University of Crete
Computer Science Department

Trawling the Deep Web

Eleni Gessiou

Master's Thesis

June 2011
Heraklion, Greece

This work has been performed at the **Foundation for Research and Technology - Hellas (FORTH), Institute of Computer Science (ICS), N. Plastira 100 Vassilika Vouton, GR-700 13 Heraklion, Crete, Greece**.

Abstract

In a networked world, content on the Web is blossoming and it is available to anyone who has access to a computer system and the Internet. Accessing the Web, is usually done through a search on a standard search engine, e.g. Google. But, is it only what we see online or is something hidden underneath all that information? The World Wide Web content which is not indexed by conventional search engines, is referred to the Deep Web. This master thesis constitutes an approach to explore several aspects of the Deep Web concerning Personal Identifiable Information (PII).

We conduct two immense privacy case studies that expose Personal Identifiable Information inside the Deep Web. First, we examine database content as the Deep Web. To this end, we highlight the privacy issues that have arisen from the introduction of the Greek Social Security Number (AMKA), in connection with the availability of personally identifiable information on Greek web sites. Second, we conduct another case study that refers to documents' metadata as Deep Web content. We analyze the metadata stored in over fifteen million of documents (DOC, PDF, XLS and PPT) found online and we present the privacy leaks that emerge from the analysis.

Also, we present countermeasures that shield our digital life against disclosure of sensitive information. We propose an information retrieval based method for information leak detection which constitutes an improvement of cyclical hashing so as to both *accelerate* leak detection and *increase* the accuracy of the result. Experiments were conducted on real-world data to prove the efficiency and effectiveness of the proposed solution.

Supervisor: Prof. Evangelos P. Markatos

Περìληψη

Σε έναν δικτυωμένο κόσμο, το περιεχόμενο του Διαδικτύου αυξάνεται και είναι διαθέσιμο σε οποιονδήποτε έχει πρόσβαση σε έναν ηλεκτρονικό υπολογιστή και στο Διαδίκτυο. Υπάρχει όμως μόνο η πληροφορία που βλέπουμε online ή είναι και κάτι παραπάνω; Αναφερόμαστε στο World Wide Web περιεχόμενο το οποίο δεν γίνεται indexed από τις συμβατικές μηχανές αναζήτησης, με τον όρο Deep Web. Η μεταπτυχιακή αυτή εργασία αποτελεί μια προσέγγιση για τη διερεύνηση διαφόρων πτυχών του Deep Web που αφορούν ευαίσθητα προσωπικά δεδομένα.

Σε αυτό το πλαίσιο, διεξάγουμε δύο ευρεία case studies που αφορούν την διαρροή προσωπικών δεδομένων από το εσωτερικό του Deep Web. Αρχικά, εξετάζουμε το περιεχόμενο βάσεων δεδομένων ως Deep Web. Τονίζουμε τα ζητήματα προστασίας των προσωπικών δεδομένων που έχουν προκύψει από την εισαγωγή του Αριθμού Μητρώου Κοινωνικής Ασφάλισης (ΑΜΚΑ), σε συνδυασμό με τη διαθεσιμότητα προσωπικών στοιχείων στα ελληνικά web sites. Στο δεύτερο case study, αναφερόμαστε στα metadata αρχείων σαν πληροφορία του Deep Web. Αναλύουμε τα metadata δεκαπέντε και άνω εκατομμυρίων online εγγράφων (DOC, PDF, XLS και PPT) και παρουσιάζουμε τις διαρροές ευαίσθητης πληροφορίας που προκύπτουν από την ανάλυση αυτή.

Ακόμη, προτείνουμε μια μέθοδο για την ανίχνευση διαρροής πληροφοριών η οποία αποτελεί μια βελτίωση του cyclical hashing , έτσι ώστε να επιταχύνει τον εντοπισμό διαρροών και να αυξάνει την ακρίβεια του αποτελέσματος. Τα πειράματα χρησιμοποιούν real-world δεδομένα που αποδεικνύουν την αποτελεσματικότητα και την αποδοτικότητα της προτεινόμενης λύσης.

Επòπτης: Καθ. Ευàγγελος Μαρκàτος

Acknowledgments

First of all, I would like to thank my supervisor Evangelos P. Markatos. Without him, my initial participation in the lab and the termination of my master studies would not have been feasible. The collaboration with Sotiris Ioannidis has been a life experience. I am particularly grateful to him, because he forced me to surpass my limits and get to know myself better. He has been the main factor for my further collaborations. I will always remember his valuable advises!

Many thanks to Alexandros Labrinidis for the perfect and successful collaboration we had and I hope to go on working together. Also, it would be my omission not to thank Quang Hieu Vu for our collaboration. I greatly appreciate their help.

Michalis Polychronakis introduced me to the idea of my first project in the lab and I deeply appreciate his presence and attitude toward me. I am also very thankful to Elias Athanasopoulos that led and helped me in several works. It was really fun to work with him!

I thank everyone in our lab for their support and the fun time we had together both inside and outside the lab, and in particular Spyros Ligouras, Antonis Papadogiannakis, George Vasiliadis, Demetres Antoniades, Alexandros Kapravelos, Nick Nikiforakis, Iasonas Polakis, Apostolis Zarras, Zacharias Tzermias and Athanasios Petsas and especially George Kontaxis.

I owe a big thank to my whole family for supporting me in every possible way, even in times that I thought I could not make it. I thank my roommate(s) and my friends for the relaxing times spent together, that refreshed me and gave me the strength to go on. Thank you all!!

At last, I would really like to thank the person that offers me calm, positive energy and a nice everyday life; thank you Vasilis!

Contents

List of Figures

List of Tables

1 Introduction

In a networked world, content on the Web is blossoming and it is available to anyone who has access to a computer system and the Internet. Accessing the Web, is usually done through a search on a standard search engine, e.g. Google. But, is it only what we see online or is something hidden underneath all that information? The data returned by search engines is only a small part of the total available information that can be found online. Deep Web [33] is the term used for referring to the World Wide Web content that is not part of the Surface Web, which is indexed by standard search engines, see Figure 1.1[1]. Traditional search engines cannot "see" or retrieve content in the Deep Web - those pages do not exist until they are created dynamically as the result of a specific search. Although, both Deep Web and Surface Web store their contents in searchable databases, Deep Web only prepares results dynamically to respond to a form submission with valid input values.

This thesis constitutes an approach to explore several aspects of the Deep Web concerning Personal Identifiable Information (PII). Especially, we conduct two immense privacy case studies that expose Personal Identifiable Information inside the Deep Web.

Database Content as Deep Web

Our first case study concerns the Deep Web as database content that is accessed only through a web form with valid input values. Greece recently introduced its own Social Security Number, called **AMKA**, that is used for all transactions relating to employment and insurance. Essentially, this number is used as the first step in the effort to modernize the numerous public pension and insurance plans. Although it is not expected to replace the national tax payer ID, already in place, we

[1] Image from Juanico Environmental Consultants Ltd.

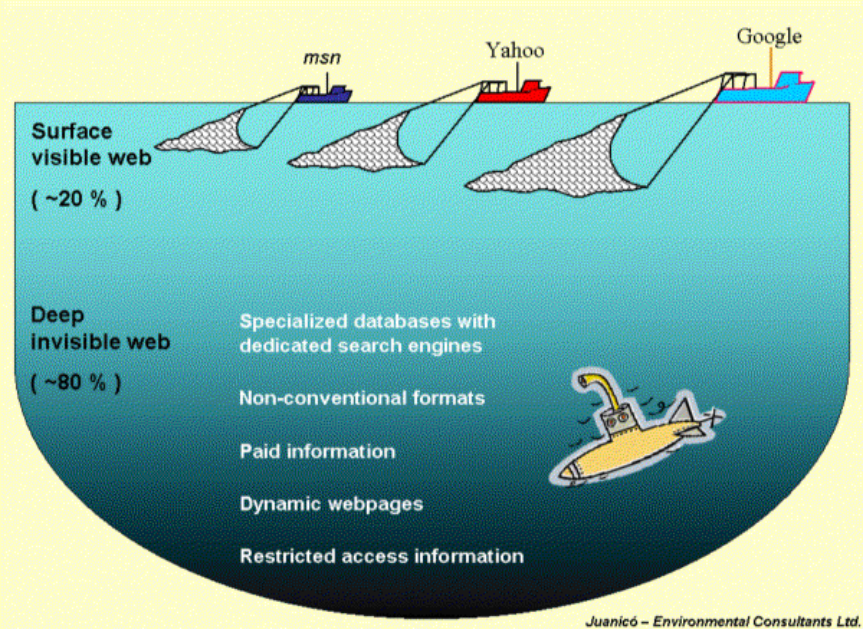


FIGURE 1.1: Visual representation of Deep Web.

strongly believe that this number can take a path similar to that of the SSN in the US and become the ubiquitous and de-facto identifier number for Greek citizens in the near future, further increasing its importance.

The introduction of the Social Security Number (SSN) in the USA happened in 1936 [11]; its original purpose was to track individuals' accounts within the New Deal Social Security program. The SSN has since come to be used as a unique identifier for individuals within the United States for a variety of purposes, from the Internal Revenue Service (IRS) to student IDs at Universities (up until a decade ago). The SSN is generally required by financial institutions to set up bank accounts, credit cards, and obtain loans, partially because it is assumed that no one except the person to whom it was issued will know it. However, the ubiquity of the SSN and its use as an authentication mechanism for financial transactions have intensified the problem of identity theft [31,42], prompting legislation against using the SSN as an identifier that is made publicly available (e.g., on the face of student ID cards at Universities [10]).

Given the importance of AMKA, its current design and implementation are troubling, since they constitute likely failures in terms of privacy and on the potential for identity theft, especially in conjunction with personally identifiable information, that is publicly available on Greek web sites. We perform a study on the availability of personally identifiable information in Greek web sites. Then, we highlight the seriousness of the problem with the design and implementation of the Greek SSN (AMKA), given such availability and illustrate the potential to obtain someone's AMKA by presenting the results of a large-scale study (using

both public figures and private citizens). Finally, we propose solutions to address the current shortcomings of AMKA. This work confirms the lack of awareness on very typical privacy issues with large-scale systems and the lack of guidelines or best practices to that effect, or at the very least, the lack of adoption of privacy standards in the real world. The results of this work can be found in [44].

Documents' Metadata as Deep Web Content

In the second case study, we integrate the notion of Deep Web into documents' metadata. Million of documents are created and shared over the Internet every day. Popular formats for these files are Microsoft Word, Excel, PowerPoint, as well as PDF. Although these document formats are most served by search engines, the documents themselves contain much more data than what was intended to by their creator, that are not indexed by search engines. This data is automatically generated by the applications, such as word processors and presentation managers, and we refer to it as **metadata**. Most of the time, the author of a document is totally unaware of the existence of any metadata associated with it. Moreover, the popularity of the web has led many users to store their files in web servers, which are freely accessible from everywhere. Very often due to weak security configurations, these files become accessible to everyone.

Microsoft Office documents include built-in and custom properties in their metadata [18]. Custom document properties are details about the file's identification, such as the *date* completed and the *author name*. Also, built-in document properties include information such as *title*, *keywords*, *subject* and *comments* that identify the document's content. Similarly, PDF documents include metadata such as *viewer preferences*, *page mode*, etc.

Unfortunately, metadata may contain very sensitive information about the people who have authored or modified the document. In this work, we investigate several security issues that should be considered when thinking about metadata. First, revealing the *creator* of a document can be used for deriving possible usernames used in web applications, such as social networks and web e-mail. Second, revealing the *application used* for the creation of the document may be helpful in determining potential attacks. Note that exploits or computer worms often target specific, known to be vulnerable, versions of an application [54, 64]. Thus, revealing the software and version used to create a document can narrow down an attack targeting a particular user.

Leakage of metadata information has very serious consequences. The most notable example to date is the case of Dodgy Dossier [3], which refers to a document of the British government on Iraq published using Microsoft Word. An analysis on the *revision history* of the document revealed that much of the material of the dossier was actually plagiarized from a US researcher on Iraq. The incident raised many questions about the involvement of UK and the quality of British intelligence during the second Iraqi War. The importance of metadata associated with a document is also highlighted by a recent incident in Arizona [16]. The Supreme Court

unanimously decided that metadata is part of public records and thus must be released when the records are also released. The Dodgy Dossier and the Arizona case are just a few real-world examples demonstrating that document metadata may contain very sensitive or even critical information.

In this work we present a large-scale study of metadata associated with over 15 million publicly accessible on-line documents collected over a period of one year. We used these documents to quantify the amount of metadata stored in on-line documents and to find any sensitive information associated with it. We employ existing libraries and tools to extract, visualize the degree of the metadata diversity and study the social graphs that emerge from this information.

We collected a large dataset consisting of over 15 millions of on-line documents and exposed all stored metadata. Using information solely present in metadata, we developed techniques for creating social cliques, comprised of users that collaborate in the production of a particular document. Next, we focused our study in social graphs deriving from authors working for Fortune 500 companies and searched Twitter for all exported social cliques identified in the documents' metadata. Our search successfully cross-correlated members of cliques with Twitter users. This unveiled that members of a clique form groups of followee and followers in Twitter.

To the best of our knowledge, our study is the first one that tries to construct social cliques solely based on information derived from metadata stored in publicly available on-line documents. The details of this work can be found in [43].

But how can we defend ourselves from this vast information leak?

Information leaks, either due to Deep Web or Surface Web, are a critical problem of computer systems. The leak of confidential data, be it accidental or intentional, may cause huge losses to the data owner. These losses may not only financial, such as termination of contracts or compensation for customers, but also reputation loss whose cost cannot be estimated easily. In particular, according to a study [47] in 2006 conducted by the Ponemon Institute at 31 organizations that lost confidential information, the average cost of a case of information leak was approximately 4.8 million USD. Typical examples of information leak include the case of MediaDefender in 2007 when over 6,000 internal emails were leaked to the outside world [2] and the case of ACS:Law firm in September of 2010 when personal details of 5,300 people became public [3].

There are two primary solutions for information leak detection. The first one is to use specific expressions, keywords or phrases to identify confidential information. For example, a leak of a Mastercard number can be detected by searching expressions of 16 digits starting with two digits in the range from 51 to 55. While this solution is simple and easily applied, its main disadvantage is that it cannot be

[2] http://www.usatoday.com/tech/news/computersecurity/hacking/2007-09-18-mediadefender-leak_N.htm?csp=34

[3] http://www.bbc.co.uk/news/technology-11418962

employed if confidential information cannot be well-defined by expressions, keywords, or phrases. In this case, the alternative solution is to generate fingerprints of confidential information, which can appear in any structure, and check the generated fingerprints against fingerprints obtained from outgoing traffic.

To protect a confidential document from information leak, a simple approach is to generate a fingerprint of the whole document and check this fingerprint against fingerprints of all outgoing documents. Since this approach cannot handle the case where the leaked information is only in part of the document, the popular approach is to employ cyclical hashing to generate a series of fingerprints for the document and use this series of fingerprints, which in turn are checked against the series of fingerprints of outgoing documents. Nevertheless, there are still two weaknesses in this popular approach. First, the popular approach incurs a high cost in both fingerprint generation and leak detection since every part of a document needs to be checked. Consequently, due to the high leak detection cost, this approach is not applicable to systems where a large number of documents must be protected. Second, it is prone to false positives if a lot of common phrases are used in confidential documents, and expectedly this will create a lot of hits when checked against outgoing documents. The reason is because these phrases often exist in all documents.

To address the problems of the popular approach, we propose a solution based on information retrieval to identify only phrases containing sensitive information for fingerprinting. The basic idea of our solution is to check the popularity of phrases before fingerprinting in two ways. We first look at available public documents of the company or organization that we want to protect from information leak. If the phrase exists in these documents, it does not convey any secret information, and hence it is not a sensitive phrase. We then submit the phrase to a search engine such as Google and measure the number of returned results. Intuitively, the higher the number of returned results is, the more popular the phrase is. What this means is that if a phrase has a large number of returned results, it is a common phrase. By removing public and common phrases in documents from the fingerprint generation process, our solution reduces both the cost of fingerprint generation and leak detection, offering higher processing speed. Furthermore, out solution can improve the accuracy of detection by reducing false positives caused by public and common phrases. In summary, our work makes the following major contributions: We propose a novel solution to improve the performance of the traditional approach for information leak detection in terms of processing speed and accuracy. Our core idea is to identify non-sensitive phrases as well as common phrases, and eliminating them from the fingerprinting process of confidential documents. To evaluate the popularity of a long combined phrase which has no returned result from search engines, we propose a novel technique to split the phrase into sub-phrases and identify the popularity of the phrase based on the popularity of its divided phrases. We conducted an extensive experimental evaluation of the effectiveness and efficiency of our proposed method [45].

Contributions: The contributions of this thesis are the following:

- We highlight the privacy risks that surface using Deep Web content
- We perform two immense privacy case studies that expose Personal Identifiable Information inside the Deep Web.
- We propose ways to shield ourselves from information leaks.

Organization: In Chapter 2 we present briefly the availability of Personal Identifiable Information on Greek Web sites. We proceed in Chapter 3, with two case studies that reveal Deep Web content. Particularly, in Section 3.1, we describe in details a case study that reveals PII derived from a database accessed by using a web form and in Section 3.2 we describe our work about documents' metadata as Deep Web content and the privacy risks that stem from them. Next, in Chapter 4 we propose countermeasures for preventing the afore-mentioned privacy leaks, and focus on an information retrieval based method for information leak detection in Section 4.3. We review related work in Chapter 5 and conclude in Chapter 6.

2

Availability of Personally Identifiable Information

In this Chapter we are interested in the availability of Personally Identifiable Information on Surface Web, and especially Greek web sites. A small scale search shows that personal information of thousands of Greek citizens is exposed in the Internet. We find information that completely and uniquely identifies an individual, such as full names paired with father's names, mothers' names, dates of birth, even taxpayer and national ID numbers.

Full Name	Father's FN	Mother's FN	DoB	ID#	Tax ID#	Total
yes	yes	yes	yes	yes	no	50
yes	yes	no	yes	yes	no	1,724
yes	yes	yes	yes	no	no	1,983
yes	yes	no	yes	no	no	3,843
yes	yes	yes	no	no	yes	4,244
yes	yes	yes	no	yes	no	4,895
yes	yes	no	no	no	yes	15,806
yes	yes	no	no	yes	no	22,099
yes	yes	yes	no	no	no	63,211

TABLE 2.1: Availability of personally identifiable information for regular citizens on Greek web sites. A person's Full Name includes his First Name and Last Name. FN stands for First Name.

2.1 Greek Web sites

The first part of our study aimed to quantify the availability of personally identifiable information in Greek web sites. Specifically: "*First Name*", "*Last Name*", "*Father's First Name*", "*Mother's First Name*", "*DoB*" (Date of Birth), "*ID#*" (National ID number) and "*Tax ID#*" (Taxpayer ID number). To quantify this, we queried Google with all possible permutations of these personal details, limiting our search to `.gr` web sites. The results were filetype-restricted to `.xls` files, because large-scale publication of personally identifiable information happens primarily through spreadsheet files.

Table 2.1 shows the amount of personally identifiable information that is available in Greek sites in spreadsheet format[1].

Note that the fields "Full Name" and "Father's First Name" are always "yes". The rest fields shown in the table were not always requested in the search query; the indication "yes" indicates their presence in the query and the indication "no" their absence. One may notice that half of the possible permutations in Table 2.1 are missing, this is because they returned no results. Note that the "Total#" should be viewed as the minimum volume of data that can easily be found on the Greek Web.

If we consider that an individual can be identified uniquely by their full name, their father's and mother's first name, then the fact that a malicious party can identify over 60 thousand people, is quite worrying. Moreover, it is surprising that more than 15 thousand taxpayer ID numbers are publicly available by simply querying a search engine.

Although knowing someone's first/last name, their father's and mother's first name, and their date of birth (i.e., for the 1,983 people in the third row of Table 2.1) might not seem to pose a serious privacy risk, the introduction of Greek Social Security Number, called AMKA, makes it quite dangerous. The fact that there is a way to find out a person's AMKA through a web form , as we will describe in Section 3.1, by knowing these five pieces of information about them, triggered our curiosity about AMKA and motivated our work concerning the introduction of Social Security Numbers in Greece [44].

2.2 Greek Government's Newspaper

The Greek Government's Newspaper is the formal mean of publishing the acts of civil, political and administrative institutions of the Greek Republic, as well as legal entities of public and private law, recognized by the current Constitution. Many issues can be found online in PDF format. Among others, Greek Government's Newspaper consist a source of individuals' Taxpayer ID numbers with the corresponding individuals' full names. For example, we extracted 270 Taxpayer ID numbers out of 490 issues of the Greek Government Newspaper. And even worse,

[1] Personally identifiable information was also found for my mother, while conducting this study!

the Chief of the Opposition party proposed the creation of an electronic KEP (e-KEP) [9], where each Greek citizen would be identified by his/her Taxpayer ID number.

2.3 Census of Greek citizens

The census of all Greek citizens occurs every ten years, where people who live in Greece are enumerated basically for statistic reasons. The most recent census of Greek citizens was planned and completed in May 2011. The enumerators that conduct the census are Greek private figures that are employed temporally for the needs of census. For informative and security reasons, all enumerators' personal information, that includes full name, father's and mother's first name, was announced online. The announcement stood for over 85K individuals who worked during the census process. This intentional information leak more than doubles the 65K number from the last row in Table 2.1.

3
Case Studies

In this Chapter, we present two case studies that reveal PII leakages in the Deep Web. First, we highlight the privacy issues that have arisen from the introduction of the Greek Social Security Number (AMKA), in connection with the availability of personally identifiable information on Greek web sites, shown in the previous Chapter. In particular, we identify privacy problems with the current AMKA setup and present data from a web study exposing these problems and information leaks in the Deep Web.

Second, we analyze the metadata stored in online documents and extract information related to social activities, no matter if the documents contain sensitive content. Our analysis reveals the existence of exactly identified cliques of users that edit, revise and collaborate on industrial and military documents. We proceed and examine cliques based on documents downloaded by Fortune-500 companies' sites. We construct their graphs and measure their properties. The graphs contain many strongly connected components, that experience the properties of a social graph. The a priori knowledge of a companyUs social graph may significantly assist an adversary in launching targeted attacks. Finally, we cross-correlate all members identified in a clique with users of Twitter, and show that it is possible to easily match them to their Twitter accounts.

3.1 A Greek (Privacy) Tragedy: The Introduction of Social Security Numbers in Greece

We use all the available information found on Greek Web sites, Section 2.1, in order to harvest as many AMKAs as we could. The harvested AMKAs constitute part of the Deep Web content and especially database content, as they become accessible only after using a web form. Search engines are not able to index them, as they are stored in a database that only makes them visible after interacting with it via an online form.

3.1.1 About AMKA

AMKA has the following 11-digit format: `YYMMDDxxxyz`, where the first 6 digits encode the person's date of birth (`YYMMDD`), the following 4 digits are a *sequence number* for people born on that date (`xxxy`) and the last digit is a control digit (`z`). The sex of the person is encoded in the last digit of the sequence number (`y` of `xxxy`): even digits are assigned to women and odd digits are assigned to men[1]. This results in disclosure of both the date of birth and the sex of a person by solely looking at their AMKA!

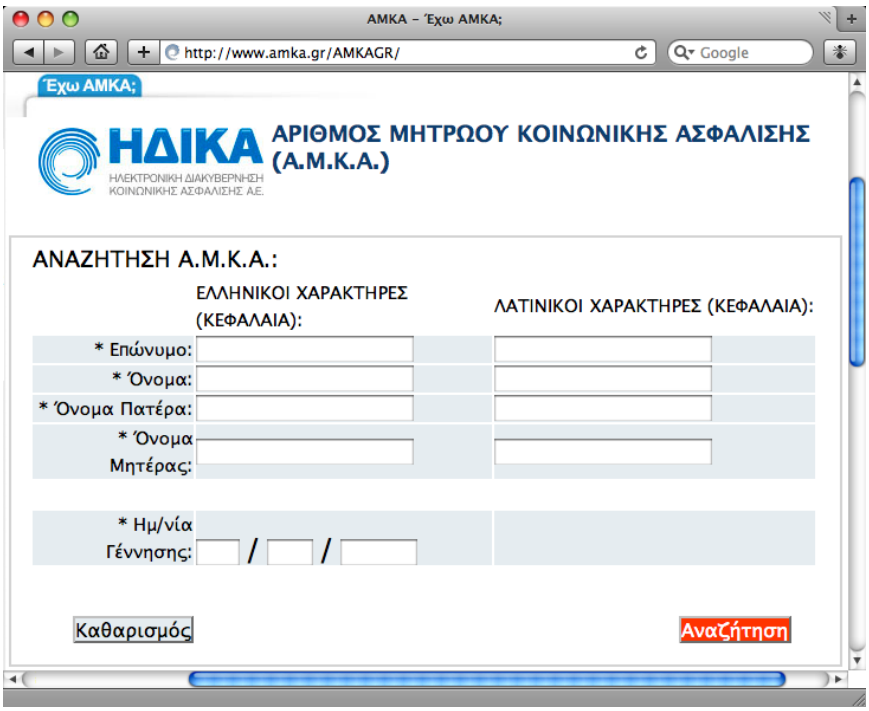


FIGURE 3.1: Screenshot of the AMKA's Web site form.

[1]This information is included in the welcome letter sent out to some AMKA recipients and was mentioned in [7, 8].

The second, and even more troubling failure in the implementation of AMKA, is in the way a person can find what their AMKA is. Currently, there is a heavily advertised web form in place (`http://www.amka.gr/AMKAGR/`), which asks for the following information:

- First Name (Όνομα)
- Last Name (Επὶθετο)
- Father's First Name (Όνομα πατὲρα)
- Mother's First Name (Όνομα μητὲρας)
- Date of birth (Ημερομηνὶα γὲννησης)

to provide a person's AMKA (Figure 3.1).

Although this can be seen as a big convenience for Greek citizens, to establish whether they have an AMKA (and what that number is) or not (and need to apply for one), it is also a problem in terms of potentially exposing many individuals to identity theft. A malicious third party can misuse such a system in three ways:

1. Find the AMKA of public figures, by obtaining the required data already available on the Web (e.g., in wikipedia or their Facebook profiles).
2. Find the AMKA of citizens, by gathering personally identifiable information that has already been (improperly) published about them on the Web, as we showed in the previous Chapter.
3. Find the AMKA of citizens for whom not all of the five above fields are known, simply by brute-force guessing.

Before testing these ideas, we wanted to be certain that such actions would not violate the web site's acceptable use policy, despite our investigation being done purely for academic purposes. To our surprise, we found that the link to the acceptable use policy was broken. In fact, it seems that it has not yet been supplied (as of June 19, 2009), since the link currently points to `http://www.amka.gr/#`, i.e., a placeholder.

3.1.2 Methodology and Results

In this subsection we present the methodology we used to conduct our study. Our goal was to discover whether it is possible to find the AMKA of a person and how much effort is needed to do so.

The data sets we used for our study include personal details of both public figures and regular citizens, found on the Web. As mentioned earlier we need the five attributes - "First Name", "Last Name", "Father's First Name", "Mother's First Name", "DoB" - in order to query the system for an AMKA. Finding all five

of them for each individual would be ideal but not always the case, as Table 2.1 shows. That led us to divide our data-set in two categories: (i) *Exact Data*, for cases when we have all five fields for a person, and (ii) *Non-exact Data*, for when one or more fields are missing.

Exact Data

The *Exact Data* set consists of the personal details of individuals (public figures and private citizens) for whom we were able to collect all five of the required attributes. The strategy we followed was straightforward; we simply queried the system for each individual. To accomplish this with the minimum amount of effort we built a script, written in python, that automated the entire procedure. This script issued one HTTP POST request per individual with the appropriate fields filled-in and parsed the AMKA when available in the results web page.

	Total	with AMKA	%
Public	259	171	66
Private	1,983	1,490	75.1

TABLE 3.1: Results of the *Exact Data* set.

The results of the *Exact Data* set are shown in Table 3.1. The first row contains the results after querying for 259 selected Greek public figures. We collected information about them from public web sites (e.g., wikipedia, their personal pages, fan sites, etc.), all discoverable through Google. As far as the private citizens were concerned, we managed to collect `.xls` documents, mainly from two different sites, which contained all the personal information required for searching, that is "DoB", "Mother's First Name" etc. Most of these lists were found on a hospital's website and their entries referred to nurses and midwives. As we can see, 3 out of 4 already had an AMKA which we collected successfully.

We should note that failure to find a person's AMKA when having all of the five required fields can be attributed to one of four reasons:

1. The person's data available on the Web is incorrect.
2. The person's data entered in the AMKA database is incorrect.
3. The person does not have an AMKA number yet.
4. For some public figures, the system does not convey the AMKA, unless you entered an extra information, such as National or Taxpayer ID Number.

The last two reasons seem to be the most possible reasons for the majority of our cases.

To compare the public availability of SSNs in the US versus AMKA numbers in Greece, we used Google to assist us in collecting as many SSNs as possible.

Our search resulted in the collection of only 270 SSNs. If we take into account the population of USA, which is in excess of 300 million and the population of Greece which is approximately 11 million, the percentage difference between the number of AMKAs and the number of SSNs that are publicly available is huge. Additionally we were able to find 27 SSNs of people that were deceased. This is an alarming fact because a malicious party can use one of these SSNs for illegal purposes, presenting to be the legal owner of it with fewer chances of getting detected.

Non-exact Data

Despite the fact that in most of the cases it was possible to find all the required personal details needed to query for an individual's AMKA, there were cases that only some of them could be found. During our study we faced two distinct categories of missing attributes, either missing "Mother's First Name" or "DoB" (day and month – the year was known). The way we dealt with both cases was brute-forcing the missing attribute. Doing so in the second case was trivial. We simply tried all possible combinations of day and month. In the former case we created a list of the most common Greek female names, 287 in number and tried them (see Table 3.2).

	Total	with AMKA	%
Public-NM	7	3	42.8
Public-ND	12	9	75
Private-NM	757	618	81.6

TABLE 3.2: Results of the *Non-exact Data* set. The -NM suffix stands for "No Mother's First Name" and the -ND stands for "No Date of birth".

The probability of guessing the missing date of birth is one, so in our case we are certain that only 75% of the public figures have AMKA. Unfortunately, in the missing mother's name case, we are not able to measure the success of our brute-forcing technique because we do not know the exact fraction of individuals that do have AMKA. Finally, in case of private figures, we note an increased rate of 81.6% in the last row of Table 3.2.

Using a larger subset of Greek female names we could have seen an increase in the success rate, but this was not the main goal of this study. Our goal was a proof-of-concept discovery of AMKAs and not an exhaustive search style database extraction.

The overwhelming majority of the public figures for whom we found an AMKA, are politicians, but we also found the AMKA of a celebrity. We also found the AMKA of three journalists from major TV stations and two AMKAs of famous Greek athletes.

3.1.3 Extended Methodology and Results

In the previous Subsection, we showed how someone can abuse the AMKA's web form in order to learn AMKAs, that otherwise could not possess. However, the methodology we used, required us to be familiar with many pieces of information for an individual. In other words we needed to know, apart from his/her full name, his/her father's first name and his/her mother's first name, or his/her date of birth. In real life, it is trivial to get to know someone's full name but there is not an easy way to find more personal information. The interesting question raised here is whether a malicious party can obtain AMKAs owned by the vast majority of Greek citizens, given that he/she may know at least the citizen's full name but no other piece of information that could help him to use the above method to extract the desired AMKAs.

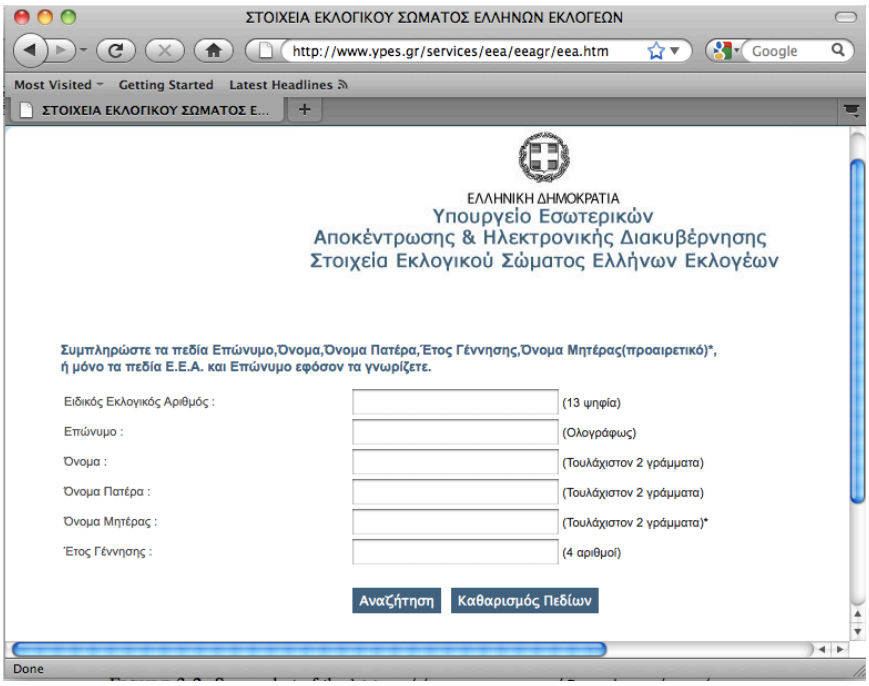


FIGURE 3.2: Screenshot of the `http://www.ypes.gr/Services/eea/eeagr/eea.htm` Web site form.

To this end, we extend our data set by using information found in multiple Greek sites in order to find more AMKAs. We aggregate pieces of information, from different web sites, aiming at extracting as many AMKAs as we can. Our new approach consists of three steps, employing a particular site in each step:

1. We use the Greek Yellow Pages (`http://www.xo.gr`), so we can supply our algorithm with full names and a father's name, or his initials, if that is possible. This way we create the initial pool of individuals that we want to

find their AMKA. Note here that an adversary can use whoever's full name, as the first step of his attack. We use Greek Yellow Pages as an easy way to provide our method with plenty of full names.

2. In the next step, we use the `http://www.ypes.gr/Services/eea/eeagr/eea.htm` site, which is provided by the Ministry Of Interior Administration and Decentralization, in order to facilitate Greek voters to find the place where they should vote during election time. The particular site provides a web form, shown in Figure 3.2, that requires from the user the following information:

   - Last Name (Επώνυμο)
   - First Name - at least 2 letters (Όνομα)
   - Father's First Name - at least 2 letters (Όνομα πατέρα)
   - Year of birth (Έτος γέννησης)

   The Mother's First Name (Όνομα μητέρας) is optional. But, apart from the voting place, its output includes full father's and mother's first names. If we already have the father's initials, we simply bruteforce the year of birth. Otherwise, we also bruteforce the first two letters of the father's first name, using a dictionary of the 100 most common Greek male names. Of course we could also exhaustively search all 2-letter combinations instead. So, our main advantage of this step is extracting full mother's name and full father's name (in case that we know only the first two letters).

3. In the final step we use `http://www.amka.gr/AMKAGR/`. In this step, we only need to bruteforce the day and the month of the date of birth, in order to find someone's AMKA.

For proof-of-concept of our extended methodology, we chose four basic business categories, e.g. doctors, electricians, layers and plumbers, from the Greek Yellow Pages in the city that our laboratory is located. The number of full names taken from the Yellow Pages is shown in the second column of Table 3.3, categorized by business in each row. The number of individuals that we could manage to gather further information using the second step of our algorithm is shown in the third column of the Table 3.3. We refer to it with the name "Ypes" derived from the url name that the particular web form is located. Finally, the last column of the Table 3.3 depicts the number of the individuals that we actually extracted AMKAs for.

Our method employs a total of three different web forms, as explained earlier. The latter two forms may constitute two different points of failure. In Table 3.3 we see that the number of individuals that we gather information for in the second step of the method ("Ypes" column), is quite smaller than the corresponding number of individuals belonging in the initial pool of our method ("Yellow Pages" column).

The percentage decrease fluctuates from 89%, for the business category of "Doctors", to 49%, in the best case of the category of "Plumbers". This decrease can be explained in two ways:

1. In case of brute forcing the year of birth in the web site form, we use a time space from 1945 to 1980. It may be the case that the individuals are either older or younger than the time space we use.

2. For brute forcing the first two letters of the father's first name, we use a dictionary of the most common Greek male names. There is possibility that we do not include in our dictionary someones' father name.

We use a limited time space and dictionary, in order to reduce the number of the queries we make and thus the time it takes. Hopefully, there is a good possibility that the information of majority of the individuals agree with our selection of years of birth and fathers' name. In total, we managed to obtain AMKAs for the 23.3% of the initial individuals.

Business Category	Yellow Pages	Ypes	with AMKA
Doctors	100	11	10
Electricians	148	43	42
Layers	178	83	48
Plumbers	144	73	33
Total	570	210	133

TABLE 3.3: Results of the extended three-step method, categorized by business category.

Using the three-step method, described above, we prove that someone is able to extract an individual's AMKA, solely by knowing his/her full name. Although, our method does not succeed in all cases, in other words we could not find the AMKA of all individuals we wanted to, we believe that a malicious party would make targeted attacks with a relatively small number of victims and a possibly targeted time space, and its percentage of success would be greater. The privacy breach exposed by this discovery should raise major concerns about the aggregation of information from several web sites.

3.1.4 Observations and Limitations

During the course of our study, we noticed some interesting characteristics and updates in the behavior of the AMKA web form (Figure 3.1) that we find worth mentioning.

First of all, we observed that if the day and the month of a date of birth are left blank, the system responds with a new form asking for the individual's Taxpayer or National ID number. An additional observation is that if someone fills the year

of the field "DoB" incorrectly then the system does not return an AMKA. We infer from these two observations that the company which is responsible for setting up the system is making some effort in securing the citizens' AMKA. Also, the system recognizes synonyms of many common Greek names and thus assists users of the web form.

At some point however, while conducting our experiments we noticed the following odd behavior. We would fill all the fields normally except the "DoB". There we would only fill in the year of birth and the website would still return the AMKA. Fortunately, in the interest of privacy of Greek citizens, this only lasted for a few days in May 2009 and then was reverted back to responding with a new form asking for the individual's Taxpayer or National ID number, when the date of birth field was incorrect or partial. Another error we noticed was in the case that only the month was missing from the "DoB" field. In that case we get a pop-up with the following message "ORA-01858 COMMUNICATE WITH THE ADMINISTRATOR". This is clearly a database error [12], and we identified it to be: "a non-numeric character found where a digit was expected".

We also noticed that there is a limitation and a variation in the number of characters in the fields "First Name", "Last Name", "Father's First Name" and "Mother's First Name". The first two fields can hold up to 20 characters, the "Father's First Name" field can hold up to 15 characters and the "Mother's First Name" field can only hold up to 10 characters. The number of characters limitation may create problems in case of multiple first or last names, which is sometimes the case in Greece [7].

Another weakness of AMKA is the fact that only two digits of the 11-digit format of AMKA encode the person's year of birth. This may cause problems since it is impossible to tell if someone was born in 1908 or in 2008, solely by looking at their AMKA. As we already mentioned, the sex of a person is encoded in the next digit - odd digits are assigned to men and even digits are assigned to women. We found however, two cases that this was not the case. Both cases were for women, the digit was odd instead of even, and we believe that to be in error.

3.1.5 Scenarios

Armed with the knowledge of a citizen's AMKA, an adversary has the opportunity to conduct several malicious and dangerous actions.

Private Data Confirmation A curious, or worse, malicious party, can use the online interface to confirm a citizens birth date, parents' names, etc. They can then use this information for other activities, from "harmless" gossip about someone's age, to impersonation.

Identity Confirmation An adversary may use the website to confirm guesses. Essentially one can plug in random values and get back confirmed identities, using the site as an oracle. If guessing is easy (hence cheap) you can actually use the interface to mine out the entire database. One can then use these identities, as before, for impersonation.

False Medical Payments Depending on how AMKA is used it may be possible to permit the generation of false medical payment records (medical exams, prescriptions, hospital visits, etc.) using harvested AMKAs. If a malicious party acquires a large number of them, they may be able to charge minute amounts in medical claims that will not trigger any alarm, but still generate substantial income.[2]

Identity Spoofing If AMKA starts being used for identity purposes, in the same way the National ID card, the passport number or the Taxpayer ID are currently being used, it can open a whole new way to commit identity spoofing with all the related problems this can cause.

Future Uses Finally, it is unclear at this point what other uses AMKA may have in the future. Therefore it is hard to estimate the full privacy impact it will have on the Greek citizens.

3.2 Digging up Social Structures from Documents on the Web

This Section examines information leaks stemming from documents' metadata. Although, search engines can now index several documents' file formats, they do not index their metadata. So, in this part of our study, we consider metadata as part of the Deep Web content.

3.2.1 Methodology

In this Subsection we outline the basic methodology we use for the data collection. We first present the tools and techniques we employ for gathering the sample. We then discuss our sample's properties. Finally, we give a short presentation of some interesting facts related to the metadata extracted from our sample.

Overview

One rich source of on-line documents is a popular search engine, like Google. We created a custom web scraper using the Python [65] scripting language, which is able to parse search results produced by Google. According to Google's policy, Google search engine does not serve more than 1,000 results per query [24]. We therefore used a dictionary to produce a series of queries which can generate a large set of search results.

The query process works as follows. We take all words of more than three letters from the English dictionary, and use them to form a query for Google. Each query is composed of one English word, taken from the dictionary, and the *filetype* directive used by the Google search engine. This directive assists in producing a result-set composed solely of specific filetypes.

[2] http://www.nytimes.com/2009/06/13/health/x13patient.html?hpw

We extract the URLs pointing to documents based on their extension (`.doc`, `.xls`, `.ppt` and `.pdf`). Once a file is spotted in a set of Google results, we download the file and verify that the extension of the file matches the MIME type [14] which is advertised in the HTTP response issued by the host of the file. We discard all documents for which the file extension does not match the advertised MIME type for the following two reasons. First, it has been documented that many web servers are not configured properly [53] to serve all files with the correct MIME type. Second, it is a well known practice for web sites that host malware to advertise wrong MIME types in order to lure the user to open the malware, which is camouflaged under a fake extension. Thus, we remove all files downloaded with a discrepancy between the extension and the MIME type, since we do not want to have a biased sample due to issues not directly related with privacy leakage.

For each downloaded file we proceed and extracted all possible metadata. We use the hachoir-metadata [6] and libextractor [5] libraries for extracting all metadata associated with Microsoft Office documents. As far as PDF files are concerned, we use the Poppler [20] rendering engine. All metadata extracted from Microsoft Office and PDF documents are stored in a MySQL database for further processing.

Sample Properties

Using the technique outlined above we collected more than 5 million MS Word documents, about 2.5 million MS Excel and 2.5 million MS PowerPoint and more than 5 million of PDF documents. Overall, our sample is over 15 million distinct documents. All documents are hashed using the MD5 cryptographic hash function, to remove potential duplicates.

There is a fairly distinct distribution of the various filetypes. Notice that PDF and MS Word files dominate the set, compared to MS Excel and MS PowerPoint files. Our intuition is that PDF and MS Word files are more likely the user's choice for exchanging documents over the web. This may be also a result of the generic nature of MS Word and PDF format, which is ideal for embedding unstructured information. On the other hand, MS Excel and MS PowerPoint documents are more suitable for usage in a corporate environment, providing information structure (financial sheets or presentation slides), and thus less likely to find on public web servers. Nevertheless, our set includes substantial contribution from all of the four non-HTML filetypes considered the most popular to date [25] and thus we consider our study highly representative.

A Peek into Document Metadata

We now present some of our findings relating to the collected metadata.

The CDF of *creation year* and *last modification year* of all Word documents in the sample are shown in Figure 3.3. Both, CDF of *creation year* and *last modification year*, present a huge raise in recent years. The intuition behind this is the

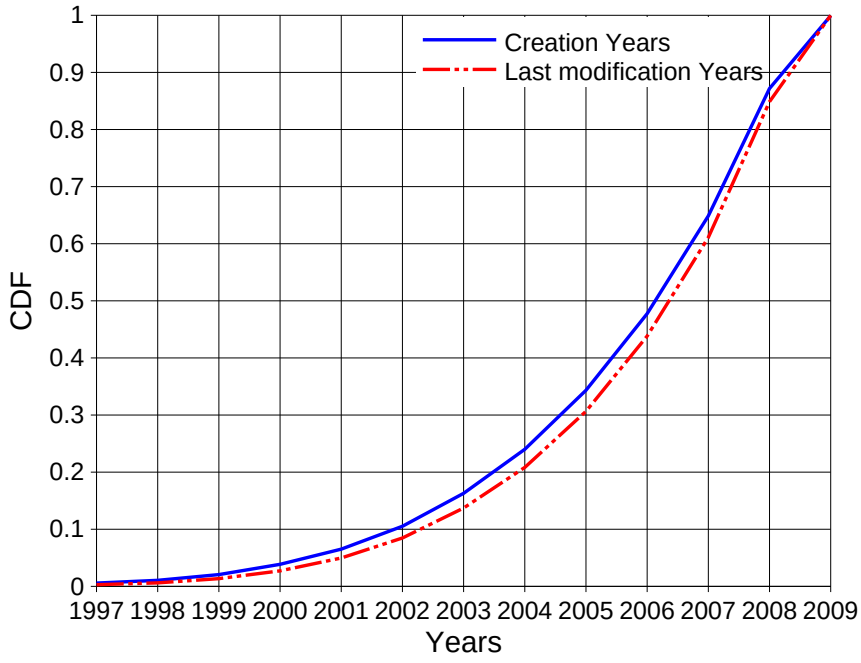


FIGURE 3.3: The CDF for creation and last modification years for Microsoft Word files. The blue, solid line is for *creation years* and the red, dashed line is for *last modification years*.

following. First, people use Word documents more frequently in recent years compared to the past. Second, some users have become more familiar with the Internet and upload more documents. Third, the Google search engine returns the more recent documents than old ones. A slight shift is observed between creation year and last modification year. Apparently, a document that was created in year *X*, is expected to be modified in the years *X+1*, *X+2*, etc.

We notice that almost 93% of Word files use the default template of Microsoft, Normal.dot. However, apart from the default Normal.dot, it seems that many organizations, especially the ones from the governmental sector, use their own custom templates. For example, nearly 1,500 `.doc` files, all downloaded from a City Council's site of a Canadian town, use the same custom template. These files have been modified by a set of different users, which can be identified through *name of creator*, *name of the person who last saved the document* and *revision history* fields. More interestingly, all these names cannot be located in the City Council's site, using the site's search service. Thus, even though these names cannot be extracted from the actual web site, they can be extracted from metadata in files that the web site hosts. In another incident of an Australian governmental organization, about 99% of all documents, based on the same *templates used*, were last mod-

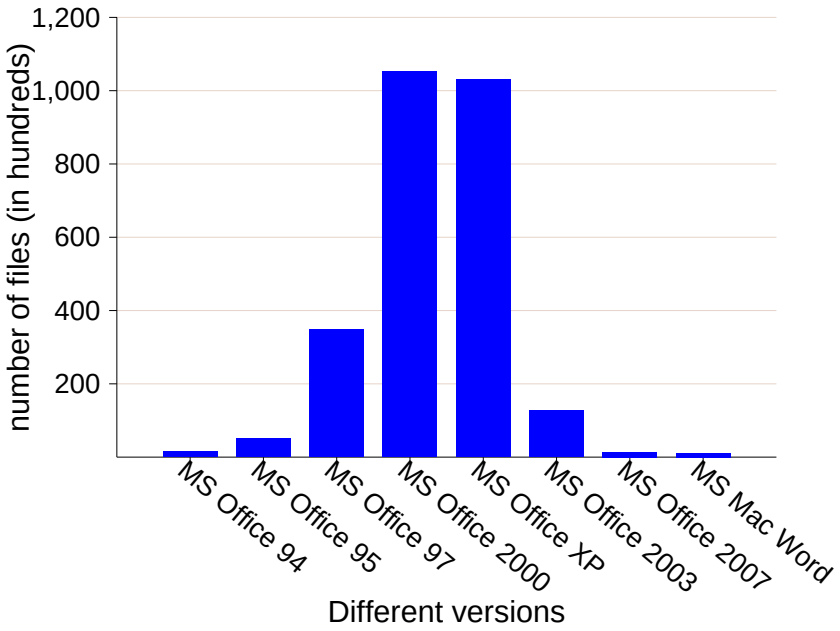


FIGURE 3.4: Versions of Microsoft Office used in Word documents.

ified by a user who is identified, through the above mentioned metadata, as the organization's CEO.

The application used for creation and modification of Word documents is mainly the Microsoft Word software. Figure 3.4 shows the popularity of each version of Microsoft Word. The figure shows that Microsoft Office 2000 (Office 9.0) and Microsoft Office XP (Office 10.0 or Office 2002) are the versions most commonly used. Only a few thousand of Word files were created by MS Word for the Mac. An intriguing aspect to examine is whether the most or the least popular versions leak more information. Although Microsoft Word 2000 is the version used to create the majority of the documents included in our set, it is obviously the version which reveals the most information according to Figure 3.5. For example, a significant fraction of documents created with Word 2000 contain information in the *revision history*. Microsoft Word for the Macintosh presents low levels of metadata presence almost in all fields, and as far as *revision history* is concerned no information is revealed. The following metadata types are not listed in Figure 3.5, because they are included by default in all versions of Microsoft Word: *creation date*, *last modification date*, *application used* and *template used*.

In Table 3.4 we present all types of metadata found in Word documents, along with the percentages of documents that contain the metadata from `.mil` sites and from `.gov` sites. Obviously, these particular documents embed about the same amount of sensitive information and as a result they experience similar information leakage. The increased percentages in the cases of *subject* and *keywords* is

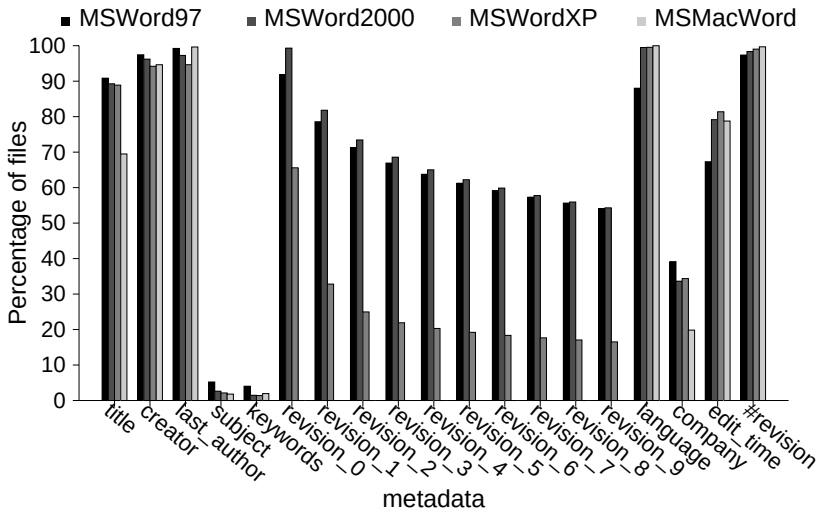


FIGURE 3.5: Existence of metadata among different versions of Microsoft Word.

apparently due to the need of taxonomy for finding documents relevant with a particular subject.

As far as military Word documents are concerned, every one in two includes *company* information, among them the more frequent are names for military departments. We found 1,500 distinct names of individuals who took part in the creation/modification of documents. All names are formatted in a similar fashion: "name.surname", e.g, "john.doe". In case of common names an ascending number is added, e.g, "john.doe1", "john.doe2", etc. Notice that the metadata of these documents reveal the scheme used in formatting usernames by this particular department. An adversary can take advantage of this information, while launching brute force attacks against SSH or other services [32].

Many companies create sample PowerPoint files which serve as *templates* for future files [61]. By inspecting our dataset we see that an initial *template* is used multiple times within a *company*. We calculate the average life time of PowerPoint files by finding the average difference between *last modification date* and *creation date*. PowerPoint files have a five times longer life-span than Word files. An interesting finding that justifies the longer life time of PowerPoint files is the following. We discovered several individuals who are the *authors* in more than one PowerPoint files. The files, in these cases, have the same *creation date* but different *last modification dates*. So, we speculate that the *authors* use the first version of the files as a seed to create new presentation files. In other words, the first PowerPoint file serves as a *template* for future presentations, and as a result these initial PowerPoint files increase the average life time of the files. Another reason that explains the long life time of PowerPoint files is that many of them, are used for lectures in university classes. We observe that specific individuals/professors create one

Metadata	% .mil	% .gov	% all
Title	83.97	85.09	86.39
Creator	89.93	88.88	92.32
Last saved by	90.58	91.90	93.08
Creation date	96.69	97.66	97.23
Last modification date	96.69	97.66	97.22
Application used	96.69	97.45	96.60
Subject	4.82	5.12	2.20
Keywords	0.76	3.29	1.54
Comments	0	0	0.0012
Template used	96.68	97.59	96.98
Format used	0	0.009	0.0011
Revision history 0	30.72	48.35	41.84
Revision history 1	25.9	39.55	30.30
Revision history 2	23.95	35.40	26.26
Revision history 3	22.43	33.22	24.24
Revision history 4	21.03	31.74	22.95
Revision history 5	21.28	30.58	21.97
Revision history 6	20.7	29.65	21.16
Revision history 7	20.20	28.87	20.42
Revision history 8	19.77	28.23	19.79
Revision history 9	19.34	27.61	19.22
Language	95.37	96.47	95.11
Company	45.02	35.26	31.90
Total editing time	75.76	72.82	77.04
Revision number	95.66	95.73	96.09

TABLE 3.4: The percentages of metadata fields in military and governmental Word documents in comparison with the total number of Word documents.

initial presentation for their classes and each year that they teach the same course, they enhance their slides with new content. Considering the above, companies and academic lecturers seem to be among the main users of PowerPoint files.

3.2.2 Digging Up Social Structures

In this Subsection we demonstrate how one can extract social structures by inspecting the authors collaborating in editing documents. We initially apply our techniques in all Excel documents collected from public web servers. In the next section we extend these techniques for studying the social graphs of Fortune-500 companies which are produced using metadata of Microsoft Word documents.

A detailed look of our collected dataset showed that a particular individual is the author in fourteen different PowerPoint documents, three different Word docu-

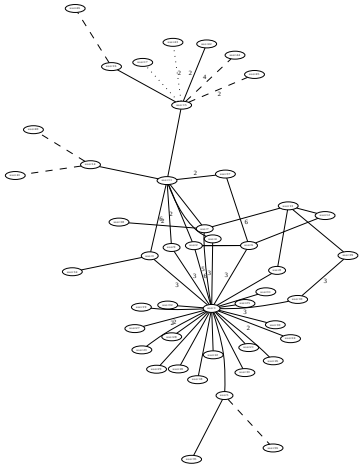


FIGURE 3.6: Clique of company A. The dotted and the dashed edges are the connections of company A with other companies.

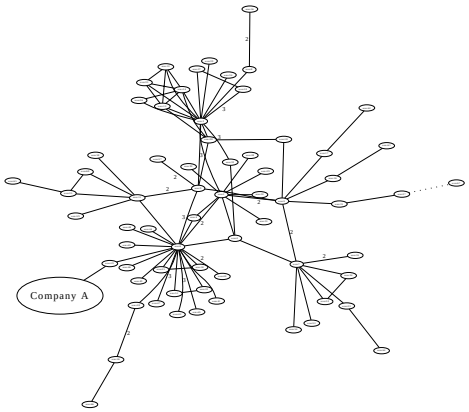


FIGURE 3.7: Clique of company B. The node labeled "Company A" represents the graph of Company A depicted in Figure 3.6.

ments and three Excel documents. In PowerPoint documents, he collaborated with seven different individuals, in Word documents with three different individuals and in Excel documents with another two. This observation lead us to investigate the possibility of extracting social structures by inspecting the metadata embedded in documents publicly available in web servers.

To conduct an initial study, we used all the Excel files of our dataset. For each document we located the metadata fields *name of creator* and *name of last author*. These fields, as it has been already stated, identify the creator and author of the document. We searched for all documents that also list these authors in the respective metadata fields. If two documents listed the same creator or author and have been downloaded by the same web server (indicated by the domain of the URL) then we considered that these authors collaborated. In this way we created graphs which have all identified authors as nodes. Each node is linked with another node if and only if these two authors are collaborating on a particular document.

In Figures 3.6 and 3.7 we show two example cliques. Note that these two example-graphs have diameter more than four and they have been constructed manually. In each graph, nodes represent authors and solid edges represent that two authors are collaborating in editing a particular document. Dashed and bold edges represent a connection where members of one clique collaborate with members of another clique. The weights on the edges indicate the number of the documents that the two authors collaborate on. If no weight is indicated on an edge, assume as being one. We proceeded and mechanically constructed 10,000 social cliques with at most four hops depth. This means that the maximum route-length connecting two individual authors, if such route exists, is of length four.

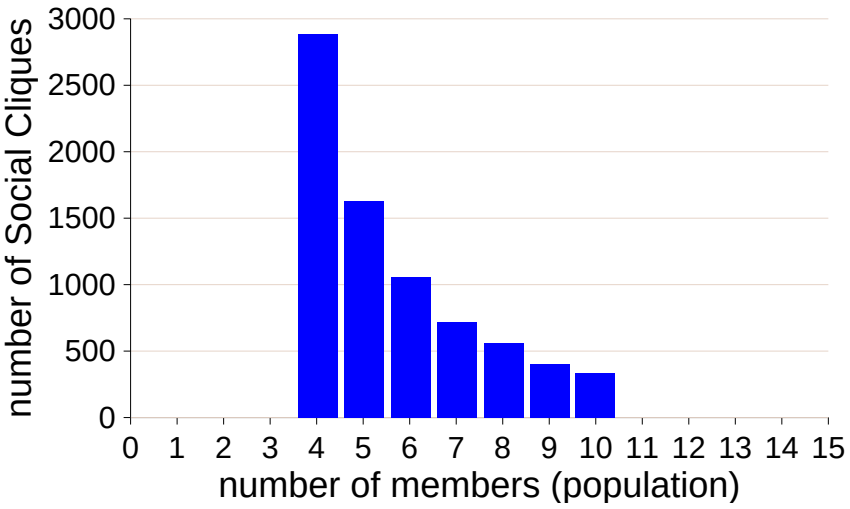


FIGURE 3.8: Distribution of the populations of social cliques. The horizontal axis shows the number of members inside a social clique, and the vertical axis indicates the number of social cliques that correspond to each population.

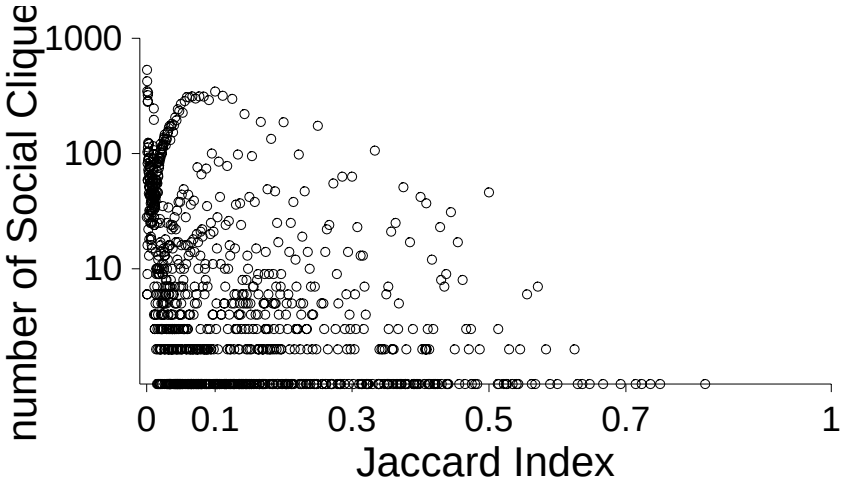


FIGURE 3.9: Distribution of Jaccard indices. The horizontal axis shows the distribution of Jaccard indices and the vertical axis indicates the number of social cliques corresponding to each Jaccard index.

The distribution of the population of the social cliques extracted is shown in Figure 3.8. There are 1,481 social cliques having more than 15 members each. We choose to exclude these groups from the graph. Only 6 social cliques consist of more than 500 members. The most populated social cliques are one with 3,886 members and another with 3,923. For each individual social clique, we seek to identify if it experiences any similarity compared to the rest of the social cliques created. We use the Jaccard similarity metric [48], which is defined as:

$$J = \frac{|A \cap B|}{|A \cup B|}. \tag{3.1}$$

The Jaccard indices of the social cliques are shown in Figure 3.9. More than 350,000 Jaccard indices are equal to 0 (not depicted in the graph). The majority of non-zero indices do not exceed 0.1. This means that the corresponding social cliques do not have more than 10 members in common. Note that many author names, as listed in the metadata fields of a document are common pseudonyms, such as “Preferred Customer”, “Valued Gateway Client”, “Unknown User”, etc.

Jaccard indices suggest that some social cliques may have common nodes. We merged social cliques with non-zero Jaccard index into larger ones. A social clique containing 78 edges merged with 70 other social cliques of different companies. The resulting clique contains 1,564 edges among 2,172 nodes, with an average degree of 1.44. The most connected node has 55 edges, meaning that the particular individual has collaborated with other 55 individuals, on writing/modifying some documents. There are 3 heavily connected nodes, 14 nodes having more or equal than 10 edges and some tens of nodes having 9 to 2 edges. The majority of the nodes have one edge, indicating that there are cliques of two individuals that collaborate together. The average clustering coefficient of this social clique is 0.013841 and there are 686 connected components. This clustering coefficient is very similar to systems that experience social properties. For example, Gnutella, a file sharing system, experiences a clustering coefficient of about 0.012 [68]. In Section 3.2.3 we produce a quite higher clustering coefficient.

3.2.3 Fortune-500 Companies

We applied the techniques outlined in Section 3.2.2 in documents associated with high profile companies. We did this for two reasons. First, we seek to identify if major companies indeed expose sensitive information via documents’ metadata. For example, if a social graph of a high profile company can be exposed, then an attacker can send a malicious document to the most highly-connected nodes of the graph. In this way, the adversary increases the probability for the document to be delivered to many more employees of the victim company. Second, our intuition is that the large companies may collaborate with each other. If this is the case, we want to see if this collaboration can be exposed by studying the metadata associated with the documents in our dataset.

We used the Fortune-500 company sites of 2010 as listed in CNN.com [3]. We selected and extracted from our original dataset all Word documents associated with these companies. A targeted crawling for a specific company, would clearly give us more extensive results, however this was not done as part of this paper and is planned as future work.

For each of the Fortune-500 company sites, we first gathered all Word files that were downloaded from the company's web server, indicated by the domain of the URL. For each document in the set, we located the metadata fields *name of creator*, *name of last author* and *revision history*. The *revision history* fields have the following format: Author 'name' worked on 'computer's location' (e.g., `Author 'User' worked on 'C:\My Documents\confidential.doc'`). If two documents listed the same name in one or more aforementioned fields we assumed that these authors collaborate. Note, that although the queries we used for collecting our dataset were not targeted towards any particular company web server, we managed to extract a set of 79 cliques out of the Fortune-500 companies. Notice, that the amount of privacy leakage exposed in this set is not the maximum. An adversary could potentially target the site of a particular company to achieve optimal results, by downloading a very precise set of documents.

Each created clique consists of more than two nodes. The average number of nodes is $\sim$29 nodes per clique and the average degree is $\sim$1.08 edges per node. The low average degree per node suggests that cliques are not strongly connected. The most populated clique contains 860 nodes, 899 edges and 246 connected components, and belongs to a leading producer of computer software. The largest connected component of this clique is depicted in Figure 3.11[4]. Nodes correspond to a company employees and edges to social or person-to-person relationships among employees of the particular company. Note, that all graphs are anonymized for privacy reasons. Overall, 50 out of the 79 cliques contain more than one connected components.

It is interesting to identify whether the metadata graphs depict social networks or random graphs. We considered all the strongly connected components that consist of more than 4 nodes, to examine their properties. The average clique degree is $\sim$3 and the average diameter is $\sim$3 with $\sim$13 nodes and $\sim$20.5 edges per clique, on average. Also, they have a very high average clustering coefficient equal to $\sim$0.54. These values are typical for real-world, popular social networks, such as Flickr and Orkut [58].

Apart from the social structure, we were also interested to see the document distribution among the authors of each company and find the most frequent document publishers. This is of interest because an individual may be the author in many documents, but they may not be part of the created graph because they never collaborate with anyone else. By creating the document distribution (an example is shown in Figure 3.10), we can extract a broader set of employees and the number

[3] http://money.cnn.com/magazines/fortune/fortune500/

[4] You can find more social cliques graphs at: `www.ics.forth.gr/~gessiou`

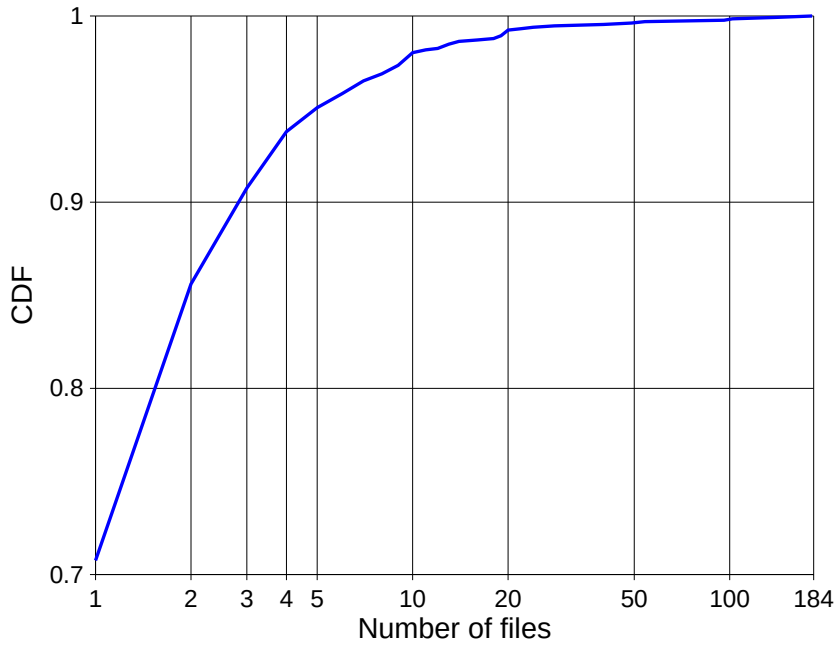


FIGURE 3.10: CDF of document distribution in the most populated clique, which consists of 860 nodes and 899 edges. The majority of the nodes has participated only in one document. There are some nodes that have worked on some tens, even hundreds of documents.

of documents they have worked on. An adversary, maybe belonging to a competing company, whose goal is to obtain the first company's sensitive data, could create such a distribution. The document distribution would help the adversary, to find which employee has created the most documents and attack their computer system. With high probability this person would have many documents and thus more information about the company in their possession. Moreover, knowing the victim's name, the attacker could use it for guessing the victim's username, or even their password using a brute-force attack [58].

Examples of Fortune-500 company whole graphs are depicted in Figure 3.12 and Figure 3.13. More specifically, Figure 3.13 presents the graph of one leading producer of personal computer and related equipment. In the figure, node #2 has the highest degree, betweenness and closeness centrality. An interesting fact is that we were not able to find any information in this company's web site about the individual represented by node #2. A more in-depth examination shows that this node participates in the graph because it is present in the *revision history* fields. This leads us to conclude that they are a company contractor or collaborator, rather than an employee. Thus, this case suggests that *revision history* fields could disclose collaborations between two companies. The initiator-company creates a document

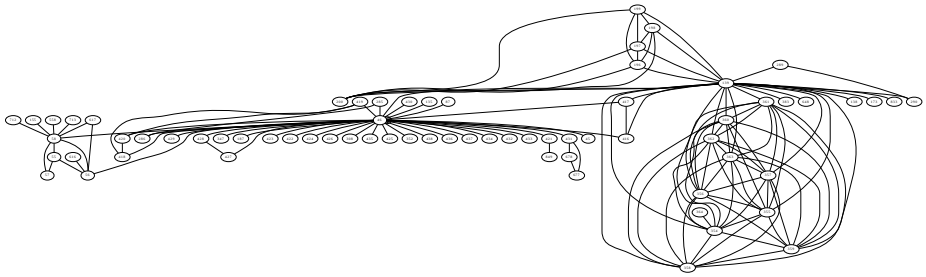


FIGURE 3.11: Example of a connected component which is part of the most populated clique. The nodes are denoted in the graph with increasing numbers. It consists of 70 nodes, 139 edges and has an average degree equal to 3.9714. The node with the highest betweenness(=0.84) and closeness(=0.66) is #46, while #139 node has the highest degree centrality(=25).

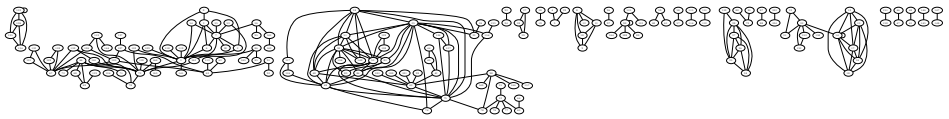


FIGURE 3.12: Example of a populated graph which consists of 167 nodes, 228 edges and 24 strongly connected components.

(*creator*), this document is then modified from both sides (*revision history*), and finally returns to the owner for inspection (*last author*). Finally, the owner has the right to upload the document to their server, where it can be downloaded by anyone.

3.2.4 Identifying Users in Social Networks

In this Subsection we seek to identify if we can efficiently fingerprint users [69] that collaborate in the production of documents by locating them in popular social networks, such as Twitter. We try to match the cliques we have already identified with users following and followed in Twitter.

First, we adjust all identified cliques by filtering out the most frequently occurring names in the documents' metadata. All the 10,000 identified social cliques include 124,779 names in total, from which 51,709 of them are unique. For the rest of our experiments we exclude the 27 most frequently appearing names, such "Preferred Customer" and names that do not contain at least 2 words of at least 2 letters (we want a full name and not just a pseudonym). Also, we do not include names that contain generic words such as "bureau", "department", "service", "city", "user", "customer", "administrator", "school", "student" and "staff", as they are popular pseudonyms selected by different organizations and thus they di-

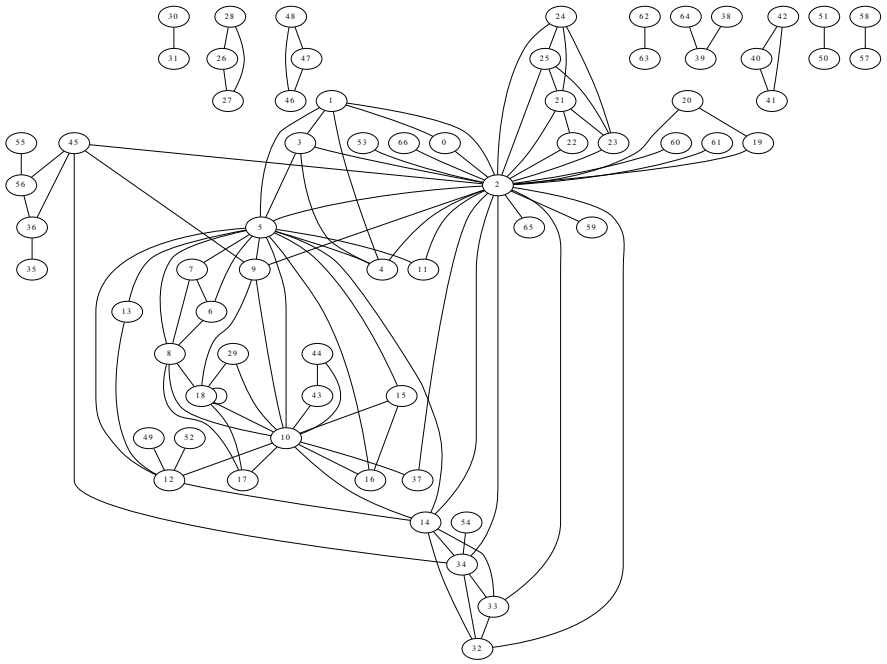


FIGURE 3.13: Example of a populated graph which consists of 67 nodes, 108 edges and 9 strongly connected components. The largest component, as it being depicted, has 47 nodes, 93 edges and an average degree of 3.9574. Node #2 has the highest degree(=26), betweenness(=0.6) and closeness centrality(=0.62).

lute the results. The experiments and results that are described below use full names of people that wrote/modified at least 9 files and at most 47 files. We search all these particular individuals at Twitter and find their followers and those they follow. We seek to extract correlation that would verify that people collaborating in the editing of a particular document can be identified in Twitter.

Overall, we examine 575 cliques, containing at total 14,969 people. We find that 1,911 people among them own a Twitter account. We manage to find 115 social cliques that a subset of their members correlate with each other through Twitter. People in these social cliques seem to have common friends in Twitter. We also find one case that 2 out of 3 individuals belonging to a social clique are friends and also have 39 friends in common. In another case, 2 people out of 19 in initial social clique are friends and moreover have 4 friends in common. There are also 3 social cliques, containing 131, 500 and 297 individuals each, that their members have common friends in Twitter and moreover there is a couple of individuals in each clique that connects to each other with direct friendship in Twitter. We also find that 2 people follow the group of the company that they work in, based on the URL in our dataset. This fact verifies that we can correctly match the identity derived from the metadata and the one registered in Twitter.

Fortune 500 Companies

The same procedure is conducted for the cliques created by the Fortune 500 companies, which is described in Section 3.2.3. We use the most populated cliques, 12 in number, that correspond to major companies. These 12 cliques consist of 1,561 unique names. We exclude names that contain generic words such as "company", "employee", "computer", etc. Finally, 1,508 unique full names are checked in Twitter for examining if they own an account. Out of them, 798 individuals seem to have a Twitter account. Via Twitter API we get all their friends and followers, and check for direct and indirect connections between them. In order to avoid false positives, we exclude from the procedure accounts that are known to come from very popular Twitter users [51]. The results show that in the most populated clique, which consist of 860 nodes, there are 1,843 indirect and 11 direct connections. In other words, there are 11 pairs of people that both belong to the clique and are friends at Twitter, and moreover there are 1,843 more pairs that have 1 common friend. The rest of the cliques contain 6 to 318 indirect connections. An indirect connection defines an implicit friendship between two Twitter users that share common friends, but they are not directly connected. Users that share a significant amount of common friends have high probability of being also friends. Last but not least, three cliques contain one, two and three direct connections. A direct connection defines an explicit friendship between two Twitter users, where one is following the other.

4

Defenses

Improperly disclosed PII, either from Deep Web or Surface Web, is often impractical or impossible to retrace and recall from unintended recipients. Thus, protective measures to prevent undesirable disclosures are critical. In this Chapter, first we present some practical defenses against the breaches described in Sections 3.1 and 3.2.

Next, in Section 4.3 we describe comprehensively an information retrieval based method for information leak detection, called IRILD. The traditional approach for detecting information leaks is to generate fingerprints of sensitive data, by partitioning and hashing it, and then comparing these fingerprints against outgoing documents. Unfortunately, this approach incurs a high computation cost as every part of document needs to be checked. As a result, it is not applicable to systems with a large number of documents that need to be protected. Additionally, the approach is prone to false positives if the fingerprints are common phrases. In this work, we propose an improvement for this approach to offer a much faster processing time with less false positives. The core idea of our solution is to eliminate common phrases and non-sensitive phrases from the fingerprinting process. Non-sensitive phrases are identified by looking at available public documents of the organization that we want to protect from information leaks and common phrases are identified with the help of a search engine. In this way, our solution both *accelerates* leak detection and *increases* the accuracy of the result. Experiments were conducted on real-world data to prove the efficiency and effectiveness of the proposed solution.

4.1 Database Content as Deep Web

In Section 3.1, we described and use several Web sites' forms that reveal Deep Web content exposing privacy leaks. To address these issues presented in Section 3.1, there are a number of steps one can take:

- Modify the AMKA web form as to always also require a person's Taxpayer ID or National ID for authentication purposes, not only in case of errors as is now. This simple solution should eliminate 100% of the AMKAs found in our study, as fas as public figures concerned.

- In case that an individual does have an AMKA, the site should just inform him that he does, and urge him to the proper authority in order to get it.

- At the site which is provided by the Ministry Of Interior Administration and Decentralization, the users should be asked to enter all their personal information, including the full names of parents. This way, we would not be able, to use it for extracting mothers' names.

- Provide a way for citizens to be taken off this online look-up service, with proper safe-guards to avoid DOS-style attack.

- Educate people to not post sensitive information about themselves or others on the web.

In general, this work presents a broader privacy issue, broader than the frontiers of Greece. In a world of emerging technology, many governments all over the world, try to keep up to date and facilitate their citizens by offering them online services. But, which is the tradeoff between the personal information privacy and the convenience of the user, in such services?

In online services that there are forms requesting user's personal information, the site should request the more available information from the user. We believe that it is not difficult for a user to enter all his personal information given the convenience that is being offered by the service.

Generally, the presentation of sensitive information should be avoided by such sites, if it is not necessary. For example, the output of the web form of AMKA should be an encouragement to the user to go and get his AMKA from the proper authority, and not the AMKA itself.

All in all, the physical presence of an individual and a certified picture of him, such a National ID Number, seems to be the most secure document in proof.

4.2 Countermeasures for Metadata Information Leaks as Deep Web Content

Throughout Section 3.2, we have highlighted various privacy risks stemming from the exposure of information stored in metadata associated with documents. We will briefly now discuss techniques for reducing the risk and the privacy leakage.

First, the sanitization techniques offered by various tools for extracting and scrubbing metadata can significantly reduce metadata leakage. There are quite few free as well as commercial tools, such as [2, 17, 21]. These tools support a wide variety of file formats and can automatically eliminate all metadata information stored in documents. However, file sanitization has some drawbacks. Metadata have many legitimate uses for sorting, categorizing and indexing user files. The existence of metadata is fundamental for the operation of these tools. Eliminating all metadata will make all these tools non-functional.

Our initial dataset contained approximately five million PDF documents. Our metadata analysis revealed that PDF documents contain dramatically less metadata information than all other formats. For example, PDFs do not contain *revision history* in the format that MS Office documents contain. Thus, one can convert Microsoft Office documents to PDFs. However, using PDF is sometimes hard for collaborative editing. Also, in cases that it is suitable, the usage of RTF files, instead of Word documents, can significantly reduce the leakage. In our initial dataset, there were some documents that had `.doc` extension but were actually RTF files. We noticed that none of them contained any metadata. However, RTF files support a limited set of text decoration and customization.

Finally, a good practice is to carefully review all configuration files associated with web servers and either prevent directory listing in folders hosting sensitive documents, or offer to server only files that are already sanitized.

4.3 IRILD: an Information Retrieval based method for Information Leak Detection

4.3.1 Background

A popular approach to detect information leaking from a confidential document is to employ cyclical hashing to split the document into multiple parts and generate fingerprints for these parts. In particular, given a document, the method repeatedly creates fingerprints for strings of C characters from the start to the end of the document, offset by O characters each time (C and O are predefined parameters). An example is shown in Figure 4.1, where C and O are set to 30 and 10 respectively. In this example, 30 characters from the 1^{st} to the 30^{th} positions are used to generate fingerprint1, 30 characters from the 10^{th} to the 40^{th} positions are used to generate fingerprint2, 30 characters from the 20^{th} to the 50^{th} positions are used to generate fingerprint3, and so on.

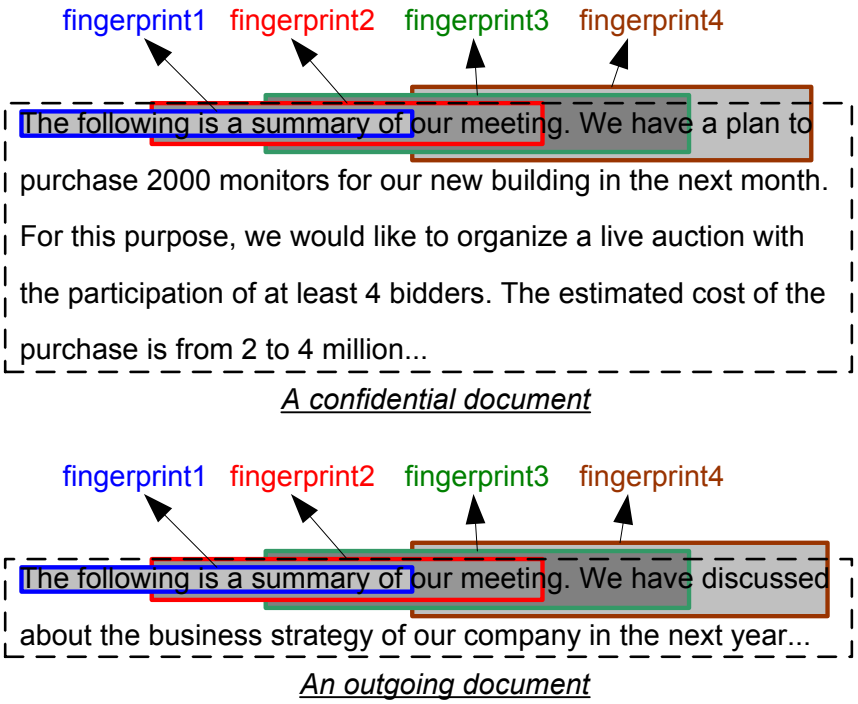


FIGURE 4.1: An example of using cyclical hashing

Given an outgoing traffic channel (e.g., an outgoing email or a file uploading to an outside server), cyclical hashing is also employed to generate a set of fingerprints for the traffic. These fingerprints are then checked against those previously extracted from confidential documents to detect information leaks. In the example in Figure 4.1, since the first three fingerprints of the outgoing document match the first three fingerprints of the confidential document, it is considered to have partial information leak.

A problem with this approach, illustrated in Figure 4.1, is that it introduces false positives when common phrases or sentences are used. In this example, even though the common sentence "the following is a summary of our meeting" appears in both the confidential document and the outgoing document, since it does not convey sensitive information, there is actually no information leakage. Furthermore, this redundant check incurs a high processing cost, and hence the approach fails to work in systems with a large number of sensitive documents.

4.3.2 IRILD

To avoid false positives involving common phrases as shown in the example of Figure 4.1 and also reduce the unnecessary cost of generating and checking fingerprints of the common phrases, we propose IRILD, an information retrieval based

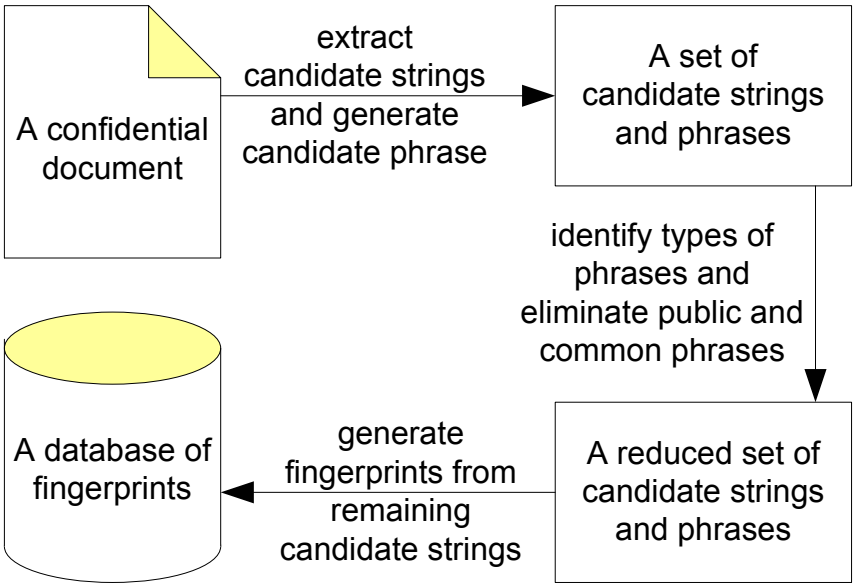


FIGURE 4.2: An overview of fingerprint generation

method that is able to identify common phrases and eliminate them from the fingerprinting process. In particular, we evaluate the popularity of phrases by submitting them to an Internet search engine such as Google and measure the number of returned results. The higher the number of returned results of a phrase is, the more common the phrase is. For example, since there are approximately 15,300 results returned from Google when searching for the sentence "the following is a summary of our meeting", the sentence is considered as a common sentence and no fingerprint should be generated for it. Furthermore, assume that the organization, which employs IRILD also maintains public documents (e.g., documents in public folders of the company's web site). IRILD will also eliminate phrases that can be found in those public documents from the fingerprinting process because these phrases contain already known information.

IRILD generates fingerprints for confidential documents in three steps. In the first step, similar to the popular approach introduced in Section 4.3.1, IRILD employs cyclical hashing on each confidential document to generate a set of candidate strings for the fingerprinting process. Note that by using a fixed number of characters to generate strings, a candidate string may not be a complete phrase (e.g., as in the example of Figure 4.1, the second candidate string, "ing is a summary of our meetin" is not a complete phrase). Thus, from this set of candidate strings, IRILD needs to generate a set of candidate, complete phrases. Each phrase corresponds to a candidate string and is the shortest phrase that totally covers the string. For example, the candidate phrase corresponding to the second candidate string in the example of Figure 4.1 is "following is a summary of our meeting". In the second

step, IRILD identifies public phrases and common phrases and removes them from the set of candidate phrases. Finally, candidate strings associated with remaining sensitive candidate phrases are used to generate fingerprints. The overview of IRILD's processing steps to generate fingerprints for a confidential document is illustrated in Figure 4.2.

Note that while the fingerprint generation of IRILD is different from that of the popular approach, the information leak detection of these two approaches is still the same, i.e., fingerprints of confidential documents in the database are used to check against fingerprints of outgoing documents for information leak detection.

Public phrase identification

The task of identifying public phrases from a set of candidate phrases is simply a search of these phrases from available public documents. To fulfill the task, a solution is to employ the basic information retrieval technique to create document indices for public documents and inverted indices for words in public documents. Each document index records words that appear in a public document. On the other hand, each inverted index records positions of a word in all documents the word appears. For example, the structure of a document index can be $\{doc_x$: $word_1$, $word_2, word_3, \cdots\}$ while the structure of an inverted index can be $\{word_y$: $[doc_1$: $pos_{11}, pos_{12}, pos_{13}, \cdots]$, $[doc_2$: $pos_{21}, pos_{22}, pos_{23}, \cdots]$, $\cdots\}$. To detect whether a phrase is a public phrase, we first parse the phrase into a list of words. After that, we search document indices to see if there is any document that contains all words in the list. If such a document exists, we then retrieve inverted indices of the words to see their positions in the document. If the words appear at adjacent positions, it forms a phrase in the document. In this case, we conclude the checking phrase is a public phrase since the checking phrase matches a phrase in a public document. While this solution always generates exact results without false positives, it incurs a high cost in search. As a result, this technique can only be used if the number of public documents is not very large. An alternative solution is to also employ cyclical hashing to generate fingerprints for public documents and compare these fingerprints to the fingerprints of the candidate strings to check if the candidate strings exist in public documents. This technique is often used if the number of public documents is large.

Common phrase identification

As previously discussed, to check whether a phrase is a common phrase, we submit the phrase to Google and measure the number of returned results. Basically, a phrase is a common phrase if there are a lot of returned results. As discussed before if the phrase can be found easily by the search engine, with high probability, it does not contain secret information. On the other hand, if the search engine returns only a few results for the search, the phrase is considered as a sensitive phrase. To make it easy to evaluate the popularity of a phrase, we propose a formula to calculate

the popularity score of the phrase from its number of returned results by the search engine as follows:

$$Score(P) = \log_{10}(N(P) + 1) \tag{4.1}$$

where P is the evaluating phrase and $N(P)$ is the number of results returned from searching P. A common phrase is a phrase whose popularity score is greater than a predefined threshold K. For example, if we set K to 4 and the search of a phrase P has 20,000 number of results, P is considered as a common phrase because $Score(P) = \log_{10}(20,000 + 1) > 4$.

The above technique to identify common phrases usually works if candidate phrases are not long (e.g., common phrases are single phrases). However, in cases where candidate phrases are long (e.g., when C is set to a large value, the candidate phrases that cover candidate strings are also long), a problem comes. In these cases, candidate phrases can be a combined phrase, a sentence, or even some sentences or a paragraph. While it is easy to identify the popularity of a single phrase by the search engine, it is more difficult to do the same thing for a combined long phrase or sentence. It is because with high probability, the long phrase cannot be found by the search engine. To deal with this problem, we suggest a simple way to parse the combined phrase or sentence into smaller single phrases and calculate the popularity of the combined phrase from the popularity of the split single phrases. In particular, let P_1, P_2, $\cdots$, P_n be n single phrases that are split from a combined phrase P and $Score(P_1)$, $Score(P_2)$, $\cdots$, $Score(P_n)$ be their popularity scores. $Score(P)$ is calculated as:

$$Score(P) = \min_{i=1..n} \{Score(P_i)\} \tag{4.2}$$

The rationale behind the above formula is that the score of the combined phrase should be equal to the minimum popularity score of its members. The intuition of the formula is straightforward. A combined phrase is a common phrase if all of its sub-phrases are common. On the other hand, a combined phrase is a sensitive phrase if at least one of its split phrases is a sensitive phrase.

Note that a concern with this approach is that sensitive information can be leaked if a determined adversary can capture all queries submitted to the search engine and reconstruct indexing documents from these queries. Our solution to this concern is to submit queries from different locations to hide the fact that all queries coming from the same source, and hence the attack should fail. In our approach, we simply used PlanetLab [19] to distribute queries before submitting them to the search engine. Alternatively, search proxies [1, 23] can be used to anonymize queries.

Improvement techniques

Besides the basic solution to identify public phrase, we suggest two improvement techniques as follows:

- We observe that if a phrase contains numbers, it is often a sensitive phrase. It is because in most cases numbers represent sensitive information, such as

TABLE 4.1: Experimental settings

Parameter	Domain values	Default value
C	15 - 30	20
O	5 - 20	10
K	3 - 6	4

telephone numbers, dates of birth, amounts of money, etc. As a result, we decided to consider all phrases containing numbers as sensitive phrases. In this way, we save time by not querying the Google for these phrases. In other words, prior to submitting a phrase to Google, whether numbers are contained in the phrase. If there are, the phrase is considered as sensitive eliminating the neet to perform a Google search.

- We propose that if we have two adjacent phases that are overlapped by at least half of the length (this happens when $O < \frac{1}{2}C$ or in other words the length of the offset is less than half of the length of the candidate string), and one of these phrases is a popular phrase with a high score, we can also skip the Google search for the other. The reason is because with the significant overlapping between these two phrases, with high probability, the other phrase will not contain sensitive information.

Note that while these two improvement techniques help to further improve the processing speed, they may introduce false positives in some cases. Nevertheless, this very low false positive rate is tolerable when compared against the total accuracy.

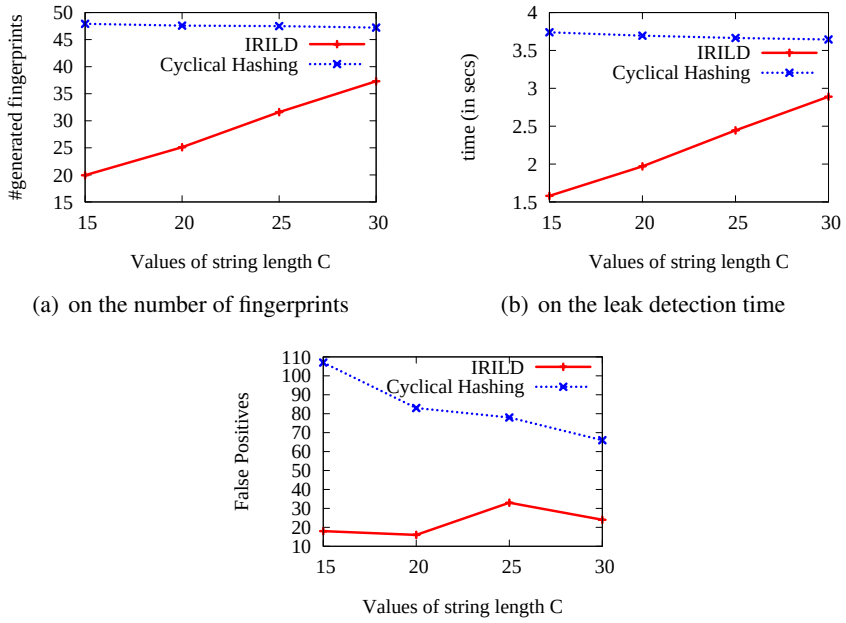
4.3.3 Experimental Study

To evaluate the efficiency and effectiveness of IRILD, we implemented it in Python 2.5 and conducted experiments with the Enron Email Dataset [4], where we randomly chose 50 emails from the "Inbox" of 10 employees as confidential documents, and used 200 random emails from their "Sent_Items" to test the accuracy of the method. We considered textual information in the Enron's website (an instance of January 2001 [15]) as public information. For comparison purpose, we compared IRILD to the popular approach that employed only cyclical hashing without the removal of public and common phrases in fingerprint generation.

In the experiments, we set the default value of C to 20 because in our opinion the value of C should be equal to the average length of the queries made in Google. Note that since the average query in Google consists of 4 words [1] and the average length of an English word is 5 characters [2], setting C equals to 20 seems quite

[1] http://www.beussery.com/blog/index.php/2008/02/google-average-number-of-words-per-query-have-increased/

[2] http://blogamundo.net/lab/wordlengths/



(a) on the number of fingerprints

(b) on the leak detection time

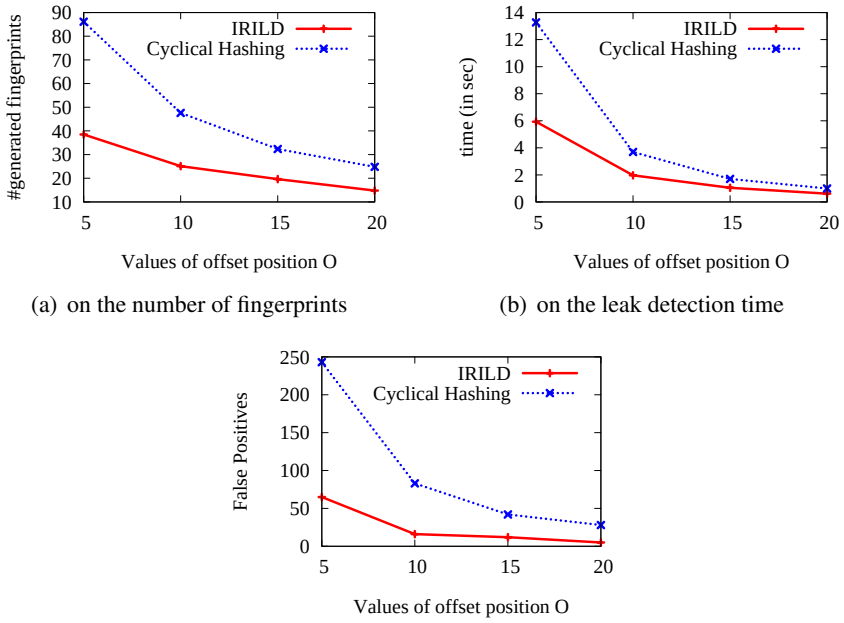
(c) on the accuracy

FIGURE 4.3: Effect of varying the string length C, keeping the offset position O = 10

reasonable. We set the default value of O to 10 (which is half of C) because smaller values of O would create more fingerprints and thus the number of false positives would be increased. On the other hand, higher values of O would lead to the decrement of phrases that would be tested for information leaks, in other words there may be potential false negatives. Finally, we choose 4 as the default value of K because we believe that a phrase should be tagged as common if its occurrences at Google are beyond 10,000, in number. To summarize, the default and range of values of C, O, and K used in experiments are listed in Table 4.1.

We evaluated the performance of IRILD and the popular approach in three aspects: (i) the cost of fingerprint generation in terms of processing time and the number of fingerprints, (ii) the cost of information leak detection in terms of processing time, and (iii) the accuracy of the results in terms of false positives and false negatives.

Note that to compute the accuracy of IRILD and cyclical hashing, we manually looked at testing documents to extract all possible leaked cases (i.e., information that is copied from confidential documents).



(a) on the number of fingerprints

(b) on the leak detection time

(c) on the accuracy

FIGURE 4.4: Effect of varying the offset position O, keeping the string length C = 20

Cost of fingerprint generation

We first evaluated the cost of fingerprint generation for confidential documents (emails in our case). As expected, IRILD incurred a processing time around 2 to 3 times longer than the basic approach, due to the extra time of submitting queries to Google to determine the popularity of candidate phrases. While this extra time can be reduced by employing PlanetLab [19] to submit queries concurrently, since fingerprint generation is done offline whenever a confidential document is submitted to the system, it does not affect the efficiency and effectiveness of IRILD.

The average number of fingerprints generated by IRILD and the basic approach when varying C and O are shown in Figure 4.3(a) and Figure 4.4(a) respectively. The results show that by removing public and common phrases from fingerprint generation, IRLID significantly reduced the number of generated fingerprints compared to the basic approach. In particular, as shown in Figure 4.3(a), the percentage difference in the number of generated fingerprints between cyclical hashing and IRILD varied from 21% when C = 30 to 58% when C = 15. On the other hand, as in Figure 4.4(a), IRILD generated 55% less fingerprints, at best case, that O = 5. Even at the worst case, when O = 15, IRILD manages to generate 39% less fingerprints than cyclical hashing. It is interesting to observe that while the number of generated fingerprints in cyclical hashing does not depend on the value of C,

in case of IRILD, the bigger the value of C is, the more the number of generated fingerprints is.

Cost of information leak detection

In this experiment, we measured the processing time required to detect information leak from testing documents. Figure 4.3(b) and Figure 4.4(b) show the processing time of information leak detection when varying C and O. As expected, since the number of fingerprints in IRILD was less than those in the basic approach, IRILD took a smaller number of comparisons, and hence it incurred a faster processing time. Actually, if we make a comparison between Figure 4.3(a) and Figure 4.3(b) as well as Figure 4.4(a) and Figure 4.4(b), the similarity between them show a strong correlation between the number of the generated fingerprints and the information leak detection time.

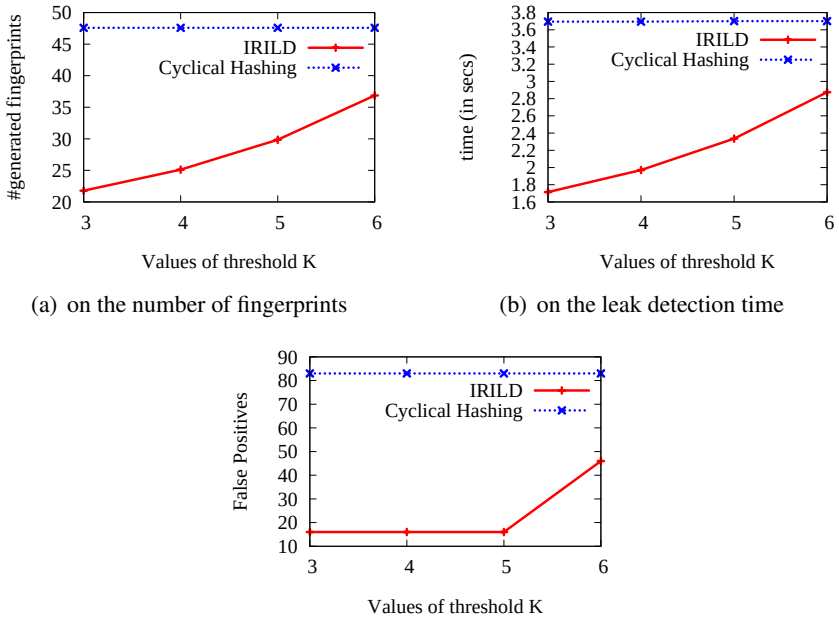
Accuracy

As discussed earlier, we evaluated the accuracy of IRILD by measuring false positives and false negatives. It is interesting to observe that we get the same number of false negatives in both IRILD and the basic approach. In particular, both methods have no false negatives when O ¡ 20 and only one false negative when O = 20. That is because both approaches employ the same technique for indexing sensitive information and searching for information leaks. However, in terms of false positives, the result of IRILD is much better than that of the basic approach. As shown in Figure 4.3(c) and Figure 4.4(c), IRILD achieved a much better accuracy compared to the basic approach.

It it important to note that in practice we would always set O to small values (e.g., $O < \frac{1}{2}C$) in order to avoid false negatives (i.e., all leaks should be detected). In this case, IRILD significantly outperforms the basic approach since according to this experiment and the previous two experiments in Sections **??** and 4.3.3, the smaller the value of O is, the bigger the improvement IRILD has compared to the basic approach in terms of fingerprint generation cost, leak detection cost, and accuracy (or false positive cost).

Effect of varying the threshold K

So far we had K fixed at 4. In this experiment, we evaluated the effect of varying K from 3 to 6 on IRILD (note that the change of K does not affect the popular approach). The experimental results displayed in Figure 4.5 show that with the increasing of K, the number of generated fingerprints as well as the processing cost increased. It is because when K increased, we put a higher boundary for phrases to be considered common phrases, and hence less common phrases were identified and removed. The consequence of having less identified common phrases was an increase in the false positives of the method (happened when K = 6). Note that in



(a) on the number of fingerprints

(b) on the leak detection time

(c) on the accuracy

FIGURE 4.5: Effect of varying the threshold K, keeping the string length $C = 20$ and the offset position $O = 10$

the worst case if we set K to infinity so that all sentences are considered sensitive in IRILD, both IRILD and the basic approach will get the same number of false positives. Nevertheless, as discussed before, a reasonable value of K should not be high (e.g., 4 or at most 5).

5
Related Work

In the last decade there are many papers [28, 29, 55, 70], indicating the value of Deep Web and presenting methods on how we can access the hidden information. But, the term of "Deep Web" was firstly introduced by Bergman [33].

The first paper that highlights the risks due to metadata in documents found on the Internet is [34]. Despite Byers et al. counting the hidden words in in a few thousand documents, they do not take into account all available kinds of metadata and their sample is much smaller than the one we used in Section 3.2. A new tool which finds personally identifiable information that may be stored in documents is introduced in [30]. LeakHunter tries to solve the problems that metadata may cause to companies and individuals, similarly to the ones we have highlighted in Section 3.2. Ideas presented in Section 3.2 are similar with the work presented in [27]. However, they focus in metadata collected by the Operating System's filesystem. The filesystem's metadata do not normally contain personal information. This is why the authors in [27] focus more on temporal changes and not in privacy issues.

In the context of privacy risks due to metadata, [26] presents several incidents that demonstrate a series of security breaches and sensitive information disclosures that have recently become a serious threat to many organizations around the world. Among other findings [22] indicates that business users in Asia are unaware of the risk of metadata. Similarly, the authors of [40] support that the overall amount of metadata associated with documents is increasing. Their assessment and results, suggest that a more detailed analysis of metadata may reveal more associations between individuals, e.g. the existence of social networks; a fact that our study confirms.

In another context, the authors of [37] take advantage of the space that metadata and generic unwanted data take up in a document, and utilize this space for steganography.

Symantec [13] shows that the majority of malicious Trojans exploiting file formats in 2007 was primarily in Word documents (67%), PowerPoint files (17%), SpreadSheet files (3%) and PDF documents (3%). This observation, and the fact that these file formats are considered to be widely used from our experience, led us to select these particular formats for our study. Many real-life and potential incidents concerning hidden data in Word, Excel, PowerPoint and PDF documents, are presented in the 13th chapter of [66]. The problems, that the revision history in Word files can cause, are the first to be mentioned. Overall, although much work has been done to identify and to remove sensitive information from documents, our study is the first that quantifies the amount of this information.

The authors of [62] developed the PRIX (PPT Residual Information eXtractor) tool. Its aim is to identify the residual information in PowerPoint documents. Residual information is created when the option "allow fast saves" is selected. In a followup work [61], apart from text residual information, PRIIX (PowerPOint Residual Information & Identifiers eXtractor) extracts slide and object identifiers, too. Data concealment and detection in Microsoft Office 2007 files that use Office Open XML (OOXML) as their basis is studied in [60]. The paper starts by proving that someone can indeed hide data in such files and it also presents algorithms for finding hidden data in these. The retrieval of any object or text previously deleted or modified, from the creation time of the document to its most recent version, is attempted in [38] for PDF documents.

There is a considerable amount of previous work in the field of extraction and analysis of social networks. P. Mika presents Flink [57], which constructs and visualizes social networks of semantic web researchers by using information from sources such as web pages, emails, publication archives and FOAF profiles. Polyphonet [56] presents a series of methods for obtaining a social network using a web search engine and used in order to enhance scalability. Polyphonet is the implementation of the algorithms, that are enabled at Japan Society of Artificial Intelligence conferences over three years and at the UbiComp conference. Recently, some steps towards characterizing social networks emerged from e-mail exchange have been done, such as [50] that presents behavioral profiles of it and how the augmentation of contact lists may be succeed, through adding contacts-of-contacts.

In the past few years, a number of white papers have been written discussing different aspects of information leak prevention in general, and how to detect information leak in particular. These include [46], which introduces basic solutions to detect and prevent information leaks, [52], which presents testing and evaluation standards for information leak prevention products and [47], which studies the cost incurred by information leaks in practice.

In general, there are two main approaches to information leak detection. One is based on defining sensitive expressions, keywords or phrases. This way, information leaks are detected if the outgoing traffic contains the specified expressions,

keywords or phrases. The other is based on fingerprints of information. For example, a popular approach for information leak detection in documents is to divide them into multiple parts and generate fingerprints of these parts. These fingerprints are checked against fingerprints of similar divided parts of outgoing traffic for leak detection.

A special type of information leak is information leak from applications. To deal with this type of leak, [49] proposes Privacy Oracle, a solution that tests an application with different inputs and maps input perturbations to output perturbations to detect potential information leaks. Alternatively, [35] introduces the use of shadow execution that runs two copies of an application at the same time in which, the one containing personal information is kept away from accessing the network while the other with non-confidential data is used to communicate over the network. The response from the network is then shared for both copies. These solutions are, in fact, complementary to these basic solutions as well as the solution presented in Section 4.3.

In addition to information leak detection, there exists a class of techniques that address access control to prevent information leak. Access control is required in cases where the information is available to someone but should be restricted from others. With respect to this aspect, [36] introduces a solution to avoid information leak caused by accidentally sending emails to unintended recipients. The basic idea of this solution is to measure the similarity between the current outgoing email and previous outgoing emails of the same recipient. If there is a big difference between them, the current outgoing email may contain information leak. On the other hand, [63] presents CLAMP, an architecture that protects confidential information in web servers by enforcing strong access control on user data while isolating code running on behalf of different users. These access control techniques are orthogonal to the basic solutions as well as the solution presented in Section 4.3..

The use of search engines to detect information leaks has been introduced in [39]. However, in our work, the purpose of using search engines is simply to detect inferences between keywords. The purpose of finding inferences between keywords is to discover sensitive documents that do not contain specified sensitive keywords but contain closely associated keywords. It is because with high probability, these sensitive documents also contain confidential data. With respect to inference detection, prior to this work, web based inference has been significantly studied in a number of papers such as [41], [59], [67].

6
Conclusions

In this thesis, we presented an approach to explore several aspects of the Deep Web concerning Personal Identifiable Information (PII). Especially, we performed two immense privacy case studies that expose Personal Identifiable Information inside the Deep Web.

First, we presented the privacy issues that have arisen with the introduction of the Greek Social Security Number (AMKA). The privacy concerns stem from the fact that possibly malicious parties can extract Greek citizen AMKAs from the online portal. We also conducted a more general study on the availability of personally identifiable information on Greek sites in contrast to US sites and found that there is a lot of it publicly available.

Next, we presented an in-depth analysis of the metadata hidden inside over 15 million of documents, which we obtained via public web servers. We highlighted a series of privacy risks involved in sharing documents that carry sensitive information in their metadata. Additionally, we showed that it is possible to extract social cliques of users that collaborate in the production of documents by simply correlating the author fields found in the metadata of documents. We were able to escalate our attack on privacy by successfully identifying some of these cliques on Twitter. This allows us to cross-correlate the public activities of someone on Twitter with their private activities, like their contribution in the editing of a particular document. Our study raises major concerns about the risks involved in privacy leakage, due to metadata embedded in documents that are stored in public web servers.

We also presented solutions for each of the afore-mentioned privacy case studies. Finally, in this thesis, we introduced IRILD, an information retrieval based solution to improve the performance of the traditional cyclical hashing approach for information leak detection. The core idea of IRILD is to *identify and remove* public phrases (found in public documents) and common phrases (identified by

checking the number of results returned by Google when querying the phrases) from the fingerprinting process, since these types of phrases do not contain sensitive information. Furthermore, we conducted extensive experimental evaluation of our solution, and proved that it significantly outperformed the cyclical hashing approach. Specifically, IRILD achieved a much faster processing speed. As a result, IRILD can be utilized by systems with a large numbers of sensitive documents. Also, IRILD achieved much higher accuracy when compared with traditional cyclical hashing due to the removal of false positives related to public and common phrases.

Bibliography

[1] *Blackbox Search*. http://www.blackboxsearch.com/.

[2] Doc scrubber. http://www.javacoolsoftware.com/docscrubber/index.html.

[3] Dodgy dossier: Microsoft word bytes tony blair in the butt. http://www.computerbytesman.com/privacy/blair.htm.

[4] *Enron Email Dataset*. http://www.cs.cmu.edu/~enron/.

[5] Gnu libextractor. http://www.gnu.org/software/libextractor/.

[6] Hachoir projects. http://bitbucket.org/haypo/hachoir/wiki/Home.

[7] http://blog.postmaster.gr/2009/05/20/on-amka/.

[8] http://malvumaldit.wordpress.com/2009/05/06/e-government-center-for-social-security-societe-anonyme/.

[9] http://www.ethnos.gr/article.asp?catid=11378&subid=2&pubid=8960826.

[10] http://www.insidehighered.com/news/2007/07/24/ idnumbers.

[11] http://www.ssa.gov/history/1930.html.

[12] http://www.techonthenet.com/oracle/errors/ ora01858.php.

[13] The hunt for file format vulnerabilities. http://www.symantec.com/connect/blogs/hunt-file-format-vulnerabilities.

[14] Iana application media types. http://www.iana.org/assignments/media-types/application/.

[15] *Internet Archive*. http://www.archive.org/.

[16] Metadata in arizona public records can't be withheld. http://yro.slashdot.org/story/09/10/30/1539241/Metadata-In-Arizona-Public-Records-Cant-Be-Withheld?from=rss.

[17] Metareveal. http://www.beclegal.com/products.aspx?id=64.

[18] Microsoft office metadata. http://www.document-metadata.com/microsoft-office-metadata.html.

[19] *Planetlab: An open platform for developing, deploying, and accessing planetary-scale services*. www.planet-lab.org.

[20] Poppler. http://poppler.freedesktop.org/.

[21] Remove hidden data. http://support.microsoft.com/kb/834427.

[22] The risk of sharing in asia - may 2005. http://www.workshare.com/downloads/whitepapers/.

[23] *Scroogle*. http://www.scroogle.org/.

[24] Search protocol reference. http://code.google.com/apis/searchappliance/documentation/64/xml_reference.html.

[25] What are the most popular non-html format files on the web? http://www.google.com/help/faq_filetypes.html#popular.

[26] Workshare global security threat report january - april 2007. www.workshare.com/go/research/07aprilthreats.pdf.

[27] N. Agrawal, W. J. Bolosky, J. R. Douceur, and J. R. Lorch. A five-year study of file-system metadata. *Trans. Storage*, 3(3):9+, 2007.

[28] M. Álvarez, J. Raposo, A. Pan, F. Cacheda, F. Bellas, and V. Carneiro. Deepbot: a focused crawler for accessing hidden web content. In *Proceedings of the 3rd international workshop on Data enginering issues in E-commerce and services: In conjunction with ACM Conference on Electronic Commerce (EC '07)*, DEECS '07, pages 18–25, New York, NY, USA, 2007. ACM.

[29] Y. J. An, J. Geller, Y.-T. Wu, and S. A. Chun. Semantic deep web: automatic attribute extraction from the deep web data sources. In *Proceedings of the 2007 ACM symposium on Applied computing*, SAC '07, pages 1667–1672, New York, NY, USA, 2007. ACM.

[30] T. Aura, T. A. Kuhn, and M. Roe. Scanning electronic documents for personally identifiable information. In *Proceedings of the 5th ACM workshop on Privacy in electronic society*, pages 41–50, New York, NY, USA, 2006. ACM.

[31] H. Berghel. Identity theft, social security numbers, and the web. *Commun. ACM*, 43(2):17–21, 2000.

[32] H. Berghel and D. Hoelzer. Pernicious ports. *Commun. ACM*, 48:23–30, December 2005.

[33] M. K. Bergman. The deep web: Surfacing hidden value. *Journal of Electronic Publishing*, 7(1):1–17, 2001.

[34] S. Byers. Information leakage caused by hidden data in published documents. *IEEE Security and Privacy*, 2(2):23–27, 2004.

[35] R. Capizzi, A. Longo, V. N. Venkatakrishnan, and A. P. Sistla. Preventing information leaks through shadow executions. In *ACSAC '08: Proceedings of the 2008 Annual Computer Security Applications Conference*, pages 322–331, Washington, DC, USA, 2008. IEEE Computer Society.

[36] V. R. Carvalho and W. W. Cohen. Preventing information leaks in email. In *SDM*, 2007.

[37] A. Castiglione, A. De Santis, and C. Soriente. Taking advantages of a disadvantage: Digital forensics and steganography using document metadata. *J. Syst. Softw.*, 80(5):750–764, 2007.

[38] A. Castiglione, A. D. Santis, and C. Soriente. Security and privacy issues in the portable document format. *Journal of Systems and Software*, 83(10):1813 – 1822, 2010.

[39] R. Chow, P. Golle, and J. Staddon. Detecting privacy leaks using corpus-based association rules. In *KDD*, pages 893–901, 2008.

[40] A. J. Clark. Document metadata, tracking and tracing. *Network Security*, 2007(7):4 – 7, 2007.

[41] B. D. Davison, D. G. Deschenes, and D. B. Lewanda. Finding relevant website queries. In *In Proc. of WWW*, 2003.

[42] S. L. Garfinkel. Risks of social security numbers. *Commun. ACM*, 38(10):146, 1995.

[43] E. Gessiou, E. Athanasopoulos, and S. Ioannidis. Digging up social structures from documents on the web. Technical Report 412, ICS-FORTH, Hellas, January 2011.

[44] E. Gessiou, A. Labrinidis, and S. Ioannidis. A greek (privacy) tragedy: the introduction of social security numbers in greece. In *Proceedings of the 8th ACM workshop on Privacy in the electronic society*, WPES '09, pages 101–104, New York, NY, USA, 2009. ACM.

[45] E. Gessiou, Q. H. Vu, and S. Ioannidis. Irild: an information retrieval based method for information leak detection. Technical Report 413, ICS-FORTH, Hellas, January 2011.

[46] S. Institute. *Understanding and Selecting a Data Loss Prevention Solution*, 2009. White Paper.

[47] T. P. Institute. *2006 Annual Study: Cost of a Data Breach*, 2006. White Paper.

[48] P. Jaccard. Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bulletin del la Société Vaudoise des Sciences Naturelles*, 37:547–579, 1901.

[49] J. Jung, A. Sheth, B. Greenstein, D. Wetherall, G. Maganis, and T. Kohno. Privacy oracle: a system for finding application leaks with black box differential testing. In *CCS '08: Proceedings of the 15th ACM conference on Computer and communications security*, pages 279–288, New York, NY, USA, 2008. ACM.

[50] T. Karagiannis and M. Vojnovic. Behavioral profiles for advanced email features. In *Proceedings of the 18th international conference on World wide web*, pages 711–720, New York, NY, USA, 2009. ACM.

[51] H. Kwak, C. Lee, H. Park, and S. Moon. What is twitter, a social network or a news media? In *Proceedings of the 19th international conference on World wide web*, pages 591–600, New York, NY, USA, 2010. ACM.

[52] P. T. Labs. *Information Leak Prevention Accuracy and Security Tests*, 2006. White Paper.

[53] H. Lin-Shung, W. Zack, E. Chris, and J. Collin. Protecting Browsers from Cross-Origin CSS Attacks. In *Proceedings of the 17th ACM Conference on Computer and Communications Security*, New York, NY, USA, 2010. ACM.

[54] L. Lu, V. Yegneswaran, P. Porras, and W. Lee. Blade: an attack-agnostic approach for preventing drive-by malware infections. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 440–450, New York, NY, USA, 2010. ACM.

[55] J. Madhavan, D. Ko, L. Kot, V. Ganapathy, A. Rasmussen, and A. Halevy. Google's deep web crawl. *Proc. VLDB Endow.*, 1:1241–1252, August 2008.

[56] Y. Matsuo, J. Mori, M. Hamasaki, T. Nishimura, H. Takeda, K. Hasida, and M. Ishizuka. Polyphonet: An advanced social network extraction system from the web. *Web Semant.*, 5(4):262–278, 2007.

[57] P. Mika. Flink: Semantic web technology for the extraction and analysis of social networks. *Web Semantics: Science, Services and Agents on the World Wide Web*, 3(2-3):211–223, October 2005.

[58] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and analysis of online social networks. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, IMC '07, pages 29–42, New York, NY, USA, 2007. ACM.

[59] P. Nakov and M. Hearst. Using the web as an implicit training set: application to structural ambiguity resolution. In *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 835–842, Morristown, NJ, USA, 2005. Association for Computational Linguistics.

[60] B. Park, J. Park, and S. Lee. Data concealment and detection in microsoft office 2007 files. *Digital Investigation*, 5(3-4):104 – 114, 2009.

[61] J. Park and S. Lee. Forensic investigation of microsoft powerpoint files. *Digital Investigation*, 6(1-2):16 – 24, 2009.

[62] J. Park, B. Park, S. Lee, S. Hong, and J. H. Park. Extraction of residual information in the microsoft powerpoint file from the viewpoint of digital forensics considering percom environment. In *Proceedings of the 2008 Sixth Annual IEEE International Conference on Pervasive Computing and Communications*, pages 584–589, Washington, DC, USA, 2008. IEEE Computer Society.

[63] B. Parno, J. M. McCune, D. Wendlandt, D. G. Andersen, and A. Perrig. Clamp: Practical prevention of large-scale data leaks. In *SP '09: Proceedings of the 2009 30th IEEE Symposium on Security and Privacy*, pages 154–169, Washington, DC, USA, 2009. IEEE Computer Society.

[64] N. Provos, P. Mavrommatis, M. A. Rajab, and F. Monrose. All your iFRAMEs point to us. In *Proceedings of the 17th USENIX Security Symposium*, pages 1–16, 2008.

[65] G. Rossum. Python reference manual. Technical report, Amsterdam, The Netherlands, The Netherlands, 1995.

[66] S. Smith and J. Marchesini. *The Craft of System Security*. Addison-Wesley Professional, 2007.

[67] J. Staddon, P. Golle, and B. Zimny. Web-based inference detection. In *SS'07: Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium*, pages 1–16, Berkeley, CA, USA, 2007. USENIX Association.

[68] D. Stutzbach and R. Rejaie. Characterizing the two-tier gnutella topology. In *Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, SIGMETRICS '05, pages 402–403, New York, NY, USA, 2005. ACM.

[69] G. Wondracek, T. Holz, E. Kirda, and C. Kruegel. A practical attack to de-anonymize social network users. In *Proceedings of the 2010 IEEE Symposium on Security and Privacy*, pages 223–238, Washington, DC, USA, 2010. IEEE Computer Society.

[70] W. Wu, C. Yu, A. Doan, and W. Meng. An interactive clustering-based approach to integrating source query interfaces on the deep web. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, SIGMOD '04, pages 95–106, New York, NY, USA, 2004. ACM.