper Case	Special Char	# Participants	Entropy (average)
Female	11.22	20.37%	60.86%	7.04%	11.73%	36	2.95
Male	14.78	25.83%	48.25%	10.13%	15.79%	28	3.37
Engineering	14.10	31.64%	43.48%	10.21%	14.67%	11	3.45
Informatics	17.61	19.90%	51.80%	11.80%	16.50%	13	3.51
Health	11.50	18.00%	64.43%	6.18%	11.39%	14	3.07
Other	11.29	27.61%	55.93%	5.26%	11.20%	14	2.83

Table 4.4: Statistics regarding length and composition of passwords per gender and per professional areas.

not always honest, when asked about private details). As such, none of the analysis provides a general picture on the usage of passwords. The results are disturbing nonetheless.

In the case of the leaked DBs, it was possible to crack approximately 36% of the passwords using PassCrackGUI with some freely available dictionaries and rules. In many cases, a comprised account is what it takes for an organization to be infiltrated. While there were three DBs in which for which no password was cracked (D4, D5 and D9), there were others where the success percentage was almost 80%. It is possible to conclude that the lack of security that probably leaded to the leak is endemic, and the organizations fail also to correctly apply adequate mechanisms. Most the analyzed databases were using the MD5 function, deprecated several years ago. The in-build MySQL hash function, which is weak too, was used to obtain the representations stored in another DB. Only one DB was using the slow `phpass` hash function, which hinders the discovery of the plaintext due to the slowness of the algorithm.

The statistics on the length and composition of the passwords of the leaked DBs need to be read as *conditioned by the set of dictionaries and rules that were used*. Those statistics suggest that lower cases and digits are the most used character sets. Notice that most dictionaries are almost entirely composed by passwords with such characteristics also, though several other combinations (using masks) with upper cases were attempted too. Looking from the opposite perspective, it is concerning that, except for the PINs, the vast majority of the cracked passwords was already

Regarding the results of the survey, it can be argued that several contractions were found, both internally and when comparing with the results for the leaked databases. In an attempt to quantify the security of the passwords provided in the last question of the survey, their hash value was calculated and stored separately. PassCrackGUI was then used to try to crack them with the same dictionaries of the earlier experiments. This time, only 8% of the passwords were discovered, suggesting stronger combinations and lengths. These observations may derive from the perhaps weak representativity of the general population by the participants of the survey (there were many participants from the Health and from the Informatics professional areas) or from the intentional effort to produce better passwords for the study. When compared with the

ones of the leaked DBs, passwords were using more upper cases and special characters. Apart from that, table 4.3 summarizes the inconsistencies. Interestingly, the position of digits in the passwords seems to differ for both studies.

It was possible to assess the existence of stereotypes, e.g., from the results in table 4.4, which suggests that men typically use longer and more diversified combinations for passwords. Participants with an Informatics background seem to also use better passwords than the ones coming from other areas.

Given the results of both experiments, it is possible to conclude that, in general, the passwords introduced in the last question of the survey are better than those of the leaked DBs. While it is not possible to know if these are indeed the types of passwords used by people in real life, most of them are passwords with a good level of security. On the other hand, given some contradictory facts, this may also be an indication that people are actually aware they are using weak passwords, making an effort to hide that fact on the survey.

4.5 Conclusions

This chapter discussed the outcomes of two important parts of this work. The application of PassCrackGUI for cracking the passwords representations of leaked DBs enabled testing the GUI, illustrate how vulnerable the associated systems are, and obtain an idea of the length and composition of 36% of the passwords. Some answers to the survey had contradicting aspects, but seem to agree on stereotypes such as that men typically define longer and more diverse passwords than women. Participants with a professional background on Informatics seem to be more aware of the need to have stronger passwords also. Apart from the future work, this chapter provides closure to the subjects addressed in the dissertation.

Chapter 5

Conclusions and Future Work

This chapter is divided into two main sections. The first section includes the final remarks and main conclusions of this work, while the directions for future work are briefly discussed in the second one.

5.1 Main Conclusions

One of the most noticeable conclusions is that there are several works on the topics of this dissertation, namely on password cracking and on password habits. On the latter topic, our findings corroborate some conclusions of others works, including stereotypes. For example, it was noticed that men reportedly have slightly stronger passwords than women, at least concerning length and heterogeneity. Unfortunately, some results suggest that people tend to lie when asked for the safety measures they use in their everyday actions. This phenomenon it also known from the literature.

Interestingly, the results of the survey on password habits were different from those obtained from leaked databases. Either the respondents were lying regarding the password they use in practice, or the leaked databases were bad examples. Since we have not specifically chose the leaked databases, and according to other contradictory artifacts of the survey, it is actually more probable that people are not using, in practice, the passwords they think they should use. The though that their answers were going to be scrutinized may have condition their choices. In the case of the leaked databases, a false sentence of security (the users though that the databases were private) may have lead the users in the chosen weaker passwords.

One of the main conclusions drawn from the study on password habits is that there is a lack of understanding on how to generate strong passwords, and the meaning of *password strength*. The study conducted for the leaked databases leads to the conclusions that developers are also using outdated or weak hash functions or password storage procedures. Awareness must thus be improved both at the user and programmer side.

There are several readily available and free tools for password cracking (e.g., John the Ripper, Hashcat, RainbowCrack). Most of these tools are open source and offer a command line interface. They impose a modest learning curve and they are sometimes difficult to use out of the box. Additionally, there are tools specialized in sub problems of password cracking, as in

the generation of masks for passwords. This is the case of PACK, an analysis tool that may be used upstream to other cracking tools for generating masks for brute-force attacks. At the final phase of this work, it was clear that a graphical user interface like PassCrackGUI could indeed be useful for both experienced and for occasional users. Experience users may use this tool for repetitive cracking tasks and leverage the integration of the several tools. The occasional users benefit from the easier interaction for simple tasks.

This work was a multidisciplinary adventure in interesting areas. On the one hand, it permitted the study of human behavior while, on the other, it provided the academic substrate for analysing and cracking leaked databases. Furthermore, the main objective of this masters program was achieved. PassCrackGUI is now at a stable version, readily available for download in [Pas15].

5.2 Directions for Future Work

Several notes along the document suggest that is still a lot of work that needs to be done within the scope of the addressed topics. The following is a list of future research lines:

- The survey revealed contradicting results. As such, it would be beneficial to rethink the questions and the environment in which they are answered (for example via in-person interviews to fully clarify the academic purpose of the study). Alternatively, the access to real databases (with non-hashed passwords) would suffice;

- Most passwords are derived from personal information or from information related with the professional activity or family of their owners. It would thus be interesting to research and develop algorithms to generate dictionaries from text, documents, personal websites and social network pages related with the owners of the passwords. This line of work would eventually lead to the implementation of crawler to gather information and automatically generate the dictionary;

- In order to improve usability, it would be beneficial to provide PassCrackGUI with a context based help system;

- The current version of PassCrackGUI is directed to a broader user base. Nonetheless, it could be useful to implement a professional version of PassCrackGUI (still free and open source) from this release specially oriented to people working in the cyber-crime area, namely police and army related entities. This version would also integrate facilities to interact with online password cracking tools;

- Though the GUI was thoroughly tested within the scope of this work, it is still needed to test its usage in more computers and operating systems, namely in the most recent generation of hardware. This would also be useful to improve PassCrackGUI and possibly

deal with parallel processing optimization;

- The future analyses of more leaked databases is an objective also. the idea is to observe the trends in terms of password habits.

Bibliography

[Air15] Aircrack. Aircrack-ng, 2015. Last accessed on May 15, 2015. Available from: http://www.aircrack-ng.org/. 13

[AJMO96] Scott A. Vanstone Alfred J. Menezes and Paul C. Van Oorschot. *Handbook of Applied Cryptography*, chapter 9- Hash Functions and Data Integrity, pages 321--383. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 1996.

[Ale15] Alex Biryukov and Daniel Dinu and and Dmitry Khovratovich. Argon2, 2015. Last accessed September 27, 2015. Available from: https://password-hashing.net/submissions/specs/Argon-v3.pdf. 10

[All15] FIDO Alliance. Fido alliance, 2015. Last accessed on May 15, 2015. Available from: https://fidoalliance.org/. 10

[BBBB10] Hristo Bojinov, Elie Bursztein, Xavier Boyen, and Dan Boneh. Kamouflage: Loss-Resistant Password Management. In Dimitris Gritzalis, Bart Preneel, and Marianthi Theoharidou, editors, *Computer Security – ESORICS 2010*, volume 6345 of *Lecture Notes in Computer Science*, pages 286--302. Springer Berlin Heidelberg, 2010. 8, 9, 10

[Bla15a] Blackploit. Blackploit, 2015. Last accessed on May 28, 2015. Available from: http://www.blackploit.com/. 15

[Bla15b] Blandyuk. Downloads Page - HashKiller.co.uk, 2015. Last accessed on June 2, 2015. Available from: http://www.hashkiller.co.uk/downloads.aspx. 34

[bof14] bofh28. crunch - wordlist generator, 2014. Last accessed September 28, 2015. Available from: http://sourceforge.net/projects/crunch-wordlist/files/crunch-wordlist/. 15, 20

[Che12] William Cheswick. Rethinking Passwords. *Queue*, 10(12):50--56, december 2012. Available from: http://doi.acm.org/10.1145/2405116.2422416. 8

[Cod10] Codahale. How To Safely Store A Password, 2010. Last accessed on May 15, 2015. Available from: http://codahale.com/how-to-safely-store-a-password/. 8, 9

[Com09] NetBeans Community. Adicionando um Seletor de Arquivos para uma Aplicação Java, 2009. Last accessed on May 25, 2015. Available from: https://netbeans.

org/kb/docs/java/gui-filechooser_pt_BR.html. 19

[D13] Markus Dürmuth. Useful Password Hashing: How to Waste Computing Cycles with Style. In *Proceedings of the 2013 Workshop on New Security Paradigms Workshop*, NSPW '13, pages 31--40, New York, NY, USA, 2013. ACM. 9

[Ead14] Lisa Eadicicco. Inventor of the password - business insider, 2014. Last accessed September 26, 2015. Available from: http://www.businessinsider.com/inventor-of-the-password-2014-5. 1

[Goo15] Google. Google Forms - create and analyze surveys, for free, 2015. Last accessed on June 5, 2015. Available from: https://www.google.com/forms/about/. 55

[Gri02] Grifter. SAM Files and NT Password Hashes, 2002. Last accessed September 26, 2015. Available from: http://news.hitb.org/content/sam-files-and-nt-password-hashes. 11

[Hac16] Hacker Film. HACKER FILM BLOG: Antivirus Maker Bitdefender Hacked, Customer Data Being Sold In Shady Black Market Deals, 2016. Last accessed September 26, 2015. Available from: http://www.hackerfilm.com/2015/07/antivirus-maker-bitdefender-hacked.html. 1

[HAF15] Shiva Houshmand, Sudhir Aggarwal, and Randy Flood. Next Gen PCFG Password Cracking. *Information Forensics and Security, IEEE Transactions on*, PP(99):1--1, 2015. 12

[Has15] Hashcat. hashcat - advanced password recovery, 2015. Last accessed on May 15, 2015. Available from: http://hashcat.net/hashcat/. 14, 20

[Hau14] Van Hauser. THC-HYDRA - fast and flexible network login hacker, December 08, 2014. Last accessed on May 15, 2015. Available from: https://www.thc.org/thc-hydra/. 12

[Hon15] Jason Hong. Password Policies are Getting Out of Control, 2015. Last accessed on May 15, 2015. Available from: http://cacm.acm.org/blogs/blog-cacm/123889-password-policies-are-getting-out-of-control/fulltext. 8

[Joe12] Joe. Foofus Networking Services - Medusa, 2012. Last accessed on May 15, 2015. Available from: http://foofus.net/goons/jmk/medusa/medusa.html. 13

[JR13] Ari Juels and Ronald L. Rivest. Honeywords: Making Password-cracking Detectable. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communi-*

cations Security, CCS '13, pages 145--160, New York, NY, USA, 2013. ACM. Available from: http://doi.acm.org/10.1145/2508859.2516671. 10

[Lab15] Nok Nok Labs. Nok nok labs, inc., 2015. Last accessed on May 15, 2015. Available from: https://www.noknok.com/. 10

[LWH07] Alexander De Luca, Roman Weiss, and Heinrich Hussmann. PassShape: Stroke Based Shape Passwords. In *Proceedings of the 19th Australasian Conference on Computer-Human Interaction: Entertaining User Interfaces*, OZCHI '07, pages 239--240, New York, NY, USA, 2007. ACM. 12

[Mar08] Simon Marechal. Advances in password cracking. *Journal in Computer Virology*, 4(1):73--81, 2008. 11

[MER+15] David McCandless, Tom Evans, VizSweet Balloon Race, Miriam Quick, Ella Hollowood, Christian Miles, and Dan Hampson. IWorld's Biggest Data Breaches; Hacks | Information Is Beautiful, 2015. Last accessed on May 31, 2015. Available from: http://www.informationisbeautiful.net/visualizations/worlds-biggest-data-breaches-hacks/. xxi, 12, 17

[Mic06] Microsoft. How to use the SysKey utility to secure the Windows Security Accounts Manager database, 2006. Last accessed September 26, 2015. Available from: https://support.microsoft.com/en-us/kb/310105.

[MKV+13] Michelle L. Mazurek, Saranga Komanduri, Timothy Vidas, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, Patrick Gage Kelley, Richard Shay, and Blase Ur. Measuring password guessability for an entire university. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, pages 173--186, New York, NY, USA, 2013. ACM. 42

[MYLL14] J. Ma, Weining Yang, Min Luo, and Ninghui Li. A Study of Probabilistic Password Models. In *Security and Privacy (SP), 2014 IEEE Symposium on*, pages 689--704, May 2014. 12

[Oec03] Philippe Oechslin. Making a Faster Cryptanalytic Time-Memory Trade-Off. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 617--630. Springer Berlin Heidelberg, 2003. 14

[Ope13] Openwall. John the Ripper - command line options, 2013. Last accessed on May 25, 2015. Available from: http://www.openwall.com/john/doc/OPTIONS.shtml. 19

[Ope15a] Openwall. John the Ripper password cracker, 2015. Last accessed on May 15, 2015. Available from: http://www.openwall.com/john/. 14, 20

[Ope15b] Openwall. Johnny - GUI for John the Ripper, 2015. Last accessed September 28, 2015. Available from: http://openwall.info/wiki/john/johnny. 14

[Ora14a] Oracle. MessageDigest (Java Platform SE 7), 2014. Last accessed on May 25, 2015. Available from: http://docs.oracle.com/javase/7/docs/api/java/security/MessageDigest.html. 19

[Ora14b] Oracle. SwingWorker (Java Platform SE 7), 2014. Last accessed on May 25, 2015. Available from: http://docs.oracle.com/javase/7/docs/api/javax/swing/SwingWorker.html. 19

[Ora15a] Oracle. How to Use Root Panes, 2015. Last accessed on May 25, 2015. Available from: http://docs.oracle.com/javase/tutorial/uiswing/components/rootpane.html#glasspane. 19

[Ora15b] Oracle. Java SE Development Kit 8 Downloads, 2015. Last accessed September 29, 2015. Available from: http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html. 20

[Pas15] PassCrackGUI. PassCrackGUI GitHub Repository, 2015. Last accessed on September 25, 2015. Available from: https://github.com/PassCrackGUI/PassCrackGUI. 46

[Pet13] Peter. password analysis and cracking kit | projects | sprawl, 2013. Last accessed on May 28, 2015. Available from: http://thesprawl.org/projects/pack/. 15

[ph15] password hashing. Password Hashing Competition, 2015. Last accessed on May 15, 2015. Available from: https://password-hashing.net/. 10

[Pie08] Pieter Ceelen. Windows passwords security, 2008. Last accessed September 26, 2015. Available from: http://www.win.tue.nl/~aeb/linux/hh/Hackers_Hut_Windows_passwords.pdf. 11

[Pro15] RainbowCrack Project. RainbowCrack - Crack Hashes with Rainbow Tables, 2015. Last accessed on May 15, 2015. Available from: http://project-rainbowcrack.com/. 14, 20

[r..11] r...@blackploit.com. hash-identifier - Hash Identifier - Google Project Hosting, 2011. Last accessed on May 28, 2015. Available from: https://code.google.com/p/hash-identifier/downloads/list. 15, 20

[Rap14] Jim Rapoza. Saying Goodbye and Good Riddance to Passwords, 2014. Last accessed on May 15, 2015. Available from: http://www.computer.org/web/computingnow/security/content?g=53319&type=article&urlTitle=saying-goodbye-and-good-riddance-to-passwords. 10

[Ric15a] Ricardo X. P. Santos and Diogo A. B. Fernandes and Pedro Tavares and Mário M. Freire and Pedro R. M. Inácio. Analysis of password habits and leaked databases. In *Atas do 7º Simpósio de Informática (INForum 2015)*, pages 0--11, September 2015. xi

[Ric15b] Ricardo X. P. Santos and Diogo A. B. Fernandes and Pedro Tavares and Mário M. Freire and Pedro R. M. Inácio. Passcrackgui — a graphical user interface for password cracking tools. In *Proceedings of the 10th Conference on Telecommunications (ConfTele2015)*, pages 0--4, September 2015. x

[Ron11] Ron Bowes. Passwords - SkullSecurity, 2011. Last accessed on June 2, 2015. Available from: https://wiki.skullsecurity.org/Passwords. 34

[San15] Ricardo Santos. Hábitos de Palavras-Passe, 2015. Last accessed on June 5, 2015. Available from: https://docs.google.com/forms/d/1uBlJFodDNTRfsD0UfHvkiYWpFd9pmp6LlQYuOrqeMw4/viewform. 55

[SBHM09] D. Schweitzer, J. Boleng, C. Hughes, and L. Murphy. Visualizing keyboard pattern passwords. In *Visualization for Cyber Security, 2009. VizSec 2009. 6th International Workshop on*, pages 69--73, Oct 2009. 12

[sg12] Edge security group. Edge-security group - wfuzz, 2012. Last accessed on May 15, 2015. Available from: http://www.edge-security.com/wfuzz.php.

[Sha14] Pavitra Shankdhar. 10 Most Popular Password Cracking Tools - InfoSec Institute, 2014. Last accessed on May 15, 2015. Available from: http://resources.infosecinstitute.com/10-popular-password-cracking-tools/. 12, 13

[Van15] Rob VandenBrink. InfoSec Handlers Diary Blog - Throwing more Hardware at Password Cracking - Lessons Learned, 2015. Last accessed on May 31, 2015. Available from: https://isc.sans.edu/diary/Throwing+more+Hardware+at+Password+Cracking+-+Lessons+Learned/19341. 12

[WAdBMG09] M. Weir, S. Aggarwa, de B. Medeiros, and B. Glodek. Password Cracking Using Probabilistic Context-Free Grammars. In *30th IEEE Symposium on Security and Privacy*, pages 391--405, May 2009. 12

Appendix A

Survey on Password Habits

A.1 Introduction

The following section contains the set of questions devised for assessing password habits of Portuguese users. The survey is thus in Portuguese [San15]. It was made and delivered via Google Forms [Goo15].

A.2 Survey

Hábitos de Palavras-Passe

O presente questionário é parte de um trabalho de investigação realizado no âmbito de uma dissertação de mestrado. Os seus resultados servem fins puramente académicos. Nenhum dado será facultado ou usado para além deste âmbito. A resposta a este questionário não lhe deve tomar mais do que alguns minutos.

*Obrigatório

1. Idade *

2. Género *

☐ Masculino ☐ Feminino

3. Qual a área mais aproximada ao seu curso? *

☐ Engenharia
☐ Arquitectura
☐ Saúde
☐ Educação
☐ Artes
☐ Economia
☐ Informática
☐ Química
☐ Outra

4. Sabe todas as suas Palavras-Passe de cabeça? *

☐ Sim ☐ Não

5. Usa, ou já usou, algum software para guardar as suas Palavras-Passe? * (exemplo keepass)

☐ Sim ☐ Não

5.1 Se respondeu sim à questão anterior, já usou uma algum gerador aleatório para criar as suas Palavras-Passe?

☐ Sim ☐ Não

6. Usa Palavras-Passe diferentes para sites ou serviços diferentes na Internet? *

☐ Uso sempre Palavras-Passe diferentes para todos os sites ou serviços.

☐ Algumas Palavras-Passe são repetidas. Uso Palavras-Passe mais fortes em sites ou serviços que considero mais importantes.

☐ Algumas Palavras-Passe são repetidas. Não tenho critério para quando Palavras-Passe diferentes em sites ou serviços diferentes.

☐ Uso quase sempre a mesma Palavra-Passe em todos os serviços ou sites.

7. Quantas Palavras-Passe costuma utilizar? *

☐ 1 ☐ 2-4 ☐ 5-7 ☐ +7

8. Quantas Palavras-Passe é que sabe de cor? *

☐ 1 ☐ 2-4 ☐ 5-7 ☐ 8-10 ☐ +10

9. Quantos caracteres costumam ter as Palavras-Passe que sabe de cor? *

☐ 2-4 ☐ 5-7 ☐ 8-10 ☐ +10

10. Costuma utilizar letras maiúsculas nas Palavras-Passe que sabe de cor? *

☐ Sim ☐ Não

10.1. Se respondeu sim à questão anterior, onde costuma usar essas letras?

☐ Início

☐ Meio

☐ Fim

☐ Alternadas

11. Costuma utilizar algarismos nas Palavras-Passe que sabe de cor? *

☐ Sim ☐ Não

11.1. Se respondeu sim à questão anterior, onde costuma usar esses algarismos?

☐ Início

☐ Meio

☐ Fim
☐ Alternados

12. Nas suas Palavras-Passe que sabe de cor costuma utilizar caracteres especiais? *
☐ Sim ☐ Não

12.1. Se respondeu sim à questão anterior, onde costumam estar esses caracteres?
☐ Início
☐ Meio
☐ Fim
☐ Alternados

12.2. Se respondeu sim à questão anterior, depois de um caracter vem uma letra maiúscula?
☐ Sim ☐ Não ☐ Ás vezes

13. Utiliza datas nas suas Palavras-Passe que sabe de cor? *
(Ex: 1994)
☐ Sim ☐ Não

14. As suas Palavras-Passe que sabe de cor podem ser, de alguma forma, lidas ou fazem algum sentido? *
(Ex: qwerty, 12345 ou password)
☐ Sim ☐ Não

15. Com qual das Palavras-Passe seguintes é que a(s) sua(s) é(são) mais parecida? *
(Selecione, no máximo, as três mais parecidas)
☐ password
☐ Passsword
☐ p4ssw0rd
☐ P4sSw0Rd
☐ PaSSWorD
☐ password1994
☐ p4s?swo_rd!

16. Se inventar agora uma Palavra-Passe parecida com a(s) sua(s), como seria? *