Stony Brook University

Content-based Access Control

A Dissertation Presented

by

Michael Andrew Hart

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Doctor of Philosophy

in

Computer Science

Stony Brook University

May 2011

Stony Brook University

The Graduate School

Michael Andrew Hart

We, the dissertation committee for the above candidate for the

Doctor of Philosophy degree, hereby recommend

acceptance of this dissertation.

Rob Johnson – Dissertation Advisor
Assistant Professor, Computer Science

Yejin Choi - Chairperson of Defense
Assistant Professor, Computer Science

Scott Stoller
Professor, Computer Science

Mónica F. Fernandez-Bugallo
Professor, Electrical and Computer Engineering, Stony Brook University

This dissertation is accepted by the Graduate School

Lawrence Martin
Dean of the Graduate School

Abstract of the Dissertation

Content-based Access Control

by

Michael Andrew Hart

Doctor of Philosophy

in

Computer Science

Stony Brook University

2011

In many contexts, users are either unable or unwilling to specify their access control policies. In Data Loss Prevention, for example, users cannot fully express what is secret in rule-based formats. Many users are unwilling to use access controls, particularly in the Web 2.0, because they are too draconian, leading to disastrous consequences in terms of privacy.

To address both of these issues, we have introduced the concept of Content-Based Access Control (CBAC). CBAC combines content recognition with policy acquisition and enforcement. A CBAC-enabled system can be trained to recognize policy violations by learning what is secret from examples. This defense will discuss how CBAC can be successfully applied to Data Loss Prevention, Wikipedia Vandalism and the Web 2.0. Usability is integral to providing better CBAC systems and privacy interfaces, and this defense demonstrates improvements in the usability of these systems.

Table of Contents

List of Tables

List of Figures

Acknowledgments

First, I want to acknowledge the constant love and support of my wife through the arduous process of completing a Ph.D. She is truly mi luz en la noche mas negra.

I also acknowledge the hard work, love and support of my parents who enabled me to succeed.

I also want to acknowledge the long list of individuals I have collaborated with. First, I thank Rob Johnson for taking me on as his first PhD student and supporting me fully as we explored uncharted territory. I also want to thank Amanda Stent because without her invaluable contributions early in my academic career, I would not be where I am now. I also want to thank Pratyusa Manadhata for excellent guidance and mentorship. I have had the pleasure to work with fantastic students including Megha Bassi, Dave Butala, Claude Castille, Ali Mirsaidi, Jeffery Mezic, Chaitanya Ravi, Sandesh Singh, and Longfei Xing.

I also acknowledge the wonderful Open Source Community which facilitated my work by providing free, high quality software.

Lastly, I acknowledge that I could not have gone this far without the dedication and support of the teachers from K-12.

The text of this dissertation in part is a reprint of the materials as it appears in “Text Classification for Data Loss Prevention”, “Language of Vandalism: Improving Wikipedia Vandalism Detection via Stylometric Analysis ”, “Prevention and Reaction: Defending Privacy in the Web 2.0 ”, “Wiki Vandalysis - Wikipedia Vandalism Analysis ”, “iTag: A Personalized Blog Tagger ”, “Usable Privacy Controls for Blogs ”, and “More Content - Less Control: Access Control in the Web 2.0 ”. The co-authors listed in the publications directed and supervised the research that forms the basis for this dissertation. For these materials, permission has been granted for their use by all authors: Rob Johnson, Pratyusa Manadhata, Manoj Harpalani, Yejin Choi, Sandesh Singh, Megha Bassi, Thanadit Phumprao, Amanda Stent and Claude Castille.

Chapter 1 Introduction

The reach of Web 2.0 applications has gone beyond most everyone's expectations. Facebook started as a social network confined to Harvard University [1] and now boasts over 700 million users, 70% of them outside the United States [2]. Twitter, despite its inherent brevity, breaks news stories [3] and the word “tweet” has entered the lexicon of many Internet users [4]. Justin Bieber, an international pop sensation, got his start singing covers of other artist's songs on the video sharing community YouTube [5].

The cost to the privacy and welfare of individuals, however, has been great. User privacy has been compromised by confusing and invasive practices [6] and Web 2.0 service providers have continuously moved to share more of their users' private information with third parties for financial gain. Users have been fired for what they have written on their social networks [7][8] [9][10] and blogs [11][12]. Lawsuits and criminal cases have emerged from comments and blog postings online where authors believed they were within their first amendment rights [13]. More gravely, harassment on Facebook [14] and personal ads on Craigslist [15] have led to tragic outcomes.

A failure of access control is partly to blame for these dire events. Because of the amorphous nature of personal information, users are often *unable* or simply *unwilling* to specify their privacy preferences. In many situations, users are *unable* to translate their policy into machine readable terms because they cannot easily encode their preferences in the rule-based

formats that underpin many access control systems, despite being able to articulate in simple language what is sensitive or not permitted. Users are also *unable* to protect their privacy when it comes to phishing. Many authentication systems require the user to be vigilant and therefore part of the authentication system. Phishers exploit lapses in user vigilance to trick them into revealing their username and password for valuable online services, such as their banking account.

Users are *unwilling* to use access controls that are tedious and require substantial effort to maintain. The problem will only grow worse as more content is stored online, new channels of communication (e.g. social networks, email, tweeting) open, and agents, such as search engines, become more capable of processing and inferring facts from this information.

The rest of this chapter will provide the background and ramifications of access control on the Web 2.0. Chapter 2 will take a closer look at why existing access controls are not capable of or simply unusable for protecting users' privacy. Chapter 3 will detail our proposal: Content-based Access Control (CBAC) that addresses the shortcomings of current access controls. Chapter 4 will describe Plog, a CBAC enabled system and policy language for blogs. iTag, our automatic tagger built for Plog, is described in Chapter 5 . Situations where users are unable to express their privacy preferences in machine-readable form are discussed in the following chapters. Chapter 6 will elaborate on our strategy for building text classifiers for Data Loss Prevention. Chapter 7 will discuss our efforts to automatically identify vandalism on Wikipedia. Chapter 8 will discuss how we can build better password-authentication systems that do not rely on user vigilance. Chapter 9 will provide a conclusion.

1.1 The Web 2.0

Before reviewing access control methodologies and consequences of popular web services, it is useful to define key concepts in the Web 2.0. Active authors and developers disagree on what features actually characterize the Web 2.0 [16]. The term, which is attributed to Dale Dougherty, garners approximately 9.5 million Google citations [16]. The notion of "Web 2.0" was created to encompass several crucial changes that web services underwent during the early 2000s. Tim O'Reilly, founder of O'Reilly Media and proponent of Web 2.0 services, enumerates key distinctions between Web 2.0 and the original conception of the Internet [16]:

- Web as a principal platform for users
- Harnessing collective intelligence
- Data as the key resource (as opposed to speed)
- End of the software release cycle
- Lightweight programming models (simple web services like RSS and Rest)
- Rich user experience

Blogs, a very popular Web 2.0 application, receive a lot of attention in the media and in research. Blogs, known initially as Weblogs, are simple, web content publishing systems. Most blog services, such as Blogger, Xanga, and LiveJournal are free. Three main themes classify blog content: personal online journals, aggregators of commentary and content on other sites and knowledge logs [17]. The blogosphere is a vernacular term denoting the space in the Internet devoted to this activity. Nardi et al has found the following common blogging practices [18].

First, most blogs are discovered through reading other blogs. They noted that despite their study's small sample size, there existed a great diversity in the themes and topics of blogs, ranging from the banal to the serious. Another important observation is that blog readers do not simply view posts. Rather, the authors found that blogging is a dynamic social activity that instigated conversations and creation of new blogs. Three key aspects of this dynamism are personal updates, expressing opinions, and solicitation of feedback.

Another popular service in the Web 2.0 is social networking. These sites, such as Facebook and MySpace, allow users to create profiles (much like home pages), search for others, leave comments and post multimedia. Many users from different backgrounds populate these networks. Business organizations are also active in these networks and use the sites for promotion and advertisements. The importance and dynamics of virtual social networks are beginning to be understood. The Pew Foundation released a report regarding social interaction across the Internet [19]. They found that web services do in fact support social networks and facilitate regular communication for large networks more efficiently than traditional technologies like telephones. Two other interesting points from the study are that email does not replace out-of-channel communication and users elicit these servers to collect responses from social networks to make important decisions. Boyd took a poignant look at teenage use of a popular social network service amongst this demographic: MySpace [20]. Whereas other social-networking sites have come and gone, Boyd speculates that the importance of profile creation and commentary as well as asynchronous communication comprised of profile to profile instant messaging sustains the popularity of MySpace. Profile creation on these sites ties into the

identity production that occurs during the teenage years. Their profiles give teenage users the opportunity to discover and discern what is "popular" and how to project themselves. Boyd also argues that MySpace has become a "digital public". Where social mores have changed such that teenagers are limited in where they may physically congregate, MySpace allows them to socialize freely. The issue here is that MySpace is not always an age appropriate space for minors. Rather, other users and agents (e.g. advertisers) regularly disseminate adult ideas and themes through these networks.

Wikis and video-sharing websites are two other popular Web 2.0 applications that may not be as intimately intertwined with the personal lives of their members. The rise of Wikis mark the culmination of the collective intelligence of the web. Wikis are websites where visitors and authors can edit, delete, comment and add content to articles or pages. Wikis focus on a particular purpose or theme. Popular Wikis include Wikipedia, a freely editable encyclopedia, and wikiHow [21], a freely editable how-to manual. Wikipedia is the most widely recognized Wiki and has made the news for it's factual accuracy [22], high profile individuals changing their entries [23], and editors with fabricated credentials [24].

With increased network capacity, video-sharing networks have also become quite popular. Services like YouTube and Yahoo! Video stream user-created and commercial videos. Users tag videos with words or phrases that provide categorical information. The ramifications of on-demand video streaming are palpable. Broadcast network CBS experienced a boost in its audience after making content available online [25] A new art form of exclusive web video

drama captures the imagination of viewers, attention of news media outlets and the efforts of aspiring filmmakers [26]. Lastly, these channels are challenging copyright laws [27].

The discussion of these Web 2.0 services does not nearly encompass the entire sphere of technologies and applications available. It should be noted, however, that many crucial observations can be made about how users interact with these systems. First, an abundance of data is available online, especially personal. We must also realize that many users extend their social lives to encompass Internet technologies. The potential exists for search engines and other data harvesters to actively mine these systems for personal information. This raises questions about privacy concerns in light of increased exposure online.

1.2 Consequences of the Web 2.0

Despite the new and exciting applications emerging in the Web 2.0, the ramifications for users are increased exposure and scrutiny. Even in the relatively short lifetimes of popular sites, the consequences of insufficient access control are well-documented in the news media:

- Bloggers have lost their jobs when their employer discovered the employee's personal blog [11][28][29].
- Sexual predators use social networking sites to find victims [30].
- The opinions and personal information placed on victim's blog increase the possibility of stalking [31].
- Universities use photographs taken at student parties and posted on student social networks or blogs as evidence to reprimand the students involved [32].

We are approaching an age where privacy is an illusion. If the microscope turns to anyone you will ALWAYS find dirt. Administrators and politicians will use our electronic footprint to build a case against anyone they want to.	*You have to be an idiot to put pictures out on the Internet like that, and then think that nobody is looking at them. Whenever we interview here at work, the first thing we do after the interview is check out their myspace or facebook pages. We've turned away a lot of candidates based on some of the stuff we see there. This is why I don't have a myspace page.*

Table 1: Two comments from the Washington Post's Blog that have diametric views of online privacy.

A didactic example of this privacy invasion resulting from using these services which received substantial media coverage highlights how users and outsiders view privacy expectations in this domain. Stacy Snyder, a student who was working toward her teaching certificate, was denied certification on the basis of a photo she posted on MySpace. The photo included the caption “Drunken Pirate”. Because of the photo, she received unsatisfactory marks for professionalism in her evaluation. This eventually prompted the university, in response to pressure from the school system where she student taught, to deny her an education diploma and granted her an English degree instead. This prevented her from becoming a teacher, her intended occupation. Reviewing the story on many different media outlets, reactions were quite varied and diverse. Table 1 shows a particular set of comments juxtaposed on the Washington Posts Blog site that stand out.

These two comments symbolize the issues when it comes to defining privacy in these new social contexts. The author of the first comment believes that the mores of the blogosphere should be respected and that this course of action by the university constitutes as an invasion of privacy. The second comment takes quite an opposite view. The author believes that since the

information is accessible, those who acquire the information may apply it in any offline context they choose.

To compound matters further, the law is not necessarily clear on exactly if a policy violation has occurred. An interesting parallel to this case that went under legal review where an employee's supervisor searched Google and found evidence of prior misconduct [33]. Ultimately the defendant was fired. The defendant claimed that Google searches constituted ex parte communication. The judge in this case ruled that ex parte communication did not occur considering there was already substantial evidence against the defendant. We must be aware that this situation possibly occurred in Ms. Snyder and others' cases because Facebook and MySpace provide search engines to find users. Legislation also is lacking for broad issues of privacy online. There are notable laws for specific issues regarding online activity. The Children's Online Privacy Protection Act ensures that children online do not submit personal information without the consent of parents [34]. Another prominent piece of legislation is the Digital Millennium Copyright Act that protects content owners from illegal distribution and prosecutors have applied it against prominent web services [35].

One legal expert, Helen Nissenbaum, proposed new definitions of privacy despite new and pervasive channels of surveillance. She advocates the idea of contextual integrity, where the norms of information flow in a particular context (social situation) are preserved regardless of the ability of modes of surveillance that are active simultaneously [36]. Bloggers and social-network users conceive proper usage of their information through interaction, even if the spaces

themselves are not tangible. Therefore, the case of Stacy Snyder should have been considered a policy invasion.

In light of the privacy and legal ambiguities that arise in the Web 2.0, what is the mandate of those providing these systems? Perhaps one attitude is to let society and law determine what constitutes privacy in these domains and leave it to the courts to take appropriate action. From a computer security standpoint, a different conclusion should be reached. Regardless of the numerous debates about online privacy expectations, the Snyder case is important from an end-user standpoint. Ms. Snyder decided that the image was innocuous enough to be posted online and enjoyed by friends. She would probably have expected that those who came across it (intentionally or unintentionally) would have viewed the image and moved along. This was not the case: someone accessed the photo and introduced it into a new social situation, Ms. Snyder's performance evaluation. Herein lies the flaw in access control in these systems: those who may fundamentally disagree with the expectations of privacy and use of the author may still gain access to the content. In this case, the result was dramatic and costly.

Therefore, this case can not only be viewed as a society grappling with new notions of privacy, but as a call to designers of these systems to devise better access controls such that users are able to express their privacy preferences easily and effectively. Users cannot simply rely on what they expect viewers to do with their content, as presumably Ms. Synder did. We advocate that the access control system needs to do more for the end user. We can inform the design and enforcement of access control systems with legal philosophies such as contextual integrity and mores established by users. For instance, when a user posts any informal content that could be

construed as questionable, then the system can help identify A) why it is potentially harmful and B) who should have access to it. In Ms. Snyder's case, the access control system should have tried to help better identify which subset of users would be the most amenable to her expectation of use for the picture in question. Society needs to debate and define the parameters for the expectation of privacy, but when it comes to the online realm, access controls should try to enforce these boundaries.

Chapter 2 Current Access Controls

This chapter will elaborate current access controls as well as access controls proposed in literature. To better understand the limitations for current access control approaches to privacy, we must first examine what type of data is being collected and how users are able to mediate access to it.

2.1 Introduction

Web 2.0 services collect personal data of many types and through a variety of channels. Bruce Schneier has proposed a taxonomy of social network information [37], which we reproduce below:

- **Service data:** Data you give to a social networking site in order to use it (e.g. your legal name, age).
- **Disclosed data:** Data you post on your own pages: blog entries, photographs, and status updates.
- **Entrusted data:** Data you post on other people's pages that you lose control over once you post it.
- **Incidental data:** Data people post about you (e.g. a paragraph about you)
- **Behavioral data:** Data the site collects about your habits via behavior on the service

- **Derived data:** Data about you that is derived from all the other data.

This taxonomy illustrates three sources for dissemination of personal information on the Web 2.0: the user himself, others, or from inference. Therefore, privacy can be compromised from any one of these three sources. For the purposes of this analysis, we consider behavioral data out of scope, although it is a great concern to user privacy with respect to third-party applications [38]. We also will note that some service data, particularly in social networks, is also disclosed data (e.g. name).

How much control can a user exercise mediating this data to viewers? We surveyed twenty three blogging and social networking sites to determine what access control and privacy features are currently available. Our survey included traditional blogging sites (e.g. Blogger and LiveJournal), social networking sites (e.g. MySpace and Facebook), photo-sharing sites (e.g. Flickr and SmugMug), and video sharing sites (e.g. YouTube). The access control features implemented in these services fell into a few broad categories, with some sites offering minor extensions.

- **Private/public:** The simplest access control systems only supports private and public objects. Private objects are only viewable by their owner. Public objects are readable by any human or computer, such as a search engine, in the world.

- **Friends:** Some sites augment the basic private/public scheme by letting users create a list of "friends." A user can restrict some of his or her content to be visible only to friends. The "friends" scheme may be bolstered by mechanisms to automate the process through invitations (Facebook) or manual specification (Windows Live Spaces). The Orkut social

network attempts to maintain some level of trust among its users by only registering new users that receive invitations from existing users.

- **Other registered users:** The Xanga blogging site lets users restrict posts to be visible only to other registered Xanga users. Since anyone can create a Xanga account for free, this feature may be primarily intended to prevent web search engines from indexing certain pages.
- **Only registered users:** For certain valuable or sensitive articles, only registered users have the ability to edit an article [39].
- **Search engines:** WordPress allows users to indicate that a post should not be visible to search engines. The WordPress server includes the advisory robots.txt mechanism in the user's directory, which search engines are urged, but not forced, to follow.
- **Password-protected Posts:** SmugMug and WordPress allow users to create password-protected posts. While this enables extremely flexible access control policies, it places the burden of managing the policy entirely on the user.
- **Age restriction:** MySpace does not allow anyone younger than thirteen register. Those users between the ages of 14 and 15 may only display partial profiles to anonymous viewers except if they are a friend of the user. Although this is a good first step to separate children and adults, the site has no way to actually verify if the registered user is the age he or she claims to be [40].

- **Network restrictions:** Facebook allows users to choose different levels of privacy determined by which networks one belongs to. These networks may either be regional or attribute based (e.g. place of employment).

Of the existing schemes, the "friends" model is the most successful because it strikes the best balance between ease-of-use and flexibility. However, it still has several flaws. First, it does not let users segregate their disparate social groups. Unfortunately, users cannot distinguish real-world friends, who presumably already know the user's home address and other personal information, from online friends, who may be close, but do not need to know offline details about the user. More generally, a user may have legitimate reasons for keeping friends from different contexts separate. For example, a blogger might want to separate his or her work-related friends from college friends, but currently the only way to do this is to maintain separate blogs. Second, in terms of privacy, there are different degrees of intimacy between an individual and each one of his or her friends, but the "friends" notion is too coarse to capture these distinctions. We found that MySpace users had a median of 115 friends based on a random sample of 91660 MySpace users. Few people have 115 close confidants, so the friends relationship is obviously being strained to encompass many levels of intimacy. Making matters worse, using the friends relation for access control forces users to choose between protecting their privacy and appearing popular or gregarious. Many users of these systems perceive a sense of popularity resulting from having large number of friends. In fact, MySpace perpetuates this notion by a feature called "Top Friends" that showcases the eight best friends in the user's profile

above other friends. In all these cases, the role of "friend" becomes ambiguous. As the notion of "friend" loses its meaning, friends-based access control also becomes meaningless.

Surprisingly, only recently have services allowed users to use group-based access controls in their services. Facebook appears to offer the most extensive and customizable array of privacy features despite being lambasted in the news media before for their privacy control [41]. Facebook has taken steps to enable users to manage incidental information. Users can limit access to photos of themselves, even if the photo was posted by someone else. Also, Facebook allows users to control which friends can view content posted on their wall (a section of a user profile to which others can write messages). The incidental information problem cannot be solved by just one service, though -- users still have no recourse when content is posted about them on sites over which they have no control. Facebook has also attempted to help users limit inference about themselves by controlling their visibility in Google and Facebook search results.

Although Facebook has made great strides to protect individual's privacy, users still experience privacy invasions. Users are often unable to see the consequences of their actions. For example, several employees have lost their jobs after criticizing their employer on their blog or social networking profile [8][9][10][7]. Studies have shown that users do not understand or check for security mechanisms [42]. Therefore, it is quite possible that most users are not aware of the full ramifications of their actions.

Lastly, the sophistication and reach of search engines endanger individual's privacy and online reputation. One particular instance that drew the attention of the news media concerned a

female law student at Yale University [43]. She was the subject of derogatory comments on the AutoAdmit law school discussion board. The law student only learned of the online discussion through a friend and after her job search resulted in no offers, despite having comparable credentials to other successful recent graduates. Although she could not conclusively prove that the comments had prevented her from being hired, it is quite likely according to the Ponemon Institute [43] which found that half of U.S. hiring officials use search engines to investigate potential employees, with one third of searches yielding information that will deny the applicant a job.

Therefore, the following trends do not bode well for privacy online:

- Most sites offer draconian access controls that are largely unappealing to users.
- The ramifications of using Web 2.0 services is beyond the user's technical grasp of the software.
- These services do not provide any oversight or a safety net for users.
- Search engines enable searchers to collect information about individuals based on information from multiple web pages.
- Sensitive information about the user may be revealed through inference.

2.2 Other privacy and access control schemes

Although Web 2.0 services appear to prefer simple and draconian access control schemes, there do exist other access control schemes and paradigms that might be brought to bear on this privacy concerns in the Web 2.0. The following subsections serve as a survey of

different schemes advanced in industry and literature that are designed either to A) improve access control or B) help mitigate damage from privacy loss by specifying expected usage.

The main drawback of deploying these proposals to address Web 2.0 privacy concerns is that they do not address two main issues: the acquisition and automated inference of privacy policies. The XACML, EUFORBIA, and Policy-Aware Web systems provide access control for the Web 2.0 and distributed systems. The main issue with these proposals is that they largely ignore the acquisition and automated enforcement of privacy policies. As established in the previous section, it is dire that users can easily express their privacy preferences in a language that is familiar to them that will provide some degree of protection through automated policy inference from accidentally compromising their privacy. We also examine three other proposals, P3P, Social Contract Core and the Accountability Perspective, that focus on empowering users by enabling them to either express or audit data usage policies. These proposals do shed light on an important aspect of data usage: accountability. By focusing on accountability, privacy invasions can be addressed through the law, which may help users recoup on losses they suffer. But the main issue is that policy exchanges simply cannot fully protect the user, particularly when the languages to express policies are confusing and circumventable. If a privacy invasion occurs and the user is simply not aware, accountability has no value to them. Therefore, users will still need a proactive means to express their privacy preferences and enforce them.

In the following sections, we will examine these systems, highlight what is useful to the Web 2.0 privacy problem, but argue why they will not ultimately be effective for the average user.

2.2.1 eXtensible Access Control Markup Language (XACML)

XACML [44] is a message passing system that utilizes XML in order to stipulate policies and enforce them. Enforcement of policies are performed in tandem by the Policy Decision Point (PDP) as well as a Policy Enforcement Point (PEP). The PDP has access to attribute sources, which are the properties about subjects and users, and policies. The Policy Enforcement Point takes request from users and formats a corresponding XACML request with the subject attributes. An important point is that PDP and PEPs may be distributed and do not necessarily require one monolithic policy nor decision server. A key difference between EPAL [45] and XACML is the fact that there exists a standardized mechanism to define new datatypes and functions, hence the term Extensible in the title.

XACML policies consist of PolicySets, Policies, and Rules. PolicySets may contain other PolicySets, and Policies. When a request is made, XACML policies contain targets that determine the applicability of a Rule, Policy or PolicySet to the attributes of the requester. Once a policy is found to apply to a request, the rules are evaluated. The rules essentially stipulate a **Condition**, a logic function on attributes that returns true or false, that determine the access control decisions of permit or deny. It is important to note that **Conditions** are arbitrarily complex and may contain non-boolean functions and attributes. XACML employs default types and functions to handle this complexity.

Lorch et al. [46] surveyed first adopters of the XACML standard. They investigated three diverse technologies: Shibboleth, Cardea and PRIMA. Shibboleth is a web-based authentication system that facilitate communication between higher education institutions. Cardea is an access

control system for NASA's Information Power Grid. What makes it different than Shibboleth is that characteristics of both the resource and requester determine access. Another interesting facet is that the subject can only provide credentials that he or she can demonstrate. This implies an absence of an attribute source used by a PEP. Cardea also uses SAML [47] in order to facilitate the XACML requests. The triumph of this system is that resource owners may stipulate arbitrarily complex policies and apply standardized policies on a heterogeneous collection of resources with external requesters. PRIMA provides distributed access control in a grid computing environment. It also allows multiple authorities and users to mediate access control on their own resources. Three type of access control information are stored on the system. The first is privilege attributes that are created to denote individual rights to data. Privilege Management Policies specify authorities of resources and how delegation and privilege management rules follow. Lastly, traditional access controls are supported as well. These early adopters have successfully incorporated a wide variety of access control methodologies in a standardized and distributed fashion. In fact, more than 50 companies now support the standard, including large software companies like Adobe [48].

The strengths of XACML lie in its ability to be distributed and extensible. The successful deployment of XACML suggests that a wide variety of privacy preferences could encoded in this language. We do not believe that it will be successful in a large-scale deployment such as the Web 2.0 where the users are largely non-technical and services largely do not communicate users' privacy policies outside of the services' domain. XACML systems inherently do not possess any policy acquisition features that will help protect users from privacy

invasions. Rather, they may have tremendous difficulty expressing their privacy policies in this language, particularly where many disparate parties must agree to a common set of data types. Also, the enforcement of policies is largely left to the implementer. There is no intrinsic quality about this language that would provide automated policy inference.

2.2.2 Platform for Privacy Preferences (P3P)

Arguably the first widespread policy platform and protocol, P3P was one of the first policy exchange protocols. Policy exchange protocols help the requester decide if it is worth accessing the resource given the owner's policies. In principle, policy invasions would be minimized because requesters would have to conform to the usage expectation of the user. P3P was originally envisioned to be an exchange between a server and client in which the server would declare its data handling and usage policies. These policies are related to the data collected from the client to the server and how it would be used subsequently. At the present time, P3P development is now officially suspended. It does serve, however, a useful pedagogical example that allows us to analyze the rise and fall of this technology, particularly with a renewed focus on use of personal data by advertisers in the US congress [49].

At the core of P3P is an XML language that encodes policies [50]. The important tags are **entity**, **access**, **disputes**, **data**, **purpose**, **recipient**, **retention** and **consequence**. The **entity** tag contains contact information for the business, organization or individual providing the content. The **access** tag specifies what information the application collects and how users may find information about themselves on the site. If the user feels that their privacy has been infringed, the **disputes** tag allows them to find out whom to contact and how to reach them. The

data tag describes what data are collected from the user and the **purpose** states how data are used. The **recipient** tag explains how data will be shared and the **retention** tag states the amount of time data will be kept. Lastly, the **consequence** tag provides a human-readable explanation of the data handling practices. Once a policy has been composed, the exchange of the policy and client is achieved by an HTTP request. The location of the policy is put in a well known location on the server. The policy is included in either a special HTTP header or in a link tag. In most cases, the links provided are a compact policy that describes the usage of cookies on that site. Once the policy has been downloaded, the user can choose to opt-in or opt-out. This means that the user either consents to the privacy policies of the site or decides to not participate. Non-participation results in no exchange of personal data, but also a denial of service.

Although P3P is now largely unused, Cranor et al. provides us a historical snapshot of early adopters of the technology. Cranor's group created the AT&T Privacy Bird [51]. First, a user will install the tool into Internet Explorer and either provide their privacy preferences or select a general security level. The tool resides in the browser plugin that displays a bird located in the bottom right of the window. The bird changes colors and make different sounds depending on the compatibility of the site's policies with the users. For example, the bird squawks and turns red if the site policy is incompatible with the user's privacy expectations. A GUI resembling a “nutrition label” highlights the important parts of the policy. This interface also presents personal preferences to the user.

The privacy tool experienced a relatively successful launch with twenty thousand downloads in the first six months. In response, the authors conducted a survey of two thousand

users, of which three hundred thirty one completed the survey. Of those who responded, ninety percent were at least somewhat concerned with their personal privacy online and ninety eight percent concerned with companies sharing information with third parties. One frequent criticism that the authors received was that most sites elicited the yellow bird, which meant no P3P policy existed for that site. This probably contributed to a sense of end users that P3P was not all that useful. Users liked the tool and rated the policy summarization 3.3 out of 5 point scale. Users also made it clear that they were interested in the reputation of the site, which is not available in the P3P protocol. Interestingly, those who never or occasionally read site privacy policies dropped to fifty three percent from seventy eight percent after installing the privacy tool. The authors concluded the fact that more people were reading site policies and a drop in those reading the policy on every site was an encouraging sign. Overall, eighty eight percent of users changed their online behavior when using the AT&T privacy bird.

Beyond lack of adoption, Internet users and services may have not adopted P3P for a variety of reasons. Grimm points out that users' opting out of the service would result in a denial of service [52]. Cranor notes that many of the policies were formulated in a legal language that was beyond the immediate comprehension of average users. Developers found developing GUIs for P3P difficult because the policy language was incredibly flexible and complex. Also, some respondents to the Privacy Bird did not trust the privacy policies on the website. This is compounded by the fact that there are no ways to audit a companies actions within the protocol. Therefore, P3P does not do much to allay user privacy concerns. It also does not appear likely to be successful mediating access between a user's content and potential user. The lack of adoption

experienced by the authors of P3P suggest that policy exchange is simply not enough to truly protect users on the Web 2.0.

2.2.3 Social Contract Core

An extension of the P3P architecture is the Social Contract Core (SCC). The authors chose the name Social Contract Core to reflect trends witnessed in the evolution in society, but are not extended to our virtual communities. The Social Contract [53] is a termed coined by the philosopher Jacques Rousseau to illuminate the agreement between individual and state. As members of society and state, individuals sacrifice some autonomy for the "greater good". In return, the people should contribute and consent to the development and enforcement of norms.

The Social Contract motivates technological adoption. An example is the telephone system. We would consider that someone calling at 4 A.M. to be outrageous. The general acceptance of this and other mores regarding the telephone then stimulates adoption and sharing of these technologies. A Social Contract also protects privacy. When a client shares personal information with a third party, there are expectations of usage. First, we expect that when a third party accesses personal information from this service, the transaction honors the agreement between the requester, service and owner for its acquisition. The service should also honor obligations and resolve disputes regarding policies. Lastly, the infrastructure that hosts the data should be secured to prevent malicious access. The issue that exists with these policy schemes, particularly P3P, is that discrepancies between user preference and site policies result in denial of service. There are no mechanisms in which the users can assert their preference to change the site's policy.

Therefore, in response to this impasse, Lorch et al [54] proposed an architecture called the Social Contract Core where users define their privacy preferences. The architecture is comprised of three modules: the Client unit, the Site Owner unit and the SCC Conventions Site. The Client Unit comprises an interface, the personal policies of the user, and a proxy to communicate preferences with others. The proxy, typically a browser extension, does the typical P3P functions of reviewing policies, determining compliance, and allowing users to opt-in and opt-out. The proxy can download online privacy profiles from SCC Convention sites that allow users to configure their settings to align with the norms and values of that group. The proxy supports a voting mechanism. The voting mechanism compiles the data handling practices of the site the user opted in or out of. This will allow web sites and SCC Convention sites to collect anonymous statistics that describe trends and help compile emerging accepted standards of use in these domains. The Site Owner is essentially the web site that hosts the desired service. It will maintain its own SCC Proxy capable of receiving votes and collect statistics regarding user policy preferences. The incentive for businesses is that the feedback from users may reveal marketing strategies acceptable to clients. Therefore, users are less likely to opt out and perhaps engender a more viable revenue stream. The authors also conceived a new web service called the SCC Conventions Site. This service aggregates group social preferences and privacy policy standards in order to evaluate web service policies and disseminate privacy preferences. These sites may also be proactive and contact popular sites that are in disagreement with their privacy policies.

The dynamic of this system is analogous to union representation. Although there is a voting mechanism, users cast votes that do not necessarily change the sites policy. In fact, it is the site owner's ultimate decision to decide whether or not to change the sites policy. By creating these SCC convention services, powerful arbitrating bodies emerge on behalf of the users. That is, if a significant percentage of a site's users enlist a particular SCC convention service's preference, it is compelling for the web site to change its policies to be compliant. This may not be the case if the service is in demand and users are not willing to arbitrate or opt-out. This type of system does reflect many relationships business and consumers have with each other in real life and how change is effected.

An SCC system will help users explicate the mores and privacy expectations of the services that they use. As of right now, many of the most popular services do not have a simple means to specify terms of usage for the users, presumably it will be done in an ad-hoc fashion if at all. An SCC system could help in resolving legal issues that arise, such as in the case of Stacy Snyder. But this system only enables reactive means to dealing with privacy invasions, and not proactive. Therefore, we believe that it is important for users to express their privacy preferences, particularly when considering the usage of advertisers of disclosed data. But users need usable systems that will acquire access control policies and perform inference to minimize privacy invasions.

2.2.4 Policy-Aware Web

Another platform for policy exchange in the Internet is the Policy-Aware Web [55]. The authors of this project claim that the fundamental impasse for users to share information over the

web is that the proper mechanisms do not exist to enforce the user's policies. The authors explain that the ultimate adoption of the Semantic Web relies on not only the technology, but the social conditions surrounding it. There are several technologies that provide the basis for portable and machine readable data exchange, but are useless if the environment is not safe to do so. The authors argue the web "has failed in equal measure at satisfying other critical policy requirements such as privacy protection, a balanced approach to intellectual property rights, and basic security and access control needs". And although it is tempting to allow law and society to determine what online privacy entails, it is up to developers to create the technical capacity to model and enforce privacy expectation and assurances for a wide array of social activity.

Although many web services replicate social interactions online, it lacks important aspects that are crucial to physical meetings. The authors give an example of an over-eager librarian to demonstrate the types of cues absent in online discourse. If we are interested in a book, we may inquire its location on the shelves. But if the librarian were to follow afterward and watch one intently, there are many ways to stop this behavior. Such actions may be as simple as staring back, displaying displeasure or intervention of third parties with increasing authority. Online transactions lack this type of response and feedback. More worrisome is that certain insidious tracking behaviors such as usage of cookies may be not well understood or observable and thus not addressed by the client. Tasks such as setting up a photo gallery online is much more onerous than in real life. Currently, one would have to specify access for identities or IP addresses and make announcements which requires much manual labor. This would be unacceptable if this were to be done for numerous photo albums. The semantic web provides the

opportunity to address this disparity between real life and online social transactions. Technologies such as OWL ontologies are able to capture the complexity of relationships between individuals and communities.

For the web to truly be policy aware, the Policy-Aware Web strives to meet three requirements. The first is transparency. This means that both end-users and machines need to easily access and understand the rules on privacy and exchange of data. For example, the end-user can identify the inherent intellectual property rights of the item (e.g. may it be publicly shared). Some efforts have been made toward this goal with web policy languages such as XACML. In order to determine if policies are adhered to, compliance models must be incorporated into the mechanisms that operate in the web (i.e. Web servers, browsers, etc.). The goal of these systems is to enforce the formalism associated to the various technologies while not burdening the user experience. Lastly, despite all efforts of privacy specification as well as compliance mechanisms, accountability is needed. That is, malicious users will take advantage of seemingly well described rules or gain unauthorized usage of resources. Auditing enables the appropriate response on the behalf of system administrators to protect content owners to maintain privacy constraints.

Although RBAC has been a popular way of providing access control, the Policy-Aware Web approaches the problem from a rule-based paradigm. First, the project developers claim there is a move away from role and identity based access controls toward rule-based access controls. A major issue with RBAC is that role engineering is difficult, especially with regards to changing needs. The authors provide an amusing anecdote where an individual needed a

document but lacked the proper role to access it. The complication arose that either the individual would have access to a class of documents that he should not have access to or the document would be released to a group that was not authorized to read it. Another subtle issue arises that RBAC policies are difficult to set up since web servers receive requests that point to a node in a file-directory structure. Therefore, it is more conducive to a rule-based approach since resources can be specified with URIs.

Rule-based access control is a system where a set of rules are associated with a resource and a requester demonstrates it can satisfy these rules. The system has been associated with Mandatory Access Control schemes since it is easy to formulate rules based on labels. Discretionary Access Control has not seen as much development of rule-based systems, but the authors believe that the Web is well suited for this task.

Implementation of the Policy-Aware Web that utilizes discretionary rule-based access control and meets the aforementioned requirements needs a compliant rule language, a rule reasoner and protocol. The authors choose a rule language based on RDF called Notation 3 (N3). The attractiveness of N3 is that N3 provides a human readable logic language experiencing considerable open-source development. N3 satisfies the transparency requirement. The authors employ an open source software package called Cwm [56] to determine if rules are satisfied. Cwm is an RDF-based reasoner that has been co-evolving with N3. It is a forward-chaining reasoner that supports a wide variety of functions including querying, transforming and filtering web content. Cwm also includes plugins that allow it to expand its functionality to

further evaluate rules. It may evaluate mathematical functions, verify digital signatures or look up information on the web. Cwm therefore ensures the aforementioned compliance requirement.

The protocol incorporating N3 and Cwm is built upon the standard HTTP protocol. The user requests a resource using HTTP-GET. If the user is not granted permission to the resource, the client receives an HTTP error code of 401. The benefit of using this error code is that it allows the passing of tokens for new authentication schemes. The N3 rules associated with the resource are sent in the body of the 401 response. The client, using the Cwm, generates a proof demonstrating that it indeed may access this resource. The structure of the proof is specified using a special OWL proof ontology. The Policy-Aware Web developers cite several groups that are defining the syntax and semantics for proof ontologies. One interesting aspect that arises is that a third party may be required for certain justification steps. For example, to prove that a user is a student, the student includes a URI that the requested service uses to verify the student's enrollment at the university. Such an “oracle” needs to be agreed upon and trusted by both parties. Another point of consideration is that the user provides the justification for accessing the resource. By pushing certain computations to other trusted sources, it reduces the size of the trusted computing base of the requesting server.

The Policy-Aware Web certainly improves the transparency and formality in policy enforcement for the Web 2.0. The system should work very well for business to business where entities will be well versed in the necessary requirements for a successful transaction that will meet all industry and legal guidelines. But with respect to the myriad of popular Web 2.0, the Policy-Aware Web may be hard to deploy. First, the user must determine the criteria for access

to a resource (which may not be immediately apparent) and then encode the policy in a language that, although usable for technical individuals, may be difficult for a large, non-technical user base. The expertise to properly encode the user's privacy preferences satisfactorily enough for a court of law may be too great. Another issue is the establishment of oracles. By serving as an oracle in a transaction, there is a legal and ethical responsibility on part of this entity that may exceed the risk the oracle is willing to undertake. Also, it is not clear how to establish the trustworthiness of these oracles without an out-of-bounds verification. The focus of the Policy-Aware Web truly rests on a providing transparent and formal policy enforcement, but does not address the larger issues of acquiring user's privacy preferences or providing them with policy inference to serve as a protective mechanism from unknowingly exposing their privacy.

2.2.5 EUFORBIA

EUFORBIA [57], which stands for "Experiments aboUt the Filtering of internet documents accORding to an unBIAsed and semantic-rich approach", was a project financed by the European Commission to incorporate various advanced technologies to filter out questionable Internet resources. These advanced technologies include knowledge representation, Semantic web and other computer science techniques. The European Commission stated that they expected the project would produce practical technologies. The result of this mandate is the creation of a filtering system known as the Milan Model and a knowledge representation language called Narrative Knowledge Representation Language. Although the systems are multidisciplinary in design, an attribute-based access control system provides the core functionality of filtering system. We will see that this system shares in common access based on

content (which is the basis for our proposal Content-based Access Control in Chapter 3) and specifying concepts using an ontology (much like our tag-based privacy policy language described in Section 4.2).

The prototype built on their Milan Model was MaX, the Multistrategy Access Control System. Administrators assign user credentials relevant to filtering content. Credential values were assigned based on their Credential-Type. Policy authors write rules that either apply to credentials or explicit individuals. Content filters take credentials and identities and grant access to a document based on either its rating or meta-information. Trusted third parties supply ratings to MaX deployments, although the EUFORBIA designers did not disclose the mechanism for ratings acquisitions. Meta data sent along with the document corresponds to ontologies with pertinent vocabularies that are essential to identifying questionable content.

The platform consists of three major components: the filtering module, database and the web interface. The filtering module verifies the validity of each request given the policies in place. To increase its functionality, the filtering module works on a variety of document labeling systems. The EUFORBIA project has defined its own NKRL EUFORBIA label. The filtering module also understands the W3C's Platform for Internet Content (PIC) label. In fact, the filtering module may confer to the PIC labels generated by the Internet Content Rating Association as well as its RSACi rating services. The database is the back end for the system and provides data to both the filtering module and web interface.

The web interface provides system management and user sign on. Management of the MaX system primarily consists of three parts: User, Credential-type and Filtering policies

management. The User management module aides in the creation and assignment of credentials to users. The Credential-type system allows system administrators to define relevant credentials and attributes for the needs of that deployment's filtering policies. Lastly, the Filtering policies management bridges the organizational policies and actual enforcement. These policies are generated by templates that apply the ratings and content label databases to specific credentials. Interoperability between software on client machines and actual policy enforcement is done through a proxy server. That is, for each session the user must use browser-based authentication in order to access resources on the net. By tying down a user to a client machine, the proxy filters successive requests.

EUFORBIA also produced another filtering prototype based on their NKRL language, which provides rule specification and ontologies. This contribution extended ontologies by adding “ontologies of events”. These events correlate to elementary events like the movement of objects, etc. The following example clarifies the benefit of adding actions to ontologies. It may be permissible for students to find websites about lions for a classroom project, but not acceptable to view a site that shows lion's mating or fighting. Sites assign content and media with EUFORBIA labels that categorize the content on the site. This label includes the aim of the site, the site's distinguishing features and an enumeration of subdomains. All labels refer to concepts that belong to an ontology tailored for filtering. Administrators author rules in NKRL with reference to NKRL-specific concepts and ontologies. When requests are made, the site provides the relevant NKRL label and an inference engine decides which rules apply and either

permit the request to go through or stop the transaction. The inference engine in this system uses "classical" backward-chaining techniques.

One last point of interest is the creation of pornography, violence, and racism ontologies. The methods to compile the ontologies included natural language resources like web based glossaries, printed dictionaries, medical sources, web sites, print media as well as focus groups. The authors found certain concepts very difficult to include in the ontologies. Although there are specific concepts that are only applicable in overtly inappropriate domains like pornography, the authors found other concepts that are more prominently used in other domains, had been applied to either pornographic, violent or racist contexts through the use of double entendre. The example of the lions demonstrate such an ambiguity in culminating the ontology.

The EUFORBIA system contains many pieces that would be relevant to improving access control on the Web 2.0. The system designers realized that conceptualizing the access control language by categorizing content would facilitate the appropriate mediation of content to users. The issue, however, is that the categorization must be submitted by the content owner. Self-policing is likely to be fallible. The NKRL language is able to encode concept ontologies, but it may prove too confusing for users and not adapt well to their existing meta-data (e.g. tags). Lastly, the system is inherently a filtering system that is focused on blocking inappropriate content. Therefore, EUFORBIA is not quite an access control system, but a policy language based on concepts may have some promise as a way for users to specify access control.

2.2.6 The Accountability Perspective

This new philosophical approach, advocated by Kagal et al [58], to data usage and privacy preservation focuses on building services that encourage users to conform to policies rather than mediating access control to accessors. Services that adhere to the accountability perspective will assist users in abiding to data usage policies by amply highlighting data usage restrictions as well as the ramifications of misuse and violations. The authors state that these services should possess the three following properties:

- **Give users due notice:** Users are made aware of how their data is being used and have the opportunity to move their data to a service more conducive to their privacy expectations.
- **Authorization and accountability:** Authorization is not enough to limit data loss. Rather, the service should install the necessary architecture to ensure the proper usage of the data. Such a system will require tracking the origination of data, machine-readable policies and the capabilities to reason over policy and transfers.
- **Privacy-enabling interfaces**: Users should have a clear understanding of not only what the policy is with respect to a situation, but also what is the best behavior for users in a given situation (e.g. how I post these pictures of me in questionable circumstances in my news feed?).

The underpinnings of this approach do not lie in constructing an impregnable fortress around one's private data, but to give recourse for those individuals that have their privacy or data usage expectations violated. The authors argue that considering our democratic society, that

problems of this nature (i.e. privacy invasion) are dealt through existing legal systems. To illustrate from previous examples, Ms. Snyder would have instructed anyone viewing her photos that they were for personal use only. Reporting these photos to her superiors would have put the person responsible for bringing it to their attention in violation and could be subject to litigation. The service provider would need to have the mechanisms in place to record that the viewers were properly informed of the data owner's data usage expectations as well as record all accessors, or similarly the service would be subject to possible litigation.

One would be hard pressed to argue the merits of the approach because applying to a governing legal body is the fairest course of action. The issue, however, is that the content owner is now primarily responsible for following up with any punitive damages incurred by a privacy invasion, if they are even aware that one took place! Also, it is not unreasonable to expect that corporations and malicious users will find "loop holes" that may allow them to circumvent the data owner's own policies. Web 2.0 services will also be disincentivized to provide rigorous accountability measures because the services are largely free and adding these technologies would require substantial infrastructure upgrades considering the volume of traffic that these sites generate. Considering the fact that users have expressed the desire for better access control and privacy protection [41], it is unlikely that users will simply accept a pure "accountability" approach. Therefore, this proposal does not provide any suggestions on how we can improve policy acquisition and enforcement. Lastly, will individual users have the financial means, time and energy to prosecute privacy infractions? Perhaps it would be possible for users to create "privacy unions" or organizations that are expressly created for protecting user privacy.

But this is a rather difficult and onerous undertaking for the individual user who simply wants to share photos or post updates about themselves.

2.3 Situations where policy enforcement is minimally useful

Despite the extensive efforts to build access control systems for the Web 2.0 and distributed systems, there are a growing number of situations where the system is unable to enforce privacy preferences of the user or organization. Wikipedia, the online encyclopedia that anyone can edit [59], is a canonical example. Wikipedia allows any visitor to the site to edit any of the articles, with the exception of certain articles that are locked by administrators. This liberal editing policy has allowed Wikipedia to grow immensely, but has invited vandalism, the malicious revising of the text and layout of articles that harms the quality of the articles. Vandalism constitutes roughly 7% of Wikipedia edits [60]. Administrators of Wikipedia have formalized not only what guidelines that its authors should follow, but also what constitutes vandalism. Below are two definitions of what administrators deem as vandalism:

General Policy: *Vandalism is any addition, removal, or change of content made in a deliberate attempt to compromise the integrity of Wikipedia*.

Silly Vandalism - *Adding profanity, graffiti, or patent nonsense to pages; creating nonsensical and obviously unencyclopedic pages, etc.*

These definitions are for the most part clear about what is considered vandalism on Wikipedia and are not written in a manner that is incomprehensible to most Wikipedia editors. But can we easily translate this policy into any of the existing, rule-based formats? There are perhaps simple heuristics that can be employed that will provide good detection such as counting

the number of obscenities in a new revision. These rules, unfortunately, are limited in their efficacy and require the intervention of human editors to revert vandalizing edits. Wikipedia presents a compelling reason to look for new ways to address access control because of the requirements put upon the enforcement mechanism.

Another situation where existing access control does not fully protect its users is enterprise data loss. Enterprises and organizations including governments generate sensitive material that needs to be protected from dissemination to outside parties. The issue here is that not all these sensitive materials are easily described in rule-based formats. This is not to say that all information defies description: Personally-Identifiable Information such as credit card numbers and social security numbers are identifiable by regular expressions. There might also exist phrases or words that exist only within the knowledge domain of the enterprise, e.g., special code names for projects. But limiting access control to this type of data will not sufficiently protect other unstructured data that is found in reports, memos and emails that also needs to be protected. Our investigation of sensitive information in Chapter 6 shows that if such salient features as keywords existed in private documents, we would simply find these keywords, use them as features in our classification approach and provide immediate protection. The main problem, which will be expounded further in Chapter 6 , is that what makes these documents sensitive is not the entities within the documents, which are largely known outside of the organization, but the fact that the enterprise combines them together in such a way that has significance to their operations. For example, in our collection of secret documents from Dyncorp, many common words such as “policy”, “procedure”, and “Afghanistan” appear as the

most informative features. The words themselves are not secretive, but when applied in the same context, the words are more meaningful and specific to the Dyncorp documents.

2.4 Synthesis

This chapter served as an overview of how privacy is protected through access controls across the Web 2.0 and distributed systems. Current Web 2.0 access control across the majority of service providers is draconian and cannot simply model the privacy needs of most users. Although some services like Facebook have begun to introduce group-based access control, users are left largely to protect their own privacy without a safety net. There also exist situations like Wikipedia where although the policy is easy to state, it cannot be encoded in simple rule-based formats.

XACML, EUFORBIA and the Policy-Aware web may provide the essential framework to protect privacy, but these systems have a few drawbacks for usage in the Web 2.0. First, they would require services to not only implement much of the necessary features, but users to become experts in these policy languages and tools. The average user is averse to complicated access control schemes and these technologies would require expertise beyond the capacity of the average user. Therefore, we do not suggest that the technology does not necessarily meet the need for comprehensive access control in many domains, but many protocols advanced in literature or used in industry may be too complex and do not target the main issue that users require a simple, intuitive policy acquisition and automated policy inference to provide a safety net.

P3P, the Social Contract Core and the Accountability perspective move away from access control to compliance. Rather than trying to mediate access, users are protected by other means, e.g., legal protections, because all participants are required to abide to a set of usage rules. One issue is that the end-user could lose out if he is not as savvy as lawyers representing the interests of malicious agents. Users simply may not be able to express their privacy preferences fully through these mechanisms. It is our belief that although accountability and compliance is important, it is not an adequate solution for end users. Accountability, for example, cannot defend against reproduction of content, particularly in different channels. Therefore, we believe access control is still integral to privacy preservation because although it may not necessarily prevent all privacy invasions, it can surely limit the exposure of data.

Chapter 3 Content-based Access Control (CBAC)

The threat to privacy for both individual users and enterprises and organizations can be addressed at the access control level. Current systems, however, leave users *unwilling* or *unable* to employ them to the desired effect.

We advocate that systems employ new paradigms for access control. We introduce in this chapter Content-based Access Control (CBAC). Content-based Access Control mediates access to objects based on the content of the system. Therefore, CBAC integrates content recognition with access control. To understand content in the system, CBAC will use categories or labels to annotate data and learn how to recognize which labels new content should have. A CBAC system will allow users to specify a set of rules R on a set of categories C. For example, with Wikipedia, the system can be specified with two simple rules:

$\{vandalism \rightarrow \text{reject}, not\text{-}vandalism \rightarrow \text{accept}\}$

Therefore, it is the responsibility of the access control system to determine which category a new revision fits into. The user of this type of system (in this case, the Wikipedian administrator) will provide examples of each category. The system will label new contents with one or more categories of from C that will inform which subset of rules to apply from R.

CBAC is not intended to replace existing access control, but rather, enhance it. By providing content-recognition, we can facilitate less cumbersome access control. For example,

as a blogger writes a new post, the system can determine which categories apply to the new post, determine the applicable policy based on the rules of the system and present this to the user. If the system performs the policy enforcement well, the user will have minimal additional work for policy enforcement. For instances where content is difficult to describe, a CBAC system can be trained to differentiate between different categories of content and apply rules based on the user's privacy policy. And since these categories will be inherently abstract, writing policies for them should not burden the user.

The rest of the chapter is structured as follows. We will expand our definition of CBAC in Section 3.1 . We will describe on a high level what an implementation of CBAC entails in Section 3.2 . We will conclude the chapter in Section 3.3 with a discussion of applications of CBAC.

3.1 Definition of a CBAC system

A CBAC system is any access control system in which access to an object is partially or entirely based on the content of the objects in the system [61][62][63]. Note that the policy for one object can depend on the content of other objects in the system. This allows policies that, for example, depend on the content of objects owned by the user requesting access.

There are essentially two requirements for such a system: usable policy specification and automatic policy application. There are already promising results in the design of usable policy interfaces that address the gap between users' mental models and actual policy. Most notably, Karat explored several intuitive user interfaces for policy acquisition, including natural language input, templates, and guides to create machine-readable policies [64]. The template approach

could allow users to succinctly specify natural polices, such as “Allow group College Friends to access entries on topic Parties.” Such policies are short, fit users' mental models, and can be applied broadly to many documents.

Once the policy is specified, we need a mechanism for automatically applying it to existing and new documents based solely on their content, with minimal or no user guidance. The access control system can implement content-based policies by reducing the content to tags on documents and users. To compute these tags, we can exploit several light-weight techniques from machine learning and natural-language processing [65]. These methods extract document meta-information, named entities mentioned in the document, and other text phrases in the document that are statistically likely to summarize its content. Based on these document features, we can infer the appropriate tags of a document by comparing it to other documents with similar features and known tags. Such a system will make occasional mistakes. We therefore will also need a good, easy-to-use feedback mechanism for users to correct erroneous tags.

3.2 Implementation

A CBAC system will comprise of a pipeline that takes user input and will compute which other users will have access to it. For the purposes of this Section, we will focus on blogs. See Figure 1 for an outline of the general CBAC system.

We assume that the system will process a large of number of text objects a day and be limited to lightweight natural language processing techniques. The pipeline will have three general phases: preprocessing, labeling and categorization. Preprocessing will comprise of

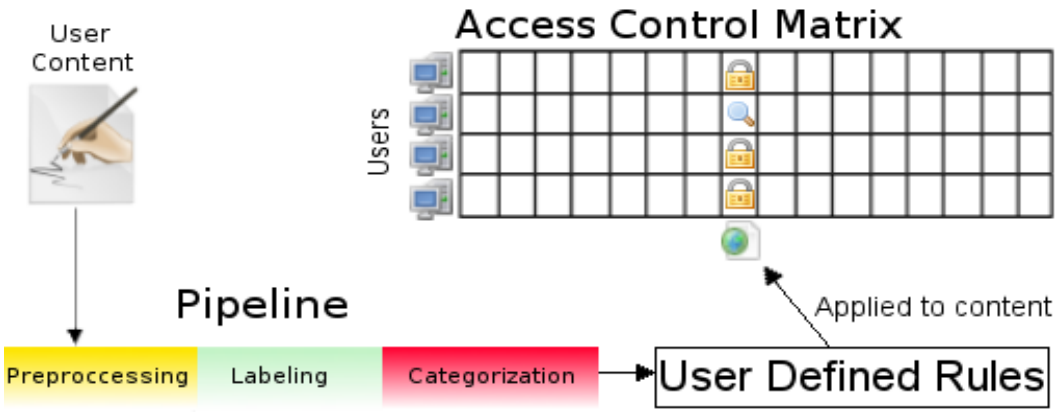


Figure 1: Pipeline in a typical CBAC system.

standard text processing techniques such as tokenization, lemmatization, part-of-speech tagging, stop word removal, emoticon identification, etc. These steps facilitate named entity recognition, the discovery of people, locations, organizations, events and date/times. We hope to build a database of entities for individuals using information from their social networking accounts (e.g. Facebook) as well as lists for publicly known entities from knowledge bases like Wikipedia. The system uses coreference to identify different lexical representations of the same entity. The last step of preprocessing is find the similarity between text objects. This will allow us to identify groups of related documents, which presumably share the same content and topics.

Labeling assigns to preprocessed text objects a uniform set of features, called labels, that will be used for categorization. The labels represent significant and diverse characteristics of the text objects: document meta-information, important text phrases, named entities and membership in a group of similar documents. These labels allow us to in turn assign categories to text objects by learning which labels correlate to certain categories. Categories may be individual to the user or applicable across the system. Standard machine learning techniques will

be used to learn and classify categories based on sets and frequencies of labels. The categories are used to cluster text objects and base access decisions by the users. Content authors can choose the categories using intuitive access control tools to grant other users permission to content objects [66].

Another facet of our implementation will be engendering interoperability among web services to provide richer access control. It would be unreasonable to assume that all of a user's potential viewers belong to a particular service. It may also not be expedient for someone to join a service such as a blogging site since it requires remembering another online identity. Therefore, to enrich access control, there is a desire to be able to identify web surfers by other accounts they own. This would expand the ability of access control policies since content owners are able to include individuals that they desire to grant access to whom do not have accounts with the services. This may also influence users from providing too lax of control to include desired viewers. In turn, this provides a richer experience for both content providers and content readers since access is better mediated and more content is available. There is also a wealth of semantic data that may be applicable to other domains. For example, Facebook allows users to create online groups as well as determine membership to real life groups. Importing this data to other services will create less burden on users such that they do not have to restructure their social groups each time they join a new service.

3.3 CBAC applications

We have successfully applied Content-based Access Control in three different domains: blogging, Data Loss Prevention and Wikipedia Vandalism. For blogging, we developed a policy

language based on tags that we present in Chapter 4 and a policy inference engine in Chapter 5 . Enterprises are concerned more than ever with data loss prevention after high profile leaks, particularly after ones as large and having as great of impact as the State Department cables leak on WikiLeaks [67]. For organizations and government, much of what is confidential exists in unstructured text where categorizing it defies simple approaches such as regular expression matching, fingerprinting and keyword searching. In Chapter 6 we discuss building text classifiers for Data Loss Prevention that help categorize outgoing messages as either *secret* or *non-secret*, enabling the access control system to handle these messages appropriately. Wikipedia states what constitutes as vandalism, but their definitions cannot be described simple rule-based formats. Therefore, we show in Chapter 7 how a classifier can be trained to recognize vandalizing text.

Chapter 4 Privacy/Policy-aware Blogging

Bloggers have been subject to serious consequences as result of their posts [12]. Many blogging services have attempted to solve address these privacy invasions producing a mish-mash of incompatible privacy features. With the exception of one blogging service, users are forced to either manage password-protected posts (i.e. distributing passwords to viewers) or limit their blog to be viewed by a select audience.

In this chapter, we demonstrate that ***tag-based*** privacy policies are a usable and flexible privacy control method for Web 2.0 applications. We present tag-based privacy controls that (1) are easy for users to understand, (2) flexible enough to express users' privacy preferences, and (3) fit naturally into users' current work flow. Content owners express their privacy policy in terms of the tags on content objects; the system then applies the policy to objects based on the tags assigned to them. Data we have collected from real blogs show that tags correlate well with privacy preferences. We have created a prototype implementation as a plugin to the WordPress blogging system, and conducted a user study to measure whether (1) users can efficiently implement privacy preferences using tag-based privacy controls, and (2) users would choose to use tag-based privacy controls over other available options. Participants in our study found our tag-based privacy controls easy to learn and use. Despite limited time, training and familiarity with the blogging system, our results show:

-

- Participants who wrote tag-based policies were just as accurate as those who set per-post policies.
- Participants using tag-based policies finished tasks significantly faster than than those using the per-post policy tool.
- Participants wrote tag-based policies of nearly optimal size.
- A third of study participants chose tag-based controls, despite having only a few minutes of experience with tag rule authoring.

The chapter will begin with a brief overview of current approaches to access controls specifically for blogs. Section 4.1 will then introduce the our policy language and its relationship to how users already tag their private posts in Section 4.2 . Our implementation of PLOG for WordPress will be detailed in Section 4.3 . We will describe our experiment in Section 4.4 and present the results in Section 4.5 . A discussion follows in Section 4.6 with related work afterwards in Section 4.7 .

4.1 Privacy schemes in blogs and social-networks

Users clearly want control over the private information they publish on Web 2.0 applications [68][69]. We surveyed the privacy features provided by seventeen blogging and social-networking sites. All the sites we surveyed offered some form of privacy control (please refer to Section 2.1 for a full list of access controls), indicating **who** has access to **what** content objects.

Current access control systems on the Web 2.0 make an implicit trade-off between simplicity and flexibility. For example, friends-based privacy controls attempt to make it easier for the user to specify ***who*** can see private information by offering only three options: everyone, no-one, or all the user's friends. Profile-based controls try to simplify the task of identifying ***what*** information of a given type (e.g. photos, status, contact information, etc.) is private. These rigid policy models may be easier to use, but they cannot meet the needs of all users. Rigid models may choose an inappropriate level of granularity (e.g. friends-based privacy controls) or may group objects or subjects in a way that does not correspond with the user's privacy preferences (e.g. with profile-based controls, photos from school and work may be grouped together, but have different intended audiences).

Users need a way to specify the privacy of objects that is both flexible and integrated into normal Web 2.0 activities. Tag-based privacy tools use the multiple levels of granularity of tags to create flexible privacy policies without incurring the management costs of complex schemes, such as password-protected posts. In the next section, we formally define tag-based policies and validate our decision to use tags as a basis for access control decisions by evaluating real-world data.

4.2 The Plog policy language

Tags are words or phrases that are paired with objects (e.g. blog posts, photos, videos). Content owners author their own tags; consequently, tag usage can be quite varied. Tags often indicate content topic, enumerating topics both broad (i.e. “golf”) and narrow (i.e. “Tiger Woods”). Tags can also indicate other content properties such as form (e.g. “poem” or “photo”),

type (e.g. “journal article”, “conference paper”), location (e.g. “Sri Lanka”), or origin (e.g. “from my window”, “from Sally”). Finally, tags may be used humorously or in other idiosyncratic ways. Because tags often correspond to topics, they are a promising basis for a privacy policy language. However, not all tags are relevant to the user's privacy policy, and content owners can be inconsistent in their application of tags. Consequently, owners should be able to over-ride the general privacy rules for some objects.

To formalize the Plog policy language, let O be the set of objects belonging to a content owner. In a blog, these objects would correspond to posts; in a social networking site, these objects could correspond to blog posts, photos, personal details, videos, etc. For each object $o \in O$, let T_o be the set of tags the user has assigned to that object. Let V be the set of all potential viewers, and let $v_0 \notin V$ be a special “anonymous” viewer. Viewers may be identified by email address, user ID, or some other method – the details are irrelevant to the policy language. Visitors must authenticate to access restricted content. Unauthenticated viewers have the access rights of the anonymous viewer.

The user can specify a set of tag privacy rules $R \subseteq T \times 2^V$, where an entry (t,V) indicates that viewers in set V can see objects assigned tag t. Given the tags, T_o, on an object, o, the set of applicable tag rules is $R'_o = \{(t,V) | t \in T_o\}$. In Plog, the content owner may disable some of these rules on a per-object basis, so access control decisions are actually made based on the owner-specified set $R_o \subseteq R'_o$. By default, $R_o = R'_o$, so owners only need to explicitly specify R_o in exceptional situations. The owner may also manually grant or deny access to a particular object.

For each, object, the user may specify a set $A_o \subseteq V$ of allowed viewers, and a set $D_o \subseteq V$ of denied viewers. By default, $A_o = D_o = \varnothing$.

The set of viewers allowed to see object *o* is:

$$V_o = \begin{cases} \bigcup_{(t,V) \in R_o} V \cup A_o \setminus D_o & \text{if } R_o \cup A_o \cup D_o \neq \varnothing \\ V \cup \{v_o\} & \text{otherwise} \end{cases}$$

Because this algorithm takes the union over all tag rules, it grants access to any viewer who is a member of V for some tag rule $(t,V) \in R_o$. An alternative implementation could choose to take intersections in this case, i.e. access would be granted only to viewers who are members of (t,V) for all tag rules $(t,V) \in R_o$. We chose union because, in our system, the user can disable tag rules he does not want applied to a particular object.

When no policies apply to an object, i.e. $R_o = A_o = D_o = \varnothing$, then the object is world-readable. In the context of blogs and social networks, this seems more appropriate than a default deny policy. This approach also ensures that, if an anonymous viewer can view an object, then so can all other viewers. It does not make sense to grant access to anonymous viewers but deny access to an authenticated viewer, because the denied viewer could circumvent the privacy controls by viewing the object anonymously.

This privacy policy language is very flexible. It also makes policy management simple in the common case. Once the content owner has created a few tag rules, the system will apply the

privacy policy to new objects as they are created. Only in a few exceptional cases will the owner wish to override the default policy.

Rules in this language are evaluated at time-of-access. This makes it easier to understand and reason about the policy language and for the content owner to retroactively adjust the policy on existing objects.

In our implementation, content owners can also define groups of viewers and use these groups in tag rule authoring. The name of the group is replaced with the list of its members for the purpose of policy evaluation.

4.2.1 Tag usage on existing private posts

A good privacy policy language should enable users to express their preferences with a few simple rules. We evaluated the Plog policy language by translating privacy settings on existing blogs into our language and measuring the size of the resulting policies.

We begin by defining the "size" of a Plog policy. Recall that with Plog, the user defines a set of generic tag rules, R, and a set of per-object policy exceptions $E = \{(R_i, A_i, D_i)\}_{i=1}^{n}$, where, by default, $R_i = R'_i$ and $A_i = D_i = \emptyset$. The user must explicitly add items to A_i and D_i, and remove items from R_i. A user can easily disable multiple rules at once, so a natural measure for the size of a Plog policy is

$$S(R, E) = |R| + \sum_{(R_i, A_i, D_i) \in E} \epsilon(R'_i \setminus R_i) + |A_i| + |D_i|$$

where $\epsilon(X)$ is 1 if $X \mathrel{!}= \emptyset$ and 0 otherwise.

We used a screen-scraper to download 377 blogs with private posts from the WordPress blogging service. The WordPress software hides the bodies of private posts, so our screen-scraper could not collect that information. Curiously, WordPress does not hide the existence of private posts or the tags assigned to them. Thus, for each blog, our screen-scraper generated a set $\{(p_i, t_i)\}_{i=1}^{n}$, where p_i is the privacy setting of the ith post (either "public" or "private"), and t_i is the set of tags on that post. Since each post must be declared private separately, the size of a privacy policy for a WordPress blog with n private posts is simply n.

Our screen-scraper discarded private posts that had no associated tags. We suspect that many users choose not to tag some private posts because, as mentioned before, WordPress does not hide the tags of private posts. Since the tags themselves may reveal important details about a private post, in some cases the user has no choice but to leave the post untagged. If it weren't for this quirk of the WordPress blogging system, we suspect users would exhibit the same tagging behavior on these posts as on their other private posts. Thus discarding them should not skew the results of our analysis.

Given an output $\{(p_i, t_i)\}_{i=1}^{n}$ from our screen-scraper, we constructed a Plog policy as follows. Let $v_1 \in V$ represent a viewer that should be able to see private posts. We implemented a policy solver that constructs a set of tag rules, R, and per-post exceptions $E=\{(R_i, A_i, D_i)\}_{i=1}^{n}$ such that each private post is visible only to v_1, each public post is visible to v_1 and v_0, the anonymous viewer, and $S(R,E)$ is minimized. Constructing such a policy is equivalent to the NP-complete weighted hitting-set problem [70] so our solver uses a brute-force branch-and-bound

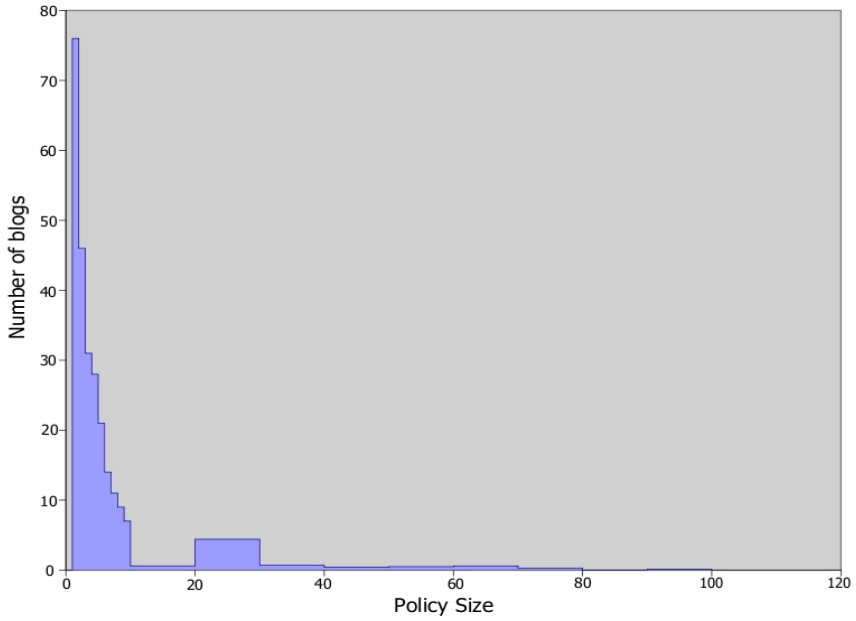


Figure 2: Histogram of optimal policy sizes for our WordPress private blog dataset.

Policy 1: private, relationships
Policy 2: parenting, tragedy, love, family
Policy 3: issues, melhoras, luv-issues

Figure 3: Three sets of tags for policies on real world data generated by our solver.

algorithm. The solver reduces the search space by removing tags that occur more often on public posts than private posts. If, after this reduction, the blog still contains more than 16 tags in consideration, the solver falls back to a greedy algorithm to approximate the optimal policy. Only three blogs in our dataset exceeded this bound.

Figure 2 shows a histogram of the policy sizes generated by our solver. Over half the bloggers could express their privacy policy in just 4 rules using Plog, and 75% could express their privacy policy using only 9 rules. A small percentage of bloggers had huge tag-based

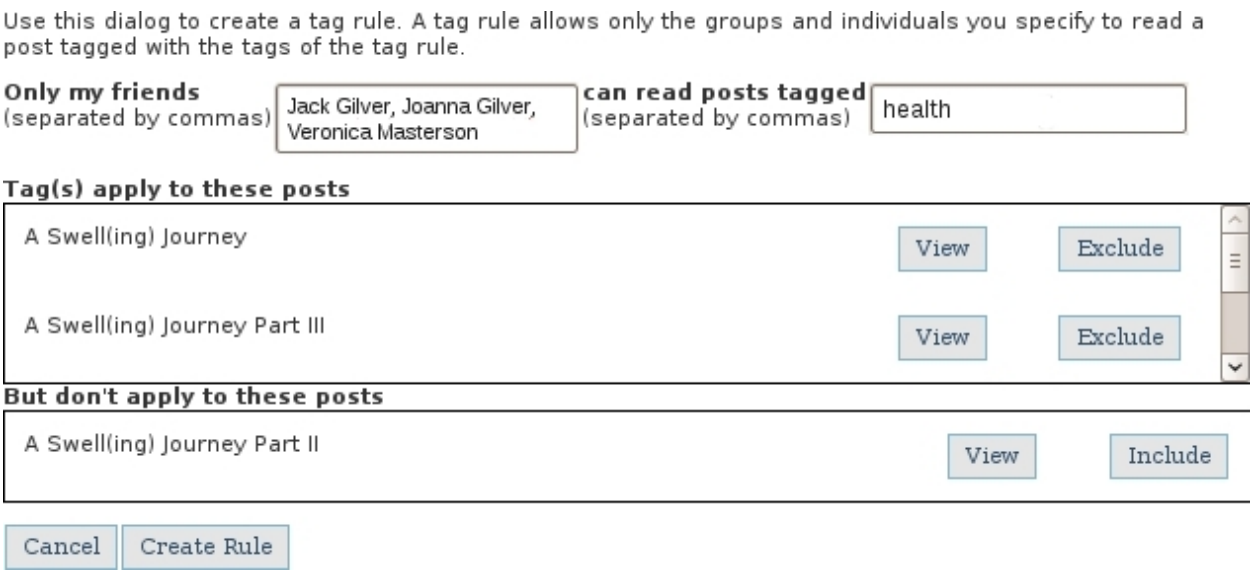


Figure 4: The Plog tag rule authoring interface.

privacy policies. These bloggers use tags so haphazardly that tag-based rules offer them no benefit. Although these bloggers derive no benefit from tag-based policy rules, but they are no worse off, either. Figure 3 shows three sets of tags for policies generated by our solver.

This analysis may overestimate or underestimate the size of policies in a deployed Plog system. Bloggers using a Plog-enabled system would have an extra incentive to tag their content consistently, which would result in smaller Plog policies and a more organized blog. On the other hand, we may have underestimated Plog policy sizes because the the solver generated policies that only distinguished public and private posts, whereas the actual policies may have granted different viewers access to different posts. Unfortunately, WordPress only indicates whether a post is public or private, so we could not obtain the true policy.

Despite the limitations of this analysis, the results suggest that tag-based privacy controls would dramatically reduce the effort required to maintain a blog privacy policy. Since users

already tag private posts, they could benefit from tag-based privacy policies immediately. In fact, many users could express their privacy preferences in less than 5 Plog rules.

4.3 Plog WordPress plugin

We have implemented Plog as a plugin to the WordPress blogging system [71]. The plugin consists of three main components: the tag rule authoring interface, the per-post policy editing interface, and the user and group management interface.

4.3.1 Tag rule authoring interface

Content owners author tag privacy rules by filling in templates, as shown in Figure 4. While the owner is authoring or editing a tag rule, the interface provides a list of the posts that would be affected by the rule. This helps owners write accurate rules and gives them an opportunity to add exceptions along with the new rule. If the owner does not recall the content of an object, she can view it in a pop-up window. The tag rule authoring interface is similar to the SPARCLE "structured authoring page", although our implementation uses text fields with auto-completion instead of check-boxes or natural language [66].

4.3.2 Per-post policy editing interface

A content owner may create policy exceptions for an object while editing the object by using the policy editing interface as shown in Figure 5. The Plog plugin displays the list of viewers that can see the current object next to the "Publish" button. This encourages owners to consider the appropriate privacy policy for each object before publishing it, and gives the owner a chance to double-check the policy inference engine's results. If the owner wishes to make an

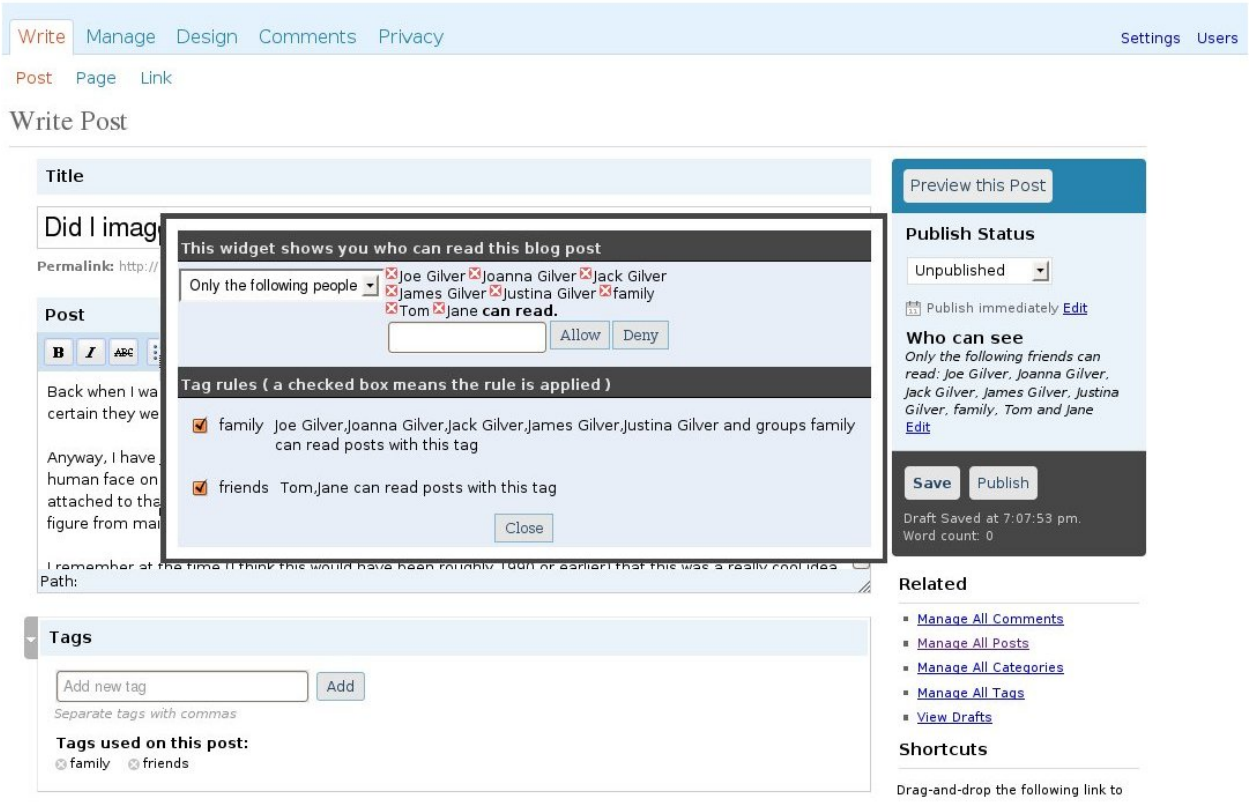


Figure 5: The Plog per-post policy editing interface.

exception to the policy for this object, she can click on the "Edit" link next to the viewer list. This produces the dialog in the center of Figure 5.

The dialog displays three meta-states that the owner can set for an object. The "Everyone" meta-state declares that any authenticated can see the object. The "Friends" meta-state declares that only viewers who have at least one credential proving they are friends with the owner can see the object. The owner can exclude friends in this meta-state, which makes sense in a scenario where only a subset of friends should have access (for example, if the object is an invitation to a surprise birthday party for a friend). Lastly, the "No-one but" meta-state permits the owner to create a list of authenticated viewers who alone will have access to the object.

Unless the user explicitly over-rides the tag-based policy for this post, this widget updates the post's privacy policy as the user edits the tags. The user can click “Edit” to see an explanation of the inferred policy for the post and to over-ride it if necessary.

A content owner's friend may not be not be a fellow user of the site, but the content owner still needs a way to identify the friend in his privacy policy. With the Plog plugin, a content owner can identify viewers by their Plog user ID, OpenID URL, Gmail username, AOL instant messenger ID, MSN username or Facebook ID, among others. Collectively, these mechanisms cover a significant percentage of the Internet user population [72], and this percentage will only increase as OpenID and other single sign-on services become more common. As an added convenience, Plog users may “import” lists of friends from these other services to organize their friends into groups that reflect their real-life social structure. When a viewer visits a Plog user's blog, the viewer sees a link at the top of the page inviting him to identify himself. Since a single viewer may be identified in multiple ways by different Plog users -- e.g. one Plog user refers to the reader by his Gmail address, another Plog user refers to the reader by his AOL IM name -- a viewer may login using multiple IDs. The first time the viewer does this, he will have to authenticate for each ID. Plog infers that all the given IDs correspond to the same viewer and, in the future, the viewer only needs to login using one of his ids.

4.4 Evaluation

We performed a user study to (1) test whether users could leverage tag rules to enforce privacy constraints, and (2) test whether users naturally preferred tag rules over per-post policies. The experiment and results are discussed in the following sections.

4.4.1 Participants

Twenty-eight undergraduate and graduate students were recruited as participants for this study (Male = 18, Female = 10, Mean Age = 21.17). All participants were at least 18 years old, fluent English speakers, and registered students of the university. None were WordPress users.

4.4.2 Experimental design

Each participant completed five training tasks and thirteen experimental tasks in which they had to role play as Ted, a blogger. The experimental phase included both privacy-related tasks and "distractor" tasks that covered normal blogging activities. The distractor tasks provide a baseline for comparing performance between participants and enable us to compare the usability of Plog and the built-in WordPress features.

In order to add realism to our tasks, we used a real blog released under the Creative Commons license, which we anonymized by replacing person and place names. This blog contained posts written by a married couple with compelling life stories. Topics discussed included struggles with kidney disease, the death of a father, and photos of family, which motivate genuine privacy concerns. So that our participants could complete the experiment in a reasonable amount of time, we shortened each blog post to a maximum length of 400 words and reduced the total number of posts to thirty. We did not modify the tags on any posts.

We used two deceptions to reduce participant bias. First, the participants were told that they were evaluating the WordPress open-source blogging system for learnability and usability. Since the participants were led to believe that we had not designed any part of the WordPress system, they had less incentive to withhold criticism, try to "get the right answer", or praise the system. Second, the distractor tasks obscured the privacy focus of our study. This reduced bias between participant performance on privacy versus non-privacy tasks.

4.4.3 Training

Participants in our study were not WordPress users, so each participant first completed five training tasks as an introduction to the WordPress system. Each participant read the entire blog (thirty posts), wrote a blog post, managed posts, used the per-post privacy control tool, and used the tag rule privacy control tool. The system randomized the order of the two privacy training tasks for each participant. This allowed us to detect whether participants were influenced towards the privacy control presented to them first.

To be confident in participants' understanding of the software, we evaluated participant performance in the training phase. All participants successfully completed most of the training tasks. Nine participants did not successfully complete the training task that involved post authoring and setting a per-post privacy policy. Five of these participants failed to specify any privacy policy for this task, but were able to successfully use the per-post privacy policy tool in the first privacy-related experimental task. The other four were excluded from our experimental analyses.

As you have read, Ted has severe kidney problems that has caused him issues with his family, insurance and work. Ted realizes that he would like to make posts on his medical issues only readable to his family because he does not think it is wise to share his frustration and isolation with future employers and strangers. Determine which posts are about his medical and insurance issues and make them readable only by his family (Joe, James, Joanna, Justina and Jack Gilver).

Figure 6: Text for Task 15.

Task	Description
7	Visit Tanya's blog and leave a comment on a post
9	Change the links listed on the side of the blog
12	Import Ted's friends using his email account
16	Change the theme of the blog
17	Add the pages, calendar, links, and recent posts widgets to Ted's blog.

Figure 7: Distractor tasks and descriptions.

Task	Privacy Concern	Posts	Total tags	Minimal Tag-based Policy	Optimal Policy
10	Church community posts	2	47	One tag rule, one exception	Tag-based policy
11	Memorial to father	1	18	One tag rule	Per-post policy
13	Work related posts	2	28	One tag rule	Tag-based policy
14	Daughter sees author naked	6	21	One tag rule	Per-post policy
15	Health problems	6	21	One tag rule	Tag-based policy
18	Humor and opinions	4	48	Two tag rules	Tag-based policy

Figure 8: Details on privacy tasks and their optimal solution.

4.4.4 Tasks

Each participant completed thirteen experimental tasks. In the distractor tasks, listed in Figure 7, the participant performed activities unrelated to privacy control, such as modifying the blog design, commenting on a post, managing the blog posts, and reading other blogs on the site. In the privacy tasks, listed in Figure 8, the participant was asked to restrict readership of topically

related posts to a group of viewers (note that tasks 1-6 were training tasks and tasks 8 and 20 were transitions). Each privacy task expressed privacy preferences as generally and plainly as possible to model realistic privacy concerns, and did not suggest a solution strategy. Participants were allowed to solve the tasks using either tag rules, per-post specifications or a combination of both. An example of a privacy task is shown in Figure 6. The intended viewers of the privacy preference model real life social groups including family, church friends, work friends and college friends.

For the privacy tasks, the level of difficulty in authoring tag rules varied greatly because of the usage of tags on the topically related posts. Tag rules were not always the optimal strategy for solving privacy tasks because there were instances where the minimal set of tag rules plus exceptions exceeded the amount of work done using the per-post tool. Figure 8 describes the optimal solution for each privacy task.

4.4.5 Procedure

The user study took place in computer teaching labs over the course of a week across multiple sessions. Participants were asked to give two hours of their time and were compensated with 20 dollars. Participants were allowed to ask questions; the experimenter performed troubleshooting if necessary. All participant interface actions (e.g. viewing a page, saving a policy, or using an auto-completion feature) were logged. Following the user study, each participant completed a fifteen-item questionnaire rating the effectiveness, learnability and ease of use of the major functions of the blogging system on a seven point Likert scale.

4.5 Results

4.5.1 Tag rule usage

Eight of the twenty four participants (the *tag-rule* group) used tag rules to solve five of the seven privacy-related experimental tasks. The remaining sixteen participants (the *per-post* group) used the per-post tool to complete all of the privacy-related experimental tasks. Four participants in the tag-rule group and four participants in the per-post group completed the tag rule training task before the per-post training task. The other sixteen participants saw the tag rule training task after the per-post training task. We conclude that participants' strategy choice was not dependent on the order in which participants were trained in the system's privacy features (p=0.36, Fisher's Exact Test).

4.5.2 Accuracy on privacy tasks

For each post *o*, let V_o be the set of viewers granted access by the participant, and $\tilde{V}_o$ be the correct set of viewers. A participant's accuracy on a post *o* was computed as:

$$A(o)=\begin{cases} 0 & \text{if } v_0 \in V_o \wedge v_0 \notin \tilde{V}_o \\ 1-\frac{V_o \triangle \tilde{V}_o}{|V|} & \text{otherwise} \end{cases}$$

Each post is associated to a single task, so for each participant, we can divide posts into a tag-rule set and a per-post set, depending on how the participant attempted to solve the task. We then compute an overall accuracy for each method by averaging, across all participants and tasks, the accuracy on posts in each set. These overall accuracy scores were compared using independent two-tailed Welch's t-tests. For all posts related to a privacy task, the accuracy on

Task	Accuracy					Time (Seconds)				
	Tag rules		Per-post rules		Significance	Tag rules		Per-post rules		Significance
	N	Mean (SD)	N	Mean (SD)		N	Mean (SD)	N	Mean (SD)	
10	7	98.10% (0.3)	17	84.51% (0.29)	n.s.	7	153.14 (37.37)	18	205.71 (58.78)	$p < 0.05$
13	8	87.10% (0.16)	16	83.13% (0.31)	n.s.	8	87.88 (36.93)	16	157 (77.65)	$p < 0.01$
15	8	88.10% (0.13)	16	87.32% (0.14)	n.s.	8	189.0 (78.71)	16	303.44 (92.23)	$p < 0.01$
18	8	96.25% (0.09)	16	84.10% (0.32)	n.s.	8	129.75 (47.30)	16	249.5 (105.11)	$p < 0.001$
21	6	39.33% (0.23)	18	70.37% (0.32)	$p < 0.01$	6	173.67 (54.11)	18	348.39 (144.90)	$p < 0.001$

Table 2: Accuracy and time completion for experimental privacy tasks.

posts where policies were set by tag rules (M =0.82, SD =0.12) were just as accurate as those posts where policies were set by the per-post policy tool (M =0.82, SD =0.13; $t(350.10)=1.97$, $p > .97$).

We also computed a per-task accuracy for each method in the same way, but only considering the posts relevant to each task. Accuracy scores were compared using two-tailed Welch's t-tests. Results are shown in Table 2. We could not perform this comparison for tasks 11 and 14 because too few participants used tag rules. For all posts related to tasks 10, 13, 15, 18, the accuracy of the tag-rule group was not significantly different from that of the per-post group. However, the per-post group achieved higher accuracy for task 21. This is because many of those who wrote tag rules for this task assumed the tags they choose, "photos" and "photography", applied to all the posts in this category, but actually the posts were inconsistently tagged.

4.5.3 Accuracy on distractor tasks

We also computed accuracy for the six experimental distractor tasks. For each task, the participant received a score of 1 if the task was completed successfully, or 0 if it was not. As before, we compared these accuracy scores using two-tailed t-tests. We found that the tag-rule

Task	Accuracy					Time (Seconds)				
	Tag rules		Per-post rules		Significance	Tag rules		Per-post rules		Significance
	N	Mean (SD)	N	Mean (SD)		N	Mean (SD)	N	Mean (SD)	
7	8	100.00% (0)	16	100.00% (0)	n.s.	8	131.13 (40.10)	16	138.15 (29.76)	n.s.
9	8	100.00% (0)	16	87.50% (0.34)	n.s.	8	97.86 (12.97)	16	135.09 (26.58)	$p < 0.01$
12	8	100.00% (0)	16	100.00% (0)	n.s.	8	112.00 (34.77)	16	124.14 (32.38)	n.s.
16	8	87.50% (0.35)	16	81.20% (0.25)	n.s.	8	50.38 (15.93)	16	98.14 (38.13)	$p < 0.01$
17	8	75.00% (0.46)	16	68.75% (0.48)	n.s.	8	42.13 (20.10)	16	77.40 (55.37)	n.s.
19	8	100% (0)	16	93.75% (0.25)	n.s.	8	131.67 (64.05)	16	177.93 (73.45)	n.s.

Table 3: Accuracy and time completion for experimental distractor tasks.

group and the per-post group performed equally well on the distractor tasks overall. A detailed analysis is provided in Table 3.

4.5.4 Time to completion of privacy tasks

We computed time to completion (in seconds) for each group for all privacy-related experimental tasks. We compared these times using two-tailed Welch's t-tests. When participants used tag rules, they completed the privacy tasks significantly faster than those using per-post policies. Results are shown in Table 2.

4.5.5 Time to completion of distractor tasks

We computed time to completion (in seconds) for each group for all distractor experimental tasks presented in Table 3. The data reveals a small subset of participants in both groups that took noticeably longer to complete these tasks. Since participants did not receive training to complete these tasks, it is likely that some participants had unusual difficulty due to non-task-related reasons, such as being unable to find a link in the interface. Therefore, outliers were removed for both the tag-rule and per-post groups by computing the interquartile range and

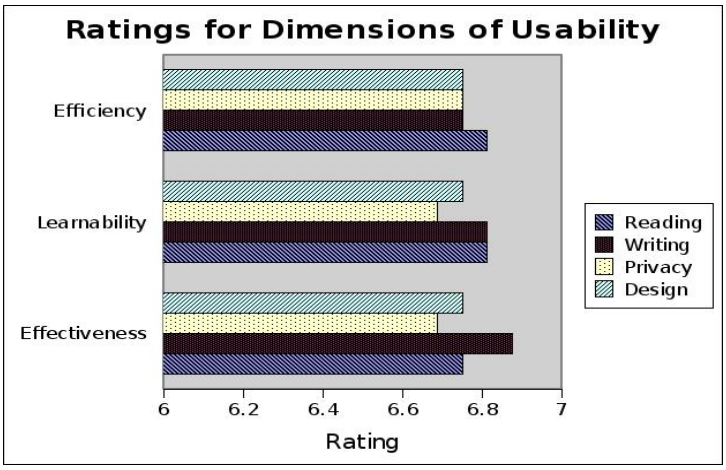


Figure 9: Average participant ratings of the major Plog and WordPress features.

removing for each task those participant's times either less than or greater than in their respective group.

We then compared task completion times using two-tailed t-tests. Participants in the tag-rule group and the per-post group did not significantly differ in the time to complete distractor tasks 7, 12, 17 or 19. Participants in the tag-rule group completed task 9 and 16 significantly faster than those in the per-post group.

4.5.6 Subjective evaluation results

Participants rated fifteen blog features on a seven point Likert scale for three aspects of usability: effectiveness, efficiency and learnability [73]. Figure 9 reports the average scores. Eight participants failed to complete the entire questionnaire and were eliminated from this part of the analysis.

We performed ANOVAs on each dimension of usability by comparing the averages of the Likert ratings for the four major blog features (reading, writing, privacy and design) and

found no statistical differences (Effectiveness $F(3,13) = 0.53$, $p = .66$; Learnability $F(3,13) = 0.30$, $p = 0.82$; Efficiency $F(3,13) = 0.08$, $p = 0.97$).

We compared the scores for each usability dimension for the tag rule group and the per-post group using two-tailed t-tests. We found no significant differences in the ratings for any dimension of usability.

4.5.7 Size of tag rule policies

In this analysis, we only consider participant data for tasks that the participant completed with perfect accuracy and by using tag rules. On average, participants' tag rule policies came within one tag of the optimal policy. Half of the perfectly accurate tag-based policies created by our participants were optimal. The mean difference in size between the optimal and participant policies was 0.92 with a standard deviation of 1.38 and Q_3=1.

We also looked at all tag-based policies regardless of accuracy. Participants in general did not write redundant tag rules (those that if removed from the policy, would not reduce the number of posts covered). On average, the tag-based policies created by our participants contained 0.79 redundant tags (SD =1.28).

4.6 Discussion

Our evaluation results show that:

- Participants in the tag-rule group wrote policies that were just as accurate as those written by participants in the per-post group.

- Participants in the tag-rule group were able to author privacy policies with minimal effort, significantly faster than those in the per-post group.
- Participants wrote near-optimal tag-based privacy policies in terms of size.

We conclude that content owners in Web 2.0 applications can benefit from tag-based privacy control. Unlike the participants in our user study, users of these applications are not novices, and are familiar with the content they own. Consequently, they should be able to write tag rules even more efficiently and accurately than our study participants. Since the resulting policies will be smaller than per-object privacy control policies, they will also be easier to review and maintain. In fact, our findings in Section 4.2.1 of this chapter indicate that real-world users could immediately see a reduction in the effort required to maintain their privacy online.

In our evaluation, we made a deliberate choice not to force study participants to use either tag-based or per-post privacy policy authoring. We did not advertise the benefits of tag rules. Nor did we provide any guidance or training in tag rule authoring, which requires the mental ability to abstract away from post content and reason from precondition to effect. Given that our participants were blogging novices, it is very promising that a third of them were able to use and prefer the abstraction of tag-based policy authoring in a short period of time. For the other two-thirds of our participants, the distractor tasks do not indicate they were less skilled or slower at completing other blogging tasks than the tag-rule group. Therefore, we think that these participants in the per-post group would also prefer tag-based policies if the time and effort savings were made clear to them, and if tutorials and help were provided.

We are also encouraged by the finding from our subjective evaluation that participants thought the privacy tools were as usable as the other functions of the blogging system. The most frequent recommendation to improve the system given by the participants was to include more themes. Therefore, privacy tools in blogs are not antithetical to usability nor do they require extensive training.

The results of this user study show two areas where we could improve our implementation. First, the Plog interface should provide automatic assistance to identify redundant rules to further help bloggers keep policies small. Second, sometimes tag rules are not the best solution if the tags are ambiguous or overloaded. Task 21 is an example of a situation where tag rules are not the optimal solution, because the target posts were not consistently tagged. Two members of the tag-rule group observed this and solved the task using the per-post tool. Plog could identify inconsistently-tagged posts and suggest alternative tags to make it easier to author tag rules.

4.7 Related work

In Role Based Access Control (RBAC), access to content is determined based on a user's roles in a particular system [74]. RBAC can be successfully used in highly structured contexts such as company intranets.

In Attribute Based Access Control (ABAC), access to content is determined based on a user's attributes (e.g. age, home town) [75]. This approach is promising when all content viewers are authenticated viewers. Plog policies may have some similarity to RBAC or ABAC policies,

but RBAC and ABAC systems must be administrated manually and usually by a security expert, which would be infeasible in large and dynamic Web 2.0 applications.

Although much research has focused on aspects of access control such as policy enforcement [55], policy reasoning engines [76], and distributed access control systems [44], few comprehensively address the policy authoring problem. IBM's SPARCLE project [66] developed several tools to help trained experts translate human-readable corporate privacy policies into machine-readable form. Our user interface for specifying privacy preferences is similar to SPARCLE's structured policy authoring interface. Expandable Grids is another notable system in policy authoring [77]. This system produces visualizations of access control policies in an interactive grid format. This work has been successfully applied to managing file permissions and comprehending P3P privacy policies. The Grey project [78] highlights the importance of modeling user's ideal policies and demonstrates that systems closely aligned with user preferences will result in less errors than other approaches where users must translate their constraints in unnatural terms

The Plog tag-based policy language is similar to Domain Specific Languages (DSL) [79]. DSLs are designed to provide a solution to a specific problem that takes advantage of the well defined and clear language of a particular domain, which can be mapped into a machine-readable form. Tags are incredibly varied and individual in their usage and do not constitute the formal and universal language required of a DSL.

Systems that use the content and categorization of an object to render access control decisions like Plog exist in specialized capacities. Adult content filters are a special instance of a

topic-based access control system [80]. Most notable in this category is the "MaX" system of the EUFORBIA project [81], which uses content meta information to enforce an attribute-based access control. Other researchers have investigated techniques for classifying objectionable images based on their content to prohibit accepting their submission [82]. Some research projects take advantage of the formal language and organized nature found in business and legal documents to translate policies and rules into machine readable form [83][84].

4.8 Conclusion

The Web 2.0 has given users unbelievable opportunities to create content and share with others. Unfortunately, the lack of privacy in most Web 2.0 applications has had significant impact on some users' lives. Developers of these applications have a responsibility to give users the tools to manage their own privacy, but it is hard to create general tools that a wide variety of users can use effectively.

Our solution, Plog, is a tag-based privacy policy control that allows users to mediate their competing desires for privacy and publicity by specifying their privacy preferences in a language that is natural to them. Tag-based policy authoring produces policies that are short, precise, and easy to create and maintain. Our implementation of Plog includes several user-interface features designed to make privacy management as simple and non-intrusive as possible and could be easily integrated into current Web 2.0 applications.

Plog offers a starting point for more intelligent and interactive privacy control in the Web 2.0. By utilizing OpenID and similar technologies, these policies could be made to "stick" through transportable policy data encodings and mechanisms since the privacy policies we create

are easily converted into machine readable form using technologies like [85]. We argue that systems like Plog can elucidate privacy policies from the user's content and categorization which could engender better privacy control across the Web for personal content.

Chapter 5 Policy Inference for PLOG

Choosing tags as the basis for our policy language has the additional benefit that we can provide policy inference by suggesting tags on newly created content. We built iTag as a policy inference tool for PLOG. To determine its predictive performance, we test our tag suggestion algorithm on real blogs to get a sense of how well it will perform in the general case.

5.1 Tag suggesters for blogs

Despite the popularity of blogs and prevalence of tagging, the most popular blogging services do not offer tag suggestion features. Researchers have developed tools for tag suggestion, but most of these tools focus on social tag prediction. Social tags are the most interesting or informative tags derived (via aggregation, scoring and filtering) from all tags assigned by users of an online community (e.g. del.icio.us, StumbleUpon, Digg) to a content item. Social tags allow members of the online community to share and interact more effectively tag [86]. They do not, however, accurately reflect the mental model of the content author.

There are two general approaches used in tag suggestion tools for blog posts. The first extracts interesting terms from the post itself [87]. This approach is useful for post clustering, but often author-assigned tags consist of terms not in the post itself. Bloggers may label their posts with category tags ("Categories" in WordPress), topical tags, dates, and locations. For example, on the widely-read "Get Rich Slowly" blog [88], there are some category tags (e.g.

"Administration", "Ask the Readers"), and some topical tags (e.g. "Cars", "Credit Cards"). The second approach uses search and scoring over a large collection of posts not necessarily belonging to the blogger [89]. For example, the TagAssist tool tags a post by constructing a search query from the post content, searching a collection of blog posts using the query, extracting the author-assigned tags from the retrieved posts, and scoring and filtering those tags [90]. In a series of evaluations on blog data, TagAssist was shown to perform well. However, it is neither personalized nor localized: it weights tags preferentially if they occur on more popular posts or assigned to a larger number of posts, but it does not give preference to posts or tags by the same author, or to posts or tags that occur near each other in time.

In this work, we present a tag suggestion tool that focuses on personalized and localized tag recommendation. Our method suggests only tags previously used by this blogger. It also incorporates temporal information when selecting tags to suggest. Our experiments demonstrate that a personalized tagger can substantially outperform a non-personalized tagger. We compared the performance of iTag and TagAssist on a random subset of posts drawn from the "Growing Blogs" WordPress RSS feed. This feed includes blogs that have had a recent increase in popularity (compared to its average number of page views), and therefore tends to contain reasonably well-written posts and few "spam blogs" (i.e. blogs that are created to artificially boost search engine results or strictly advertise a product or company). On this data set, iTag achieved precision and recall scores over 60%, while TagAssist scored below 10% in these measures.

The rest of this chapter is structured as follows: in Section 5.2 we present what is to the best of our knowledge the first analysis of tag usage on a per-blogger basis. In Section 5.3 we describe a novel and lightweight machine learning technique for tag suggestion that uses blogger identity and temporal information as features. In Section 5.4 we describe an evaluation of our approach and compare its performance with that of TagAssist. We observe dramatic increases in both precision and recall on tagging posts, even when tags that only occur once are taken into consideration. This suggests that personalization must be incorporated into an auto-tagger to provide effective tag recommendations.

5.2 Tag usage on individual blogs

We investigated tag usage on individual blogs by downloading 1246 blogs from the WordPress "Growing Blogs" feed [91] between April 1 and April 25. We chose WordPress because it allows bloggers to separate tags into two types: tags and categories. Intuitively, bloggers may separate their tags so that the more frequent or taxonomic tags are designated as categories. 84.85% of the blogs in our data set utilized categories. The "Growing Blogs" feed [91] collects blogs that have experienced a relative surge in page views over the last 24 hours. We chose this feed because it rarely contains spam blogs, is generally well written, exhibited tremendous diversity in topic and content, and similar to other blogs in the long tail of popularity.

We measured the frequency of use of each tag and category by individual bloggers, then aggregated across all the blogs we analyzed. The resulting distribution of tag frequencies is shown in Figure 10. As one would expect, bloggers reuse categories more frequently than tags.

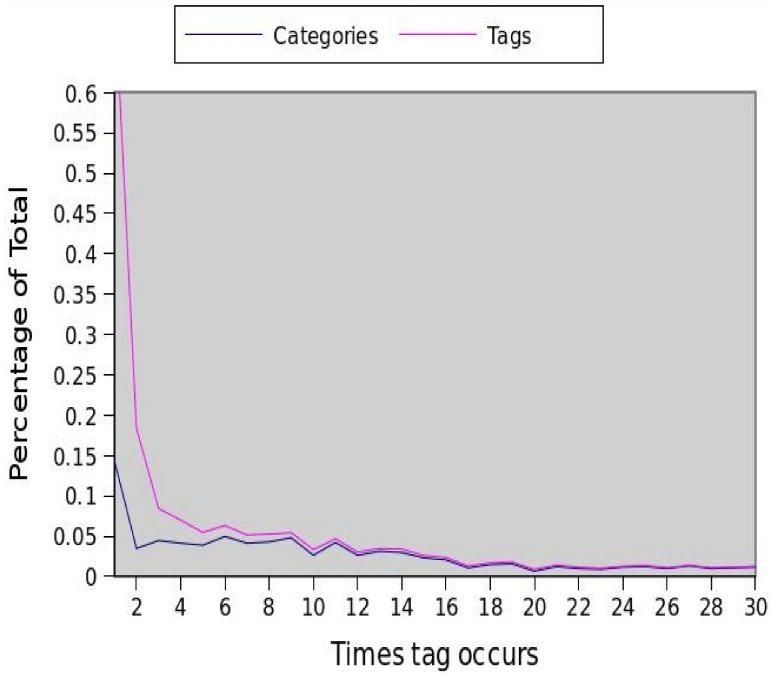


Figure 10: A distribution of tag and category occurrences across all blogs.

We note that most tags occur only once per blog (58.82%). On the other hand, 51.2% of categories occur more than ten times per blog. On average, bloggers assign 10.2 tags to each post, but only 2.2 categories. Although most tags occur only once, most posts contain 9.37 previously seen tags, i.e. all but one of the tags are re-used. Bloggers also reuse 2.03 categories on average, introducing a new category about once every 6 posts. 81.4% of posts use only pre-existing categories, whereas 23.8% of posts use only pre-existing tags.

We also measured individual bloggers' consistency in tag use, as this is an indicator of how well a localized tag suggestion tool can perform. For each post, we determined the minimum number of previous posts such that the set of tags aggregated from these previous posts provides total recall for the tags that have occurred at least once before and are assigned to the target post. If bloggers use tags haphazardly, we would expect many posts to be required to

Approach	Average Number of Posts	Recall
All previous posts	1.38	1.0
5 most similar posts	1.20	0.91
10 most similar posts	1.27	0.94
10 most recent posts	1.22	0.92
20 most recent posts	1.28	0.95

Table 4: Average number of posts required for maximal recall on previously seen tags.

provide total recall. For this analysis, we selected a subset of 100 blogs that each contained 50 to 150 posts.

Finding the minimum number of posts to provide total recall is an instance of the set-cover problem, and therefore we employed a linear programming solver. We investigated several different strategies for previous post set selection. First, we selected all previous posts regardless of order. Second, we selected the *n* most recent previous posts (by creation date order). Third, we selected the *n* most similar previous posts (based on the cosine similarity of the documents).

Results are shown in Table 4. We see that for most bloggers use of tags is highly local, both with respect to topic and with respect to time.

Our analysis of tag usage on individual blogs suggests that a personalized and localized tag suggestion tool can be highly effective. First, more than 40% of tags and 85% of categories are reused by individual bloggers. In fact, the majority of tags and categories on each post have been used before. Our data suggests that a local approach can have good recall even though most tags are unique and it suggests only previously used tags. Second, a relatively small post history (10 posts) gives high recall for previously used tags, indicating that bloggers cluster related posts together temporally. At the same time, a high recall for previously used tags based solely on

similar posts by the same blogger indicates that document similarity can continue to be a useful feature for personalized tag suggestion

5.3 iTag description

iTag uses a modified form of the approach taken in TagAssist [90]. We start by summarizing TagAssist's search and score strategy. We then highlight key differences in iTag.

5.3.1 TagAssist

When TagAssist is presented with a target post, it selects tags to suggest using a search and score method on a large set of training data.

Training Data: TagAssist uses a corpus of blog posts and tags indexed with Lucene [92]. As part of indexing, TagAssist normalizes tags by trimming white space and punctuation, stemming each word in each tag, and ordering the words in multi-word tags alphabetically. It then clusters the normalized tags using tag co-occurrence information to find the minimal number of sets of semantically-related tags.

Tag Retrieval: TagAssist generates a query of up to 30 unigrams and bigrams from the target post that have high TFIDF scores in its training corpus. It retrieves up to 35 result posts from its index for this query, and retrieves the tags for each result post. Tags that occur on only one result post are discarded.

Tag Scoring: TagAssist scores the retrieved tags using a weighted sum over the following features: frequency (in the bag of retrieved tags), text occurrence (in the target post), tag count (frequency in the training corpus), rank (popularity of the blog containing the retrieved

post labeled with the retrieved tag), and co-occurrence with other retrieved tags (in the training corpus).

Tag Selection: TagAssist suggests tags that score above the average of all tag scores.

5.3.2 iTag

iTag also uses a search and score method, but significantly differs from TagAssist in each part of the system.

Training Data: iTag only considers the previous posts, and consequently only the tags, of the blogger.

Tag Retrieval: iTag generates a query in the same way that TagAssist does. However, iTag only retrieves the 10 most similar previous posts by this blogger (which achieved recall of 0.94 in the analysis described previously). iTag does not remove any retrieved tags from consideration, regardless of their absolute frequency of occurrence.

Tag Scoring: iTag does not use the rank, tag count, or co-occurrence features used by TagAssist. Instead, it uses the following features which we have found provide the best information for tag suggestion with respect to previous posts:

- **Count:** the number of times the retrieved tag appears in the retrieved tag set.
- **Highest rank:** the rank of the retrieved post labeled with this retrieved tag that is most similar to the target post.
- **Contained:** 1.0 if the retrieved tag appears in the target post, 0.0 otherwise.

- **Tag.IDF:** a variation on tf-idf, this feature is computed as the number of times the retrieved tag appears in the retrieved tag set, multiplied by the blogger's inverse document frequency for the tag.
- **Last Recently Used:** distance in number of posts since this tag was last used by this blogger. This feature draws on the work of Cattuto et al.[93] by rewarding tags that have occurred recently, and is motivated by the analysis in Section 5.2 .

Tag Selection: We devised two methods for tag selection. Our first method, adaptive co-occurrence tag selection, is a modification to the TagAssist tag selection method. Our second method, classification-based tag selection, uses a binary classifier trained on the features described above.

Adaptive Co-occurrence Tag Selection: In this mode, iTag suggests any tag that scores above average. It also suggests any tag ***a*** that strongly co-occurs for this blogger with a tag ***b*** that scores above average, as long as

$$\frac{co-occurrence(a,b)}{min(count(a),count(b))} \geq 0.35$$

Our testing indicates that a co-occurrence threshold of 35% provides good tag suggestions. Every time a blogger makes a change in tag assignments, the co-occurrence frequencies for that blogger are updated.

Classification-Based Tag Selection We apply binary classifiers, using the above features, for tag suggestion using the C4.5 decision tree implementation provided by WEKA [94]

(decision trees outperformed SVMs, Naive Bayes, and nearest neighbor classification algorithms for our task). The label for each tag was 1 if the tag was applied on the post, and 0 otherwise.

iTag takes two approaches to classification-based tag suggestion. In the *pre-trained* approach, we trained the classifier on 15,000 instances of tags from the tag retrieval step of our algorithm on posts from our WordPress data set, regardless of blogger of origin. Since the features of the instances are relate to the search set results and presence in the post and not to the tag or author specifically, we hypothesize that the behavior of applied tags behaves similarly across blogs. In the *locally-trained* approach, by contrast, we train a separate classifier for each blog. The pre-trained approach is faster since the classifier only has to be trained once. The locally-trained approach must be retrained each time the blogger makes a change in tag assignments. Note that even when using the pre-trained classifier, iTag is still personalized since it only chooses tags from the blogger's previous posts.

If the classification-based method produces no tag suggestions, iTag backs off to the adaptive co-occurrence method.

5.4 Evaluations and results

Our evaluation uses the same set of blogs described in Section 5.3 . We eliminated blogs that contained fewer than 30 posts, since these blogs were likely created by new bloggers not yet familiar with post authoring and tagging. We eliminated posts that contained fewer than 10 non-stop words because they had little information or contained only photos or links. We did not distinguish between tags and categories, but included a tag only once if it was assigned as both a category and tag.

Tag Frequency	All Tags	Adaptive	Classification-based		TagAssist
		Co-occurrence	Locally-trained	Pre-trained	
Every tag on the post	0.34 (0.77)	0.48 (0.57)	0.67 (0.59)	0.64 (0.56)	0.02 (0.07)
Occur more than once	0.41 (0.84)	0.52 (0.62)	0.67 (0.63)	0.66 (0.60)	0.02 (0.08)

Table 5: Precision and recall for tag suggestion method with respect to 1000 tagged WordPress posts from April 2009. "All Tags" refers to the entire set of tags returned by our localized search results.

We implemented TagAssist to compare with our approach. We trained our implementation using the ICWSM 2009 data set [95]. This data set consists of 6.9 million posts with 1.7 million unique tags and 1.4 million TagAssist normalized tags. We chose this data set because it is the only freely available data set we could find with the popularity information and size required by TagAssist.

We set aside 400 randomly-selected blogs for testing data. We used the remaining blogs to create the pre-trained classifier. When evaluating the locally-trained classifier on a post, we trained it on all the preceding posts in the same blog.

As testing data, we used the 400 blogs set aside earlier. We then sampled 1000 posts from the last 20% of posts in each blog (with respect to creation date), with no blog contributing more than 3 posts.

For tag suggestion, we report average per-post precision and recall across all 1000 testing posts. Precision and recall for each post are normalized using the number of tags applied to the post (for recall) and suggested (for precision). We report results separately for all tags, and for tags the blogger uses more than once. Our evaluation results are presented in Table 5.

5.5 Discussion

iTag achieves high precision and recall for posts, even when considering tags that occur only on the target post. This indicates that it is a useful method for tag suggestion which could be deployed almost immediately.

Both classification-based approaches outperform the adaptive co-occurrence method. Surprisingly, the pre-trained approach performs almost as well as the locally-trained approach. We note that precision and recall for tags that occur more than once when the classifier suggested at least one tag were 77.88% and 67.77% for the locally-trained approach and 79.17% and 63.36% for the pre-trained approach. This means that although the classification-based method gives higher precision, it may fail altogether, and the adaptive co-occurrence method gives some robustness.

iTag performs substantially better than our TagAssist implementation in terms of both precision and recall. However, the precision and recall scores of our TagAssist implementation are less than half those reported by the TagAssist creators. There are several possible explanations for this. First, many of the tags suggested by our TagAssist implementation were reasonable. However, as our tag usage analysis showed, tag assignment is highly personal. Second, in the original TagAssist evaluation, a set of contemporaneous blogs was used for training and testing, and some of the same blogs were used for training and testing. We did not include other posts by the same blogger or from the same time period in the training corpus for our TagAssist implementation. Third, the blogs used in the original TagAssist evaluation came from Technorati and appear to be news and technology oriented, while many of the blogs in our

data set were personal and had a wider variety of themes. Fourth, many normalized tags occurred relatively rarely in the ICWSM data set (63% of the tags on the test posts occurred on 10 posts or fewer in the ICWSM data set). Consequently, the removal of tags that occurred only once in the retrieved tag set may have adversely affected the precision and recall of TagAssist. Curiously, we also found that TagAssist's tag normalization yielded clusters that where tags were not semantically related. For example, "Cheney", "Cheetos", "Chi" and "Children" all normalize to "ch".

Both iTag and TagAssist may face problems of scale. For TagAssist, the problem is related to storing tag co-occurrence information. The 1.7 million tags in the ICWSM data set yielded 20 million instances of tag co-occurrence. Without aggressively caching this data, searching for co-occurrence data can be incredibly costly. For iTag, the problem is related to storing classifiers or adaptive co-occurrence features for each blog; however, the tag co-occurrence problem is reduced when only local tag co-occurrences need to be stored. Also, with the pre-trained approach, we can alleviate the necessity to store training data for each blog; however, the tag co-occurrence problem is reduced when only local tag co-occurrences need to be stored.

5.6 Future work

It is clear that although the purely local approach performed well, it is limited in two crucial ways. First, it suffers from the "cold start" problem where initial data is required to obtain any results. Secondly, an individual blog is a relatively small corpus containing at most a thousand entries. A local approach inevitably encounters sparsity of data when tagging new

posts. This experiment demonstrates that a purely global approach may not take advantage of the personal taxonomy and consistency in tagging behaviors of bloggers. Our goal is to combine global and local information to provide personalized suggestions that also anticipates new tags. Lastly, existing tag suggestion systems have not taken advantage of user interaction and feedback. It will be important to perform a user study to observe user interaction with auto-taggers and how better suggestions can be crafted.

5.7 Conclusions

We have described iTag, a personalized and localized tag suggestion tool motivated by analysis of bloggers' post tagging behavior. We have demonstrated that iTag outperforms taggers trained on large multi-blog, multi-tag data sets. We also demonstrated that incorporating machine learning leads to improved tag suggestion with minimal cost.

Chapter 6 Data Loss Prevention

Modern enterprises increasingly depend on data sharing, both inside and outside their organizations. Increased sharing has led to an increasing number of *data breaches*, i.e., malicious or inadvertent disclosures of confidential and sensitive information, such as social security numbers (SSN), medical records, trade secrets, and enterprise financial information, to unintended parties. The consequences of data breach can also be severe: violation of customers' privacy, loss of competitive advantage, loss of customers and reputation, punitive fines, and tangible monetary loss. The Ponemon Institute's 2009 *Cost of a Data Breach Study* found that a data breach costs an average of $6.6 million to an organization [96]. The Privacy Rights Clearinghouse lists almost 500 million records that have been leaked in data breaches since 2005 [97].

Security vendors have begun to offer a raft of "Data Loss Prevention" (DLP) products designed to help businesses avoid data breaches [98][99][100][101][102]. DLP systems identify confidential data on network storage servers, monitor network traffic and output channels to peripheral devices such as USB ports, and either enforce data control policies or generate reports that administrators can use to investigate potential breaches.

Although existing DLP solutions are quite sophisticated in detecting, capturing and assembling information flows, they are currently limited in their capability to recognize sensitive information. Many vendors offer solutions that rely on keywords, regular expressions and

fingerprinting, but these techniques alone cannot fully capture the organization's secrets when it is re-phrased or re-formatted. More elaborate and comprehensive human annotations and access control will not solve the problem because they rely on users to encode in a machine-readable form the sensitive contents of the message. This is simply infeasible for certain types of data, too time consuming and too error prone. Security vendors now recognize the need for DLP systems to learn and automatically classify sensitive materials [99].

In this chapter we develop practical, accurate, and efficient machine learning algorithms to learn what is sensitive and classify both structured and unstructured enterprise documents as either public or private. Our scheme is practical because enterprise administrators need only provide an initial set of public and private documents. Our system trains a classifier using these documents, and then uses the resulting classifier to distinguish public and private documents. System administrators do not have to develop and maintain keyword lists, and our classifier can recognize private information, even in documents that do not have a substantial overlap with previously-observed private documents.

We summarize the results of our classifier on 5 testing corpora in Section 6.4 and compare the results with a baseline off-the-shelf classifier (Section 6.2). Our classifier achieves an average false positive rate of 0.46% and an average false negative rate of 1.6% on our testing corpora. The classifier also achieves a much lower false discover rate (FDR), i.e., the percentage of false alarms raised by the classifier, than the baseline classifier. A low FDR is essential since users will ignore a system that frequently raises false alarms. If we assume a typical enterprise network (Section 6.4), then our classifier has an average FDR rate of 0.47% compared to the

baseline classifier's average FDR rate of 16.65%. These results demonstrate that our classifier can meet the demanding needs of enterprise administrators.

In summary, this chapter makes the following key contributions to the field of enterprise data loss prevention:

- We demonstrate that simply training a classifier combining enterprise data, both public and private, yields prohibitively high false positive rates on non-enterprise data, indicating that it will not perform well in real networks.
- We present a new algorithm for classifying sensitive enterprise documents with low false negative rates and false positive rates. This algorithm employs a new training technique, supplement and adjust, to better distinguish between sensitive, public and non-enterprise documents. Our algorithm scales to real time enterprise network traffic and does not rely on any meta data.
- We construct the first publicly available corpora for evaluating DLP systems.

The rest of the chapter is organized as follows. We briefly describe a typical DLP system in Section 6.1 and discuss how our classifier fits into a DLP system. We introduce our classification algorithms in Section 6.2 and describe our test corpora in Section 6.3 . We discuss our classification results in Sections 6.4 and 6.5 . In Section 6.6 , we compare our work with related work. We conclude with a summary and possible avenues of future work in Section 6.7 .

6.1 Data Loss Prevention systems

In this section, we describe a typical DLP system's building blocks and discuss how our proposed approach fits into the system. A DLP system aims to protect three types of data in an enterprise: *data-at-rest*, *data-in-motion*, and *data-in-use*. Data-at-rest is static data stored on enterprise devices such as document management systems, email servers, file servers, network address storage, personal computers, and storage area networks. Data-in-motion is enterprise data contained in outbound network traffic such as emails, IMs, and web traffic. Data-in-use is data being "used" by the enterprise's employees on end point devices, e.g., a file being copied to a USB drive.

Let us consider the definition of confidential for an organization. There certainly exists certain types of data such as Personally Identifiable Information, e.g., names, credit cards, social security numbers, that should be confidential regardless of the organization. The definition becomes more difficult to articulate, however, when we consider trade secrets and internal communications, which may be unstructured. Broadly, we define *secret* as information generated within the organization that is either not general known, e.g., facts that can be found in an encyclopedia or industry magazines, or contained in public materials from the company. A DLP system will include some functionality to identify sensitive information in one or more of the aforementioned data types.

A DLP system performs three broad steps to prevent enterprise data loss. First, the system discovers the three types of enterprise data by scanning storage devices, by intercepting network traffic in real time, and by monitoring user actions on end point devices. Second, the

system identifies confidential enterprise data from the data discovered in the first step. Third, the system enforces enterprise policies on confidential data. For example, the system may encrypt confidential data-at-rest to prevent unauthorized use; the system may block confidential data-in-motion from leaving the enterprise and may prevent confidential data from being copied to a USB device.

A DLP system faces two operational challenges: performance and accuracy. In an enterprise setting, the system should scan terabytes of data-at-rest, monitor hundreds of megabytes of real time network traffic, and monitor user actions on thousands of end point devices. The system should identify confidential data accurately in a scalable manner without producing many false positives or false negatives.

Current DLP products identify confidential data in three ways: regular expressions, keywords, and hashing. Regular expressions are used primarily to recognize data by type, e.g., social security numbers, telephone numbers, addresses, and other data that has a significant amount of structure. Keyword matching is appropriate when a small number of known keywords can identify private data. For example, medical or financial records may meet this criteria. For less structured data, DLP products use hash fingerprinting. The DLP system takes as input a set of private documents and computes a database of hashes of substrings of those documents. The system considers a new document private if it contains a substring with a matching hash in the database. Regular expressions are good for detecting well-structured data, but keyword lists can be difficult to maintain and fingerprint-based methods can miss confidential information if it is reformatted or rephrased for different contexts such as email or social networks.

It is also unlikely that more sophisticated access controls and additional user annotation will necessarily improve DLP products. First, it is likely that most sensitive materials contain a fair amount of public knowledge. Former analysts of the C.I.A. have noted that only 5% of intelligence was captured through covert actions, meaning that 95% of information in these reports is derived from public sources [103]. Therefore, assigning the privacy level to text copied and pasted from such a document is not guaranteed be the correct action. Relying on the users themselves to better identify and police sensitive materials poses several complications. Users may find encoding sensitive material to not be trivial. Even if the user has the ability to sufficiently define what is confidential in this system, it is possible for the user to forget or make a mistake. Lastly, it may not be feasible to expect that all users annotate their content consistently.

In this chapter, we propose automatic document classification techniques to identify confidential data in a scalable and accurate manner. In our approach, the enterprise IT administrator provides a labeled training set of secret and non-secret documents to the DLP system instead of keywords and regular expression. We *learn* a classifier from the training set; the classifier can accurately label both structured and unstructured content as confidential and non-confidential. The DLP system will use the classifier to identify confidential data stored on the enterprise devices or sent through the network.

Our approach builds on a well-studied machine learning technique, Support Vector Machines (SVMs), that scales well to large data sets [104]. The classifier can meet an enterprise's needs ranging from a small collection of a user's sensitive material to a large

enterprise-wide corpus of documents. We assume that the DLP system cannot access to meta data associated with documents, e.g., author, location, time of creation, and type. We also assume that administrators will only provide the document classifier with labeled training data. Employees and managers, therefore, can provide confidential documents directly to the classifier, alleviating the burden of collecting a training set on IT administrators and minimizing their exposure to confidential information.

The major drawback of confidential data identification schemes used in DLP systems, including ours, is the inability of these systems to classify data they do not "understand." Encrypted data and multimedia content are examples of such data. Loss of confidential data via encryption is relatively rare in practice: only 1 out of more than 200 data breaches use encryption [105]. Hence we leave the challenges of identifying confidential data in encrypted content and multimedia content as future work.

6.2 Text classifiers for DLP

This section will discuss present our approach for building text classifiers for Data Loss Prevention. It will begin by discussing the types of data it will encounter with respect to prominence and privacy. We will then describe our baseline approach for performance comparison. We will conclude the section with our approach to building text classifiers for DLP.

Enterprise networks and computers handle three types of data: public enterprise data, private enterprise data, and non-enterprise data. Public enterprise data (*public*) includes public web pages, emails to customers and other external entities, public relations blog posts, etc. Private enterprise data (*secret*) may include internal policy manuals, legal agreements, financial

records, private customer data, source code or other trade secrets. Non-enterprise data (*NE*) is everything else, and so it cannot be described succinctly, but is likely to include personal emails, Facebook pages, news articles, and web pages from other organizations, some of which may be topically related to the business of the enterprise. We consider private documents to be confidential and require protection whereas *NE* and *public* documents do not. From this high-level description, we can draw several conclusions:

- Enterprise public and private documents are likely to be relatively similar since they discuss different aspects of the same underlying topics.
- Many non-enterprise documents will share almost no features with enterprise documents.
- Some non-enterprise documents may be quite similar to enterprise public documents. For example, non-enterprise documents may include news articles about the enterprise or web pages from related organizations.

A DLP text classifier is thus faced with two contradictory requirements: it must be finely tuned to enterprise documents so that it can make the subtle distinction between public and private documents that discuss the same topic, but it must not overfit the data so that it can correctly mark non-enterprise documents as public. As explained below, our solution uses a two-step classifier to solve this problem. The first step eliminates most non-enterprise documents that have little in common with enterprise documents, and the second step uses a classifier focused on documents related to the enterprise to make the finer distinction between enterprise public and private documents.

6.2.1 Baseline approach

We are not aware of any previously published results on text classification for DLP. We also could not test our solution against existing DLP solutions because we could not verify if the software adhered to the constraints our classifier abides to, i.e., no meta-data is associated with documents). We first developed a baseline classifier to provide a basis for comparison and to garner insight into the structure of the DLP text classification problem.

We performed a brute search evaluating multiple machine learning algorithms and feature spaces known for their text classification performance for our baseline classifier, including SVMs [104], naive Bayesian classifiers [106], and Rocchio classifiers [106] from the the WEKA toolkit [107] to determine the best classifier across all the datasets. We found that a support vector machine with a linear kernel performed the best on our test corpora (described in Section 6.3). The best performing feature space across all corpora is unigrams, i.e. single words, with binary weights. We eliminated stop words, common words such as “is” and “the”, and limited the total number of features to 20,000. If a corpus contained more than 20,000 unique non-stop words, we choose the 20,000 most frequently-occurring non-stop words as our features. We use this configuration as our baseline classifier for all experiments reported in Section 6.4 .

An SVM trained on enterprise documents achieves reasonable performance on enterprise documents, but has an unacceptably high false positive rate on non-enterprise (*NE*) documents. The poor performance can be explained by identifying weaknesses in the training approach. First, for two of our testing sets, the classifier was biased towards the *secret* class, e.g., its initial expectation was most documents to be secret. And since many *NE* documents share very few

features in common with secret documents, the classifier mislabeled these instances because it had too little information to contradict its a priori expectation. The second issue arose from overfitting of features. The public documents could not alone capture the behavior of these features for non-*secret* documents. It will, therefore, overweight certain features; we noticed common words like "policy" and "procedure" being instrumental in the misclassification of *NE* documents.

A DLP text classifier is thus faced with two contradictory requirements: it must be finely tuned to enterprise documents so that it can make the subtle distinction between *public* (non-*secret* enterprise documents like press releases) and *secret* documents that discuss the same topic, but it must not overfit the data so that it can correctly mark *NE* documents as non-*secret*. We address this problem using a two-step classifier. In the first step, we train a classifier on both enterprise and non-enterprise documents. This classifier better distinguishes *secret* documents from *NE* documents, but does increase the *public* false positive rate. We apply a second classifier that addresses mislabeled instances from the first step by checking to see if the instance is topically related to *secret* documents and comparing it to a classifier specialized to recognize *public* documents.

6.2.2 Supplement and Adjust

To remedy overfitting and overweighting common features, we *supplement* the classifier by adding training data from non-enterprise collections such as Wikipedia [59], Reuters [108], or other public corpora. As we will show in Section 6.4 , our supplemental corpus does not need to

be comprehensive. The presence of supplementary data does not train the classifier to recognize *NE* documents, but prevents it from overfitting the enterprise data.

We use 10,000 randomly-selected Wikipedia articles and a 1,100 document set featuring documents on finance, law and sport as our supplementary data set. We labeled the supplementary articles as *public* during training. The supplement classifier uses the same feature set as the baseline classifier and does not include features found in the supplemental data set. This prevents the classifier from using words from the supplemental data set to learn to distinguish *secret* and *NE* documents.

Adding supplemental training data will likely introduce a new problem: class imbalance. Supplemental instances will bias the classifier towards *public* documents because the size of this class will overwhelm the size of *secret* documents. This will result in a high false-negative rate on *secret* documents. Therefore, we need to adjust the decision boundary towards *public* instances. This will reduce the false negative rate while increasing the false positive rate. For our classifier, we measure the distance between the decision boundary and the closest, correctly classified *public* instance (either *NE* or *public*) and move the boundary x% of the distance towards it, for some value of x. We chose $x = 90\%$, although we show in 6.5 that our classifier is robust and performs well when $50\% \leq x \leq 90\%$.

The supplement and adjustment technique can be applied to train classifiers tailored to both *public* and *secret* documents, with the supplemental instances in both cases drawing from the same source, e.g., Wikipedia. Therefore, we denote a supplement and adjust classifier as SA_{class} where class is either *public* or *secret*. When training an SA_{secret} classifier, we combine

public and *NE* documents and adjust the boundary to the closest, correctly classified *public* or *NE*. An SA_{public} classifier is constructed by combining *secret* and *NE* documents and adjust the boundary to the closest, correctly classified *secret* or *NE* document. We employ an SA_{secret} classifier as the first stage of our DLP text classification system.

6.2.3 Meta-space classification

The first-level classifier significantly reduces the number of false positives generated by *NE* documents, but not completely. These documents tend to contain salient features of the *secret* class, but upon further inspection, clearly unrelated topically to confidential documents. Also, the number of false positives for *public* documents increases. Therefore, we apply a second step to eliminate false positives from documents labeled *secret* by the first step.

We address these remaining false positives in three different ways. First, for a target document, we will measure how similar it is to either the *secret* or *public* set of documents. Second, we build classifiers specifically tailored for the *public* class. *Secret* and *public* documents will likely overlap in content since they are topically related and may even discuss the same entities employing similar language. Therefore, our system will attempt to learn what combination of features make these documents *public* rather than *secret*. We can use the output of this classifier in conjunction with the first step to better gauge if a document should be labeled *secret* or not. Lastly, we classify the target document based on the output of the similarity measures and classifiers (hence why we refer to this classifier as a "meta-space" classifier). We use three classes (*public*, *NE*, *secret*) instead of two classes (*secret*,¬*secret*) for this step. Three classes assist the classification of *secret* documents because *NE* false positives exhibit different

behaviors than *public* false positives for these features, making classification much more difficult if we group *NE* and *public* together.

To address the problem of topically unrelated documents being labeled as *secret*, we created two attributes, $xtra.info_{secret}$ and $xtra.info_{public}$, that measure the percentage of words in a document that do not appear in any document from the *secret* and *public* training corpora, respectively. These features are intended to measure the overall dissimilarity of a document, *d*, to documents in the *public* and *secret* corpora. For example, if *d* has a large value for $xtra.info_{public}$, then it is very different from documents in the *public* training corpus. We can improve the *xtra.info* features by ignoring words that occur commonly in English and hence convey little contextual information. We compute for each word *w* an estimate, df_w of how often *w* occurs in "general" English documents. We can then ignore all words that have a high df_w value. We used 400,000 randomly-selected Wikipedia articles to estimate df_w for all words across all our training sets. If a word in our training set never occurred in our sample of Wikipedia, we assigned it a frequency of $\frac{1}{400,000}$. We then computed

$$xtra.info_c(d) = \frac{d_{df} \setminus \cup_{d' \in c} d'}{|d_{df}|}$$

where $d_{df} = \{w \in d | df_w \leq df\}$. In our experiments, we used $df = 0.5\%$.

The $xtra.info_{secret}$ attribute aides the classifier by giving some context information about the document being classified. If the test document is truly *secret*, than we expect it to be similar

to existing *secret* documents with respect to non-trivial language (enforced by the *df* threshold). Table 10 shows that for *NE* examples from the Wikipedia Test corpus, the $xtra.info_{secret}$ is quite high and enables a second classifier to easily separate these documents from true *secret* documents.

To better differentiate between *public* and *secret* documents, we train a SA_{public} classifier. By combining *secret* and *NE* documents, the classifier will better recognize which features correlate with *public* documents. On its own, the output of the classifier will not necessarily exceed the performance of the SA_{secret} classifier. But when combined with the output of SA_{secret}, $xtra.info_{public}$ and $xtra.info_{secret}$, the meta-space classifier better discriminates between *public* and *secret* enterprise documents.

The usage of this meta-space classification is improved by using three classes instead two (i.e. *secret* or ¬*secret*). Combining *public* and *NE* is not optimal because we expect much different behavior for each of the attributes. *NE* documents will most likely have higher $xtra.info_{private}$ and $xtra.info_{public}$ scores than *public* documents and be classified ¬*public* by SA_{public}. This will negatively affect classification for these attributes because the separability of these values is diminished by grouping them together. Our SVM uses Hastie et al. [109] pairwise coupling algorithm for multiclass classification.

In summary, our meta-space classifier is trained four features: the outputs of SA_{public} and SA_{secret} classifiers, $xtra.info_{public}$ and $xtra.info_{secret}$. We train the classifier on the *NE*, *public*, and

secret documents that were misclassified by SA_{public}. *NE* and *public* documents are not combined together as in the $SA_{private}$ classifier, but rather, assigned to one of three classes (*NE*, *public* and *secret*) based on its prominence. To classify a new document, *d*, we first compute $SA_{secret}(d)$. If this classifier indicates that *d* is not *secret*, we mark *d* as *public*. Otherwise, we compute $SA_{public}(d)$ and $xtra.info_{public}$ and $xtra.info_{secret}$ for *d* and apply the meta-space classifier to obtain a final decision.

6.3 DLP corpora

We have created five corpora for training and evaluating DLP classification algorithms. Table 6 gives a brief description of each corpus. To our knowledge, these are the first publicly-available corpora for evaluating DLP systems. Constructing DLP corpora is challenging because they should contain private information from some enterprise, but private information is, by definition, difficult to obtain.

Three of our corpora – DynCorp, TM, and Mormon – contain private documents leaked from these organizations to Wikileaks and public documents taken from the organizations' public web sites. DynCorp is a military contractor that has drawn substantial controversy for its actions in Bosnia, Iraq, and Afghanistan. The private document set includes their field manual for operatives. The Transcendental Meditation movement is a quasi-religious organization that operates clinics and workshops and has been declared a cult in France and Israel. The private documents obtained from Wikileaks include workshop instructions written by high-ranking members of the organization. The Mormon corpus includes a Mormon handbook that is not to be

Dataset	Source of Sensitive Documents	Source of Public Documents	Comment
DynCorp	WikiLeaks	www.dyncorp.com	23 private documents leaked from the military contractor Dyncorp
TM	WikiLeaks	www.alltm.org www.tmscotland.org	102 documents from high ranking officials in the Transcendental Meditation movement/cult
Mormon	WikiLeaks	www.lds.org	Private Mormon handbook split into 1000 word chunks
Enron	Enron Email	Enron website via Internet Archive's WayBack Machine	399 emails labeled by Hearst et al. as business-related [110]
Google	Google Product Blogs	Google Public Relation Blogs	Label product-related posts as private and public relations posts as public
Wikipedia NE datasets	-	Wikipedia	10K randomly selected articles for false positive detection
Brown Corpus	-	A variety of sources including press releases, reviews and books	500 texts selected to represent modern American English
Reuters-21578	-	Reuters News Service	10788 news items published by the news service

Table 6: Corpora used in our evaluation.

distributed outside of its members. We split the handbook into 1000 character-long pieces and added other smaller supplemental organizational documents from the church available through WikiLeaks. Note that our inclusion of texts from religious organizations is not intended to

denigrate these faiths – we include these texts solely because they are documents that these organizations tried to keep secret.

The Enron corpus contains emails released during the Federal Energy Regulatory Commission investigation of Enron, Inc. The archive contains emails from 150 users, including management. A small subset of the emails have been labeled by Hearst et al. [110]. Our data set only includes "business-related" emails. Since Enron is now defunct, we used the Internet Archive [111] to obtain documents from its public website.

The Google private document dataset consists of posts by Google employees to software-development blogs. Google collaborates with many open-source software projects, so much of its software development discussions take place in public. If these same projects were conducted as closed source development, then these blog posts would be private, internal documents, so we treat them as such in our dataset. Public documents were taken from PR-related blogs.

Finally, we include several corpora that are intended to represent non-enterprise documents. In addition to sampling 10K randomly selected Wikipedia articles, we also test the robustness of our classifier on the Brown [112] and Reuters [108] corpora.

6.4 Evaluation

A successful DLP classifier must meet several evaluation criteria. It must have a low false negative rate (i.e. misclassifying *secret* documents) and low false positive rate for any non-*secret* document. It should also achieve a low false discovery rate. Furthermore, we need to show that our classifier is robust with respect to its training parameters, in particular: the choice of the

	DynCorp		TM		Enron		Mormon		Google	
Classifier	FP	FN	FP	FN	FP	FN	FP	FN	FP	FN
Baseline	0.0%	0.0%	2.5%	0.98%	0.87%	0.0%	0.72%	1.4%	1.8%	1.9%
Supplement	0.0%	8.0%	0.0%	11.0%	0.0%	5.0%	0.0%	0.3%	0.0%	3.7%
Supplement and Adjust	2.0%	0.0%	28.3%	0.0%	4.1%	1.2%	4.6%	0.0%	15.9%	0.3%
Two-step	0.0%	0.0%	0.0%	0.98%	0.87%	3.0%	0.36%	1.4%	1.0%	2.1%

Table 7: The false positive (FP) and false negative (FN) rates on the enterprise corpora for each of our classification strategies. 11,100 instances and an adjustment of 90% are used.

Non-enterprise False Positive Rate					
Classifier	DynCorp	Enron	Mormon	Google	TM
Baseline	4.7%	87.2%	0.16%	7.9%	25.1%
Supplement	0.0%	0.01%	0.06%	0.0%	0.0%
Supplement and Adjust	0.26%	2.5%	0.1%	2.8%	0.93%
Two-step	0.0%	0.05%	0.0%	0.06%	0.01%

Table 8: The false positive rates on our Wikipedia Test corpus for each of the classification strategies.

Dataset	Baseline FDR	Our classifier FDR
DynCorp	4.49%	**0.00%**
Enron	47.05%	**0.92%**
Google	8.99%	**1.06%**
Mormon	0.88%	**0.36%**
TM	22.06%	**0.01%**
Average	16.69%	**0.47%**

Table 9: The False Discovery Rate of the baseline approach far exceeds our classifier, implying that the baseline approach would fare poorly in real world networks whereas ours would not raise much fewer alarms.

supplemental corpus, the size of the supplemental corpus, and the degree of adjustment used in the supplement and adjust classifier.

We present the results of our training strategy against a baseline classifier. For all our classifiers, we tokenize all our datasets and use unigrams for features. For a baseline classifier, we only train the classifier on enterprise documents using the binary weighting scheme. For the results presented in Table 7 and Table 8, we supplement the classifiers with 10000 Wikipedia

articles and 1100 topical articles and adjust the classifier to move the decision boundary 90% of the distance between the decision boundary and the closest correctly labeled *public* instance. We use a document frequency of 0.5% to compute $xtra.info_{secret}$ and $xtra.info_{public}$. We compute the false negative and false positive rates by performing a 10-fold cross validation on each of the corpora, and then determine the false positive rate for *NE* documents by training the classifier on the entire enterprise dataset and then classifying our Wikipedia false positive corpus.

The results of our classification tests show that our training strategy maintains low false negative and false positive rates on enterprise documents while dramatically improving the false positive rate on *NE* documents. The baseline approach would be unusable in practice because of its high false positive rate on *NE* documents.

In our results shown in Table 9, we assume the following traffic composition in a typical enterprise network: 25% enterprise secret documents, 25% enterprise public documents, and 50% non-enterprise documents. We believe that our approach will not engender "alarm fatigue", whereas the baseline approach is likely to overwhelm operators with false alarms.

The supplement and adjust classifier achieves a low false positive rate on *NE* documents for several reasons. The supplement and adjustment classifier did not rely on finding features that were strongly indicative of the *public* class. This is a crucial benefit because the *NE* document set's size is so large that it would be impossible to create a set of features that were *strongly indicative* of all possible *public* documents. In addition to relying less on features that were indicative of *public* documents, the supplement and adjustment classifier moves the expectation further towards the *public* class, which is in line with our expectation of the problem

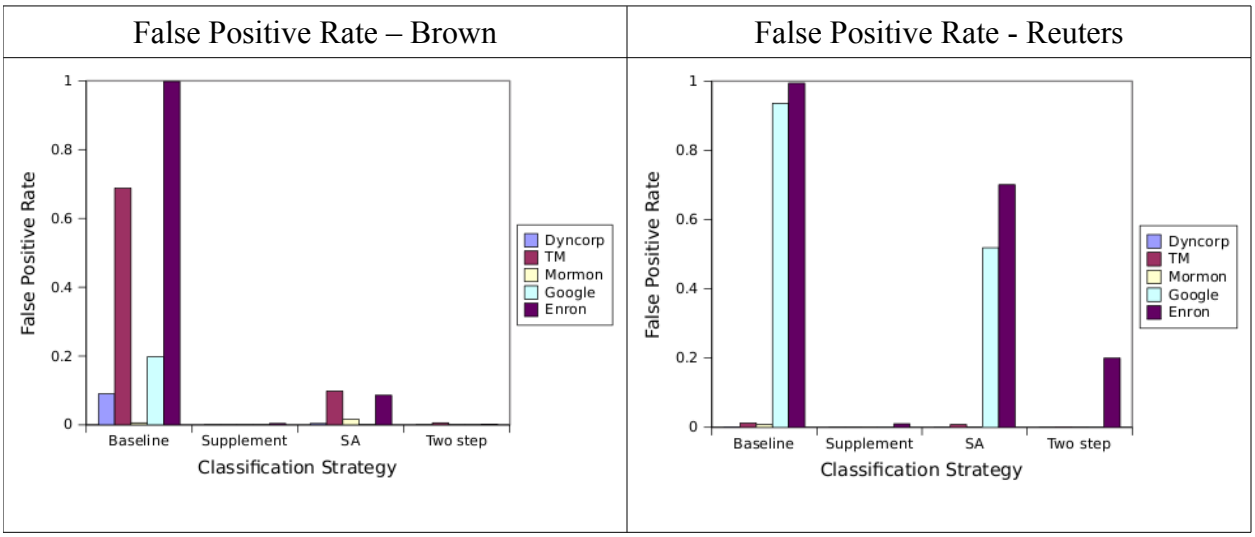


Figure 11: The false positive rates for each classification strategy. The two-step classifier is able to maintain a low false positive rate across all the different corpora for each non-enterprise corpora.

outlined in the problem description. And by performing an adjustment to the decision boundary, the classifier reduces the false negative rate without increasing the false positive rate, when combined with the second level classifier.

6.4.1 Effective training parameters

Table 7 demonstrates that our classifier is robust with respect to the choice of the supplemental corpus. Our supplemental corpus consisted solely of Wikipedia documents but, as Figure 11 shows, the resulting two-step classifier has a low false positive rate on *NE* documents drawn from drastically different corpora, such as the Brown or Reuters news corpora. Thus, we can build a standard non-enterprise corpus that is used by all enterprises to train their DLP systems. The corpus will not need to be customized for each enterprise or for each new form of Internet traffic.

Figure 12: The effect on the false negative and false positive rates for our corpora when supplementing the training instances with Wikipedia examples. For the Mormon corpus, the effect of adding any supplemental instances seems to affect the classification of the same documents.

Figure 13: The false positive and false negative rates on enterprise documents after applying the supplement and adjust classifier.

As expected, a larger supplemental corpus decreases the false positive rate but increases the false negative rate as the classifier becomes more biased towards *public* documents (see Figures 12 and 13 for details). Note that Google is a clear outlier in this evaluation. We suspect that this may be because the Google corpus is the only artificial corpus in our data set. Recall that all the Google documents, including the "private" ones, are in reality *public* documents, unlike our other corpora which contain genuine private enterprise documents. The second step of

our approach remedies the errors made on *public* enterprise documents. We also conclude that the supplemental corpus does not need to be too large – about 10,000 documents suffice.

We also investigated the effect of the adjustment value on the classifier. According to the graphs in Figure 11 an adjustment value of 0.5 provides a good trade-off between increased false positives and false negatives in the supplement and adjust classifier. However, since we added a second-level classifier that can filter out many false positives, we chose an adjustment value of 0.9 in order to achieve a slightly lower false negative rate.

6.5 Discussion

The algorithm presented in this chapter should prevent accidental leakages of information, but how will it fare against intentional leakages? According to ProofPoint [113], most data leakages are accidental. The most common intentional leakage occurs when employees download sensitive information upon termination of employment. Our method coupled with the DLP system's ability to recognize data flow from a trusted to an untrusted device should prevent these type of leakages. If the data were encrypted or re-encoded, this would exceed the capability of our classifier. These more sophisticated attacks, fortunately, only account for 1 in 200 data breaches [105].

It is instructive to highlight key differences between our solution and existing semi-supervised and class imbalance solutions. Our algorithm is a supervised learning approach: all examples are labeled. During training, the classifier will know if the enterprise document is confidential or not. Since supplemental training instances do not come from the enterprise, these instances are labeled opposite from the class we wish to train on, e.g., for the $SA_{private}$ classifier,

these supplemental instances are labeled as *public*. For the purposes of our algorithm, we focus on recognizing sensitive information that either it has either seen before or is similar to an existing confidential document. In the future, we hope to explore how the system can infer if a document is sensitive if it has zero training data to support this decision (possibly relying on meta data).

Our study demonstrates that DLP systems face an inherent class imbalance issue: nearly all documents that exist are outside the organization and are not sensitive. To train a classifier on this class is simply infeasible because of its size. Our key insight into this problem is recognizing that the classifiers needed to be trained to effectively learn what is *secret*, and not rely too heavily upon features that were correlated with non-*secret* documents. The problem of class imbalance has been extensively studied before and work in this area is discussed in Section 6.6 Once we recognized that class imbalance would be an issue for achieving maximal performance, we tried many of the approaches listed in the Section 6.6 , but found that they were ineffectual on this specific problem.

Our approach is unique from other class imbalance techniques because we attempt to better determine which features correlate with sensitive information by adding additional samples that express a diverse usage of language and better capture the general usage of the n-gram. We cannot say how well this technique will extrapolate to other machine learning problems, but it is applicable to our specific problem of generating a classifier robust enough to perform well in the presence of many unrelated documents. To the best of our knowledge, using

Mean	Dyncorp	Enron	Google	Mormon	TM
Secret documents	0.54 (0.10)	0.83 (0.09)	0.70 (0.15)	0.49 (0.15)	0.66 (0.11)
NE documents	0.96 (0.03)	0.99 (0.02)	0.98 (0.04)	0.95 (0.08)	0.99 (0.02)

Table 10: This table presents the means for the $xtra.info_{secret}$ attribute for each of our private corpora and the document classes secret and NE. The significant differences between the means for these classes suggest that this attribute will aide the classifier in distinguishing NE documents from secret.

supplemental data (not synthetically generated) to generate negative examples has not been applied to the class imbalance for text classification.

An important design decision in this algorithm was to restrict the vector space to features included only in *secret* and *public* documents. The reasoning behind this decision is related to the class imbalance aspect of this problem. Since the number of non-*secret* documents is so large, adding additional features to the vector space would have resulted in overfitting because those features would factor prominently into classifying *NE* documents in the training step. The classifier may not accurately reweight features that *secret* documents share with non-*secret* documents. And since it would be impossible to provide the classifier with training representative of everything that is *NE*, the classifier would be more likely to generate false positives.

The *xtra.info* attribute performs exceedingly well in maximizing separability between *NE* and *secret* documents, as shown in Table 10. Contextual information is quite important because we have limited our vector space to only enterprise documents, which these terms are assumed to be related to the knowledge domain of the enterprise. Using a unigram vector space, we lose contextual information that may help counteract the effect of polysemy that contributes to the misclassification of *NE* documents. Our *xtra.info* attribute is effective in the second level of

classification in providing contextual information to disambiguate between *secret* and *NE* classes and is easily computable.

The techniques of our algorithm performed well for many different different types of media and organizations. One limitation in creating our DLP corpora is that it the documents for each organization do not represent the entirety of its operations. It was not feasible to either find or build a corpus of this nature because of the risk for corporations assembling and releasing this data. We believe, however, that since our algorithm performed well in many different instances, it will perform well enterprise wide. Depending on the size and structure of the organization, multiple classifiers can be built for each of the different departments and group. Text clustering can also assist in building cogent collections of documents to train and build classifiers. And since the classification techniques we use are not computationally expensive, the penalty for evaluating multiple classifiers is not prohibitively greater.

The system described in this chapter will most likely be part of a larger enforcement framework that will defend the network from intrusions and malware. Administrators will need to provide instances of *secret* and *public* documents because the training of our system is supervised. This collection of samples, however, should not be difficult to obtain because it can be automated and does not require further annotations. Employees can designate folders on storage devices that contain either *secret* or *public* documents or manually submit examples through a collection facility (e.g. email or web-based system). Depending on the enterprise policy enforcement guidelines, messages that the classifier suspects to be *secret* may prompt the

sender to reconsider, queue the message for an administrator to review, or simply block the transaction. The toolkit we implemented will be made available from the authors' website.

6.6 Related work

Automated document classification is a well studied research area. Research in the document classification field dates back to 1960s [114][115]. The use of machine learning in text classification, however, became popular in the last two decades. Sebastiani provides an excellent overview of the area: he describes various text categorization algorithms, approaches to evaluate the algorithms, and various application of automated text categorization [116]. The proliferation of digital documents and the explosion of the web has given rise to many applications of document classification, e.g., automatic document indexing for information retrieval systems [117], classification of news stories [118], email filtering to classify emails into categories [119], spam filtering to identify spam from legitimate email messages [120], automatic categorization of web pages [121][122], and product review classification [123]. The research community has explored many different machine learning approaches for text categorization, e.g., Bayesian classifiers [120][124], decision trees [125], k-nearest neighbors [126], neural networks [127], regression models [128], and support vector machines [129]. Researchers have also experimented with the idea of combining multiple classifiers to increase efficacy, most notable being Schapire et al.'s *boosting* approach [130].

We utilize Support Vector Machines, a powerful margin-based classification and regression technique introduced by Cortes and Vapnik, in our classifier [131]. Joachims applied SVMs to the text classification task [104] and identified properties of text categorization that

makes SVMs suitable for the task. For example, text categorization has to deal with large numbers of features, as words present in a document are considered the document's features. Feature selection is a traditional approach to select a few *relevant* features from many. In the case of text, however, most features are relevant. Hence a good text classifier should be able to handle many features. SVMs can handle large numbers of features due to overfitting protection. Also, SVMs are good *linear* classifiers and many text categorization tasks are linearly separable.

Text classification for DLP presents difficulties that standard classifiers cannot solve because of the lack of a proper training set. It is difficult to supply the classifier with an adequate representation of what should be *public* (i.e., not *secret*). Therefore, this chapter addresses the precise problem of an unrepresentative dataset for text classification with the techniques of supplement and adjust, *xtra.info*, and utilizing a two-step classifier. Other research has focused on related topics.

Accurate text classification in the case of limited training examples is a challenging task. Joachims used a transductive approach to handle the problem [132]; his approach focuses on improving the classification accuracy of SVMs for a given test set. Blum and Mitchell introduced a co-training approach to categorize web pages [133]. They improved classification accuracy by adding a larger number of unlabeled examples to a smaller set of labeled examples. Toutanova et al. demonstrated the use of hierarchical mixture models in the presence of many text categories [134].

Researchers have also investigated mitigating the effect of class imbalance on classification performance [135]. Both oversampling and undersampling classes in the training

instances has been widely investigated. The sampling can be random or directed. Synthetic generation of examples for underrepresented classes has also been explored and combined with under and over-sampling [136]. One class learning classifiers have been proposed to improve classification for target classes where examples are relatively scarce compared to other classes [137]. An instance is compared with training examples in terms of similarity to determine whether the instance is a member of the target class. Lastly, feature selection techniques can improve classification of underrepresented classes because high dimensional data may overfit or be biased towards the majority classes [138].

Several projects have used Wikipedia to enhance text classification, particularly where context is unavailable due to the brevity of the text to be classified [139][140][141][142]. Gabrilovich et al. [139] first proposed transforming a document into its representation in Wikipedia topic space. Others have modified this basic idea by including topic hyponomy and synonymy [141] or performing LSA on this topic space [142]. Others have investigated using Wikipedia to determine relatedness between texts, particularly short texts [143]. To our knowledge, no one has investigated using Wikipedia explicitly to augment their training corpus.

6.7 Conclusion and future work

This chapter presents a simple, efficient, and effective way to train classifiers and perform classification for Data Loss Prevention. In doing so, it presents the first corpora for the DLP task. Our results indicate a naive approach to training a classifier, solely on documents from the enterprise, will lead to a high false positive rate on unrelated documents, indicating poor real

world performance. The chapter presents a novel technique, supplement and adjust, which reduced the false positive rate for documents unrelated to the core business function.

We plan to further study the efficacy of our text classification approach by deploying it on existing private, enterprise and governmental networks. We will also look to expand our approach to include encrypted and multimedia content. In this work, we only consider the content of a document to render a decision. We would like to investigate what meta data associated with the content could be used to improve classification.

Lastly, not all secret documents in the world are written in English. We will hope to expand our private corpus in the future to include non-English sources. Our intuition is that many language processing techniques developed to handle language specific obstacles should be applied to the processing of these documents. We will also have to adjust our supplemental corpus accordingly to provide realistic behavior for *NE* feature behavior.

Chapter 7 Wikipedia Vandalism

Wikipedia, the "free encyclopedia" [59], ranks among the top 200 most visited websites worldwide [144]. This editable encyclopedia has amassed over 15 million articles across hundreds of languages. The English language encyclopedia alone has over 3.5 million articles and receives over 1.25 million edits (and sometimes upwards of 3 million) daily [145]. But allowing anonymous edits is a double-edged sword: nearly 7% [60] of edits are vandalism, i.e. revisions to articles that undermine the quality and veracity of the content. As Wikipedia continues to grow, it will become increasingly infeasible for Wikipedia users and administrators to manually police articles. This pressing issue has spawned recent research activities to understand and counteract vandalism [146]. Much of previous work relies on hand-picked rules such as lexical cues (e.g., vulgar words) and metadata (e.g., anonymity, edit frequency) to automatically detect vandalism in Wikipedia (e.g., [147][148]). Although some recent work has started exploring the use of natural language processing, most work to date is based on shallow lexico-syntactic patterns (e.g., [149][150][151]).

This chapter presents a machine-learning-based vandalism detector that uses several features to classify vandalism and achieves 79% precision and 60% recall on vandalism and an AUC score of over 94%. Our results significantly outperform a baseline classifier based on a previous approach [152]. Our classifier uses several simple features as well as Natural Language

Processing (NLP) motivated features to catch different forms of vandalism, such as inserting obscenities or long, repeating patterns of text.

We explore more linguistically motivated approaches to detect vandalism in this chapter. There are many avenues that seem quite germane to the vandalism detection problem: sentiment analysis, word sense disambiguation and stylometric analysis. Our hypothesis is that textual vandalism constitutes a unique *genre* where a group of people share similar linguistic behavior that can be measured through stylometric analysis. Some obvious hallmarks of this style include usage of obscenities, misspellings, and slang usage, but we aim to automatically uncover stylistic cues to effectively discriminate between vandalizing and normal text. Experimental results suggest that (1) statistical models give evidence to unique language styles in vandalism, and that (2) deep syntactic patterns based on probabilistic context free grammar (PCFG) discriminate vandalism more effectively than shallow lexico-syntactic patterns based on n-grams and (3) we can employ co-training corpora from the Internet to improve our NLP approaches and (4) character level language models are incredibly effective in distinguishing between language appropriate and inappropriate for Wikipedia.

The rest of this chapter is organized as follows. We introduce our training corpus and features in Section 7.1 . A detailed description of how we processed the text will be discussed in Section 7.2 . Section 7.3 discusses the classifier that will label revisions as either regular or vandalism. We detail our evaluation in Section 7.4 . Results are presented in Section 7.5 with a discussion following in Section 7.6 . The chapter will conclude with a survey of related work in Section 7.7 and a conclusion in Section 7.8 .

7.1 Training corpus and feature extraction

The training corpus for the classification was provided by the organizers of the PAN workshop [153][154]. The corpus consisted of 32444 edits coupled with the previous revisions. Along with the training corpus in WikiMarkup format, we were also provided with meta-data including the edit id, the old revision id, the new revision id, the user name or IP of the author who performed the edit, the comment of the author, and whether the edit vandalized the article.

7.1.1 Shallow features

After analyzing samples of vandalized and regular edits and reviewing early work on automated vandalism detection [147], we observed that certain shallow features of the edits distinguished vandalism from regular edits. Before investigating more sophisticated features, we attempted to identify "shallow" or superficial attributes that could be supplied to the classifier for identification. This includes observing changes in the text, who submitted the edit, and if the edit contained vulgar language. We used the following shallow features of the edits for classification:

Edit Distance: Our classifier calculates the Damerau-Levenstein Distance using the LingPipe API [155] to determine the number of changes required to convert the old revision of an article to its new revision.

Edit Type: Our classifier determines whether the edit inserted, deleted and/or modified text or a combination of these actions.

Text Changes: Our classifier determines the edit length, word count, words inserted, words deleted, and words changed using java-diff [156]. It also uses LingPipe's English

Sentence chunker to tokenize an article into sentences and calculate exact changes sentence-by-sentence using java-diff.

Spelling Errors: Our classifier counts the number of apparent spelling mistakes in the edit and the ratio of spelling errors to correctly spelled words. Our spell-checking [157] software contained 200K English words, including named entities such as proper names and geographic places.

Obscene Words: Our classifier enumerates the total number of obscene words in the edit and the ratio of obscene words to benign words. We started with a dictionary of obscene words [158] and manually added other obscene words that we observed frequently in the vandalized articles of our training set.

Repeated Patterns: "Silly" vandalism often employs repeated patterns like upper case words, exclamation marks, and repetition of words/letters (e.g. "heyyyyyy", "oh! ! ! ! ! ", "wow wow wow", "hahahaha", "WIKIPEDIAAA"). Our classifier counts these patterns using the regular expressions.

Grammatical errors: Wikipedia editors strive for good grammar, but vandals do not generally follow these rules. They may insert words or phrases into the middle of existing sentences or write ungrammatical sentences deliberately or unintentionally. Our classifier parses the edits into sentences that had been inserted or changed by using java-diff and LingPipe's sentence tokenizer and counts the grammatical errors in them using CMU's Link Grammar Parser [159].

Sum of metrics: This simple but effective meta-feature is the sum of the number of Repeated Letters, Repeated Words, Capitalized Words and Multiple Exclamation Marks.

Article History: Vandals do not necessarily target articles at random to vandalize. Rather, some articles receive a disproportionate amount of vandalism than others (e.g. Michael Jackson). This feature is the number of times an article was vandalized in the previous 5000 edits on the article. We denote vandalism as a comment left by an editor that contains "reverted", "user" and "vandalism" in this order. We count also how many times an article was reverted, regardless if it was explicitly vandalism or not.

Editor information: Knowing who contributed an edit can give us some expectation of the quality of the change. For example, we expect an active and registered Wikipedia editor with several thousand edits to be more trustworthy compared to an unregistered editor that is only identifiable from an IP address and who has not contributed before. Our classifier uses several editor-based features: whether the editor is registered or unregistered, how long the editor has been registered, the total number of contributions made to Wikipedia during the period that training data was collected, the total number of edits made to Wikipedia by the editor up to the date the training data was finalized, the number of reverts on previous revisions by the author deemed to be vandalism (using the same heuristic as for article history) and the total number of previous modifications made by the editor on the article they are revising. We were careful to make sure for edit totals to not include future modifications. If the user is not registered, the classifier uses their IP address as a proxy for their identity.

7.1.2 Employing Natural Language Processing

During the PAN 2010 workshop [153], it became evident that many of the participants converged on the same sets of meta and shallow features. The issue, however, is how effective these features ultimately can be for more subtle types of vandalism. For example, a vandal does not need to employ obscenities or insert first or second pronouns to add opinionated text into an article. Also, new ways of conveying opinions in terms of memes and signs are being constantly generated by Internet users such that there exist websites to keep track of them [160]. Therefore, there is a need to go beyond features driven by heuristics to truly understand the content and effect of an edit.

Natural Language Processing is the study of the intersection of computer interaction and natural language. This field strives to derive meaning from streams of natural language by statistical and rule-based modeling. This includes tasks such as sentence parsing to sentiment extraction. Many of the problems addressed by NLP may be brought to bear on the vandalism problem. For instance, Wikipedia mandates that articles maintain objectivity and factuality. Vandals do insert irrelevant and opinionated text into articles.

The following subsections will explore how NLP can be applied to the vandalism problem. The next section will describe four corpora that we collected for co-training with our articles. We identified three potential areas of NLP that can be applied to this problem. First, we will use sentiment analysis to identify if a vandal has inserted opinionated or subjective text. Second, we will perform semantic modeling using language modeling to determine if the language used more closely resembles normal or vandalizing edits. Lastly, we will employ

stylometric analysis to analyze the editor's style to glean if the intention was to enhance or vandalize an edit.

7.1.3 NLP co-training

One issue that we have is that despite the monumental efforts of Potthast to collect samples of vandalizing edits to Wikipedia, there are still less than 2500 samples in the training set to work from. We set out to discover sources for examples of text that are highly likely to be unsuitable for Wikipedia, i.e., if we were to insert the text into an article – even if topically related – would likely be rejected for being vandalism or not stylistically appropriate. To find examples of this text, we look for websites that exemplify the type of language deemed inappropriate for Wikipedia, but representative of the web in general. We choose the following four websites to sample text from: 4chan [161], YouTube [162], MySpace [163] and Twitter [164]. 4chan is an image-board website that has been active since 2003. A compelling reason to sample text from this site is the comments posted to the site have the qualities we typically ascribe to "silly vandalism" (the Guardian described the comments as "lunatic" and "juvenile" [165]). Another source of candidate text comes from YouTube comments. A good majority of these comments are crude, capture Internet style and syntax, and/or not written well, e.g., misspellings. To sample text, we scraped comments from the most popular videos, which tend to solicit a substantial amount of feedback. MySpace, the popular social network, allows users to post messages to other user's profiles. We found that if we retrieved the "browse people" interface from MySpace's website, we could retrieve a random sample of users. From here, we scraped comments from the user's profile and use them for training. Lastly, we used tweets from

Twitter to generate training data. Twitter has become the world's most popular micro-blogging platform. This reach allows us to potentially sample from a large number of individuals with diverse backgrounds, education levels and language (and indeed we sampled text in languages other than English including Spanish). Therefore, we hypothesize that Twitter represents Internet speech well considering its pervasiveness and popularity. We sampled tweets by finding messages posted to Trending Topics [166] (those topics identified by tags that are currently experiencing a surge in traffic).

To generate these co-training corpora, we sampled these sites every 15 minutes over the course of three days. We extracted 278463, 55234, 768361, and 612916 samples from 4chan, MySpace, YouTube and Twitter respectively for over 1.7 million text samples with on average 92 characters and roughly 16.4 words per sample.

7.1.4 Sentiment analysis

Statements expressing opinions are common features of vandalism. We perform sentiment analysis on the revision to uncover subjective or opinionated (positive or negative) text. We use LingPipe's Sentiment Analysis Tool trained on movie review data. The classifier counts objective or opinionated sentences and measures the change in total number of positive or subjective sentences.

7.1.5 Semantic modeling

One goal in NLP is to learn the meaning of text. Although perhaps machine understanding will not parallel human intelligence for quite some time, there are still some ways in which shallow machine processing techniques can glean semantic information from text to

differentiate contexts. In fact, [149] investigated shallow language modeling of syntax and semantics in order to determine relevance of edits to new text. The authors of the "Got You!" system built two language models on a per-article basis: N-tag and Syntactic N-Gram. The N-tag language model is a trigram language model trained on a composite part-of-speech tag and word feature, rather than the word alone. Combining the part-of-speech provides some contextual information for semantic disambiguation. The training for this model is generated by performing a combination Yahoo! and Bing search on the article title and collecting the top 100 results from Yahoo! and Bing. The second language model, Syntactic N-Gram, is trained on only part-of-speech tags. The intuition is Wikipedia articles will not share similar syntactic behaviors of vandals, such as using multiple punctuation marks.

For the purposes of this dissertation, we do not compare our results with this work for two reasons. First, the semantic model is computationally expensive and requires much storage because for each article it requires generating the language model. This computation could be optimized for real world deployment by perhaps limiting the number of searches or building language models that are applicable to multiple articles. Also, the authors present results on a balanced corpus, which is not representative of real world vandalism rates. Therefore, we will evaluate our techniques on an unbalanced dataset similar to what we expect in the real world.

We do, however, train two trigram language model (LM) with Good-Turing discounting and Katz backoff for smoothing of vandalizing edits (based on the text difference between the vandalizing and previous revision) and good edits (based on the text difference between the new

and previous revision). Admittedly, these language models will not have the ability to anticipate the inclusion of semantically relevant words.

7.1.6 Stylometric features

Stylometric features attempt to recognize patterns of style in text. These techniques have been traditionally applied to authorship attribution [167][168], opinion mining [169], and forensic linguistics [170]. For our purposes, we hypothesize that different stylistic features differentiate the voices of regular and vandalizing edits. For regular edits, honest editors will strive to follow the stylistic guidelines set forth by Wikipedia (e.g. objectivity, neutrality and factuality). For edits that vandalize articles, these users may converge on common ways of vandalizing articles.

This subsection will cover several different stylometric analyses to identify vandalism. We analyze sentence structure, stop word analysis, spelling mistakes, and readability.

Probabilistic Context-free Grammars

In identifying differences in style, one feature of style to examine is sentence structure. Sentence structure has been a source of author identification in literary analysis [168]. If we consider that vandalism constitutes its own *genre* of writing, we expect to see differences in the construction of sentences. One anecdotal example from an instance of vandalism in our test set is the following: “One day rodrigo was in the school and he saw a girl and she love her now and they are happy together.” This sentence demonstrates an excessive use of the conjunction “and”. We would expect that well written articles in Wikipedia to only employ conjunctions when it is expedient to do so and to separate related ideas using semi-colons or start new sentences. This

example, however, uses “and” successive times to string multiple ideas together, implying that the writer does not conform to the style and quality of writing required by Wikipedia. Therefore, we can employ syntax parsing to build parse trees to analyze the stylistic differences between vandalism and non-vandalizing sentences.

To assist in discovering of different styles in the writing of sentences, we employ probabilistic context-free grammars (PCFG). [171] reported for the first time that PCFG models are effective in learning stylometric signature of authorship at deep syntactic levels. We explore the use of PCFG models for vandalism detection, by viewing the task as a genre detection problem, where a group of authors share similar linguistic behavior. We give a concise description of the use of PCFG models below, referring the reader to [171] for more details.

1. Given a training corpus *D* for vandalism detection and a generic PCFG parser trained on a manually tree-banked corpus such as WSJ or Brown, tree-bank each training document using the generic PCFG parser .
2. Learn vandalism language by training a new PCFG parser $P_{vandalism}$ using only those tree-banked documents in *D* that correspond to vandalism. Likewise, learn regular Wikipedia language by training a new PCFG parser $P_{regular}$ using only those tree-banked documents in *D* that correspond to regular Wikipedia edits.
3. For each test document, compare the probability of the edit determined by $P_{vandalism}$ and $P_{regular}$, where the parser with the higher score determines the class of the edit.

We employ many different strategies for collecting the documents D to create treebanks and train parsers. As a first cut, we take train two parsers on regular edits ($D_{regular}$) and vandalizing edits ($D_{vandalism}$.). We also construct separate tree-banked documents for each of our co-training copora ($D_{twitter}$, $D_{myspace}$, D_{4chan}, $D_{YouTube}$) . We hypothesize that having these additional tree-banked documents could potentially anticipate styles of vandalism that stylistically transfer from different domains that simply have not been observed in $D_{\text{vandalism}}$.

In addition to constructing tree banks from the above sources, we also investigated if it were possible to improve modeling of non-vandalizing edits by training separate parsers on sentences from different categories. This idea is motivated by the observation that articles from different knowledge domains will use different sentence constructions to convey ideas. For example, mathematical articles will contain sentence structures to describe theorems that may not be well represented in biographical articles. Therefore, for each of category M that is a main category in Wikipedia [172], we first find the set of categories M_c that are children of M in the Wikipedia taxonomy. From here, we then extract all articles from the Wikipedia November 2008 article dump [173] that contain at least one tag from $M \cup M_c$. We then sampled 10000 sentences from each of these articles by randomly selecting an article and then selecting a sentence from the article. In addition, we combined all the sentences from each M into a "master" treebank. This experiment in total generated treebanks for the following main categories: Arts, Belief, Computing, Culture, Education, Environment, Geography, Health, History, Humanities, Language, Law, Master, Mathematics, Nature, People, Politics, Science, Society, Technology.

We also attempted to improve the quality of the $D_{vandalism}$ by clustering the trees into separate treebanks. The motivation is to cluster sentence parses that have similar construction internally to build a more precise parser for a particular type of sentence structure. To do so, for each parse tree in $D_{vandalism}$, we generate the binarized tree representation by first converting it to Chomsky's Normal Form. From here, for each internal node P in the parse tree, we extract each child node C and represent the tree as a bag consisting of each $P \rightarrow C$. We do not consider terminals for this evaluation, therefore there does not exist any C that is a terminal. We perform clustering by mapping the bag into a vector space where each dimension is corresponds to a rule derivation $P \rightarrow C$. We used K-Means clustering [174] for the following set of K values: $\{2,4,8,16\}$. We found that the majority of sentences mapped into 5 distinct clusters (from analysis of the clusters generated by K-Means for K equal to 8 or 16). We then mapped the documents from $D_{\{Twitter,MySpace,YouTube,4chan\}}$ into these 5 clusters by mapping each sentence into the vector space generated by $P \rightarrow C$ using the procedure described above and assigning the sentence to the cluster whose centroid was closest to the sentence's representation in the feature space. We then generated 5 treebanks $D_{\{cluster10,\ cluster13,\ cluster4,\ cluster2,\ cluster6\}}$ and built corresponding parsers from these clusters on the original, non-binarized sentence parse.

We generated the following features. For each revision R and parser P, we collect the log probability of the sentence parse for each modified sentence (defined below in Section 7.2) of R. We calculate the minimum (min), maximum (max), standard deviation or 0 if only one modified sentence (stdev), mean and sum of the log probabilities for the modified sentences of R and create a feature for each parser and measure. We then compute difference for each of the

measures (min,max,stdev,mean,sum) between $P_{vandalism}$ and $P_{regular}$. In addition, we calculate ***composite*** features that take the difference between the best scores, i.e. best probability, for each of the measures across a set of parsers. Our "best.regular" feature takes the best score for each of measure across the regular, master and categorically trained parsers. Our "best.vandalism" parser takes the best score for each of the measures across the vandalism, YouTube and Twitter parsers. Our "cluster.vandalism" takes the best parse from the set of parsers generated by clustering vandalism and Internet co-training sentences. We then computed the difference for each measure for the following pairs: (regular, vandalism), (regular, best.vandalism), (best.regular,vandalism), (best.regular, best.vandalism), (regular, cluster.vandalism), (best.regular, cluster.vandalism). We found that the pair (regular, best.vandalism) had the most information gain (InfoGain) among the different measures.

Stop word Analysis

Stop words, such as "is", "and", and "I", are words that belong to a set of candidate words that are generally filtered out before or after processing text. For many tasks, these words do not generally add substantial value to the analysis, despite the fact these words constitute over half the words in an English language [175]. There has been growing research that in fact, these common words can be successfully applied to various NLP tasks. Recent results indicate the utility of stop word analysis for authorship attribution [176] and sentiment classification [177].

Therefore, our hypothesis is that vandals employ different usages of stop words. Our first test is to use language models for modeling stop word utilization. The intuition is that certain types of vandalism exhibit peculiar patterns of stop words, including the over usage of

conjunctions like “and” or seemingly irrelevant words to Wikipedia like “hi”. We trained a trigram language model using the LingPipe toolkit on 5000 articles selected randomly from Wikipedia that was categorized with one of the main categories of Wikipedia [172] with all non-stop words removed. For each edits' inserted text, we remove all non-stop words and calculate the log probability of the edit according to this language model.

We perform a second analysis of stop words in the inserted text by treating each stop word as a feature and training a classifier to discriminate between vandalizing and non-vandalizing usages of stop words. For this task, we only consider the inserted text rather than the modified sentences. We first construct a feature vector space where the dimensions are stop words. For each stop word *s* in the inserted text, and *n* the total number of stop words in the inserted text, we weight the corresponding dimension as $\frac{count(s)}{n}$.

An interesting question that arises is which classifier should we employ for analysis? We experimented with three classifiers. First, recent results from a student project in CSE628 (Stony Brook University) this past fall suggest that SVMs are well suited for this task. In addition, we experimented with Naive Bayesian classifier and Bayesian Networks. One issue that we observed from an initial examination of training data generation was that a substantial number of instances (7544) were empty.

We attempt to remedy the numerous empty instances with two approaches. We generate a non-empty version of the dataset by simply eliminating empty instances. The classifier will generally handle this case by appealing to the class that constitutes the majority. Second, we apply a meta-classification on top of each of our classification approaches. With a cost-sensitive

approach [178], we can bias our classifier to be more intolerant of false positives and false negatives. In our preliminary analysis, the false negative rate for SVMs exceeded 90%. Conversely, the false positive rate was extraordinarily high for the Naive Bayesian classifier. Therefore, in addition to the raw classifier scores, we apply a cost-sensitive approach to each of the classifiers to decrease specific error rates (although potentially at the expense of other error rates).

Readability

Readability is crucial for maintaining the integrity of Wikipedia articles. We expect that articles that have low readability are not appropriate for Wikipedia because it will not achieve its goal "... to empower and engage people around the world to collect and develop educational content under a free license or in the public domain, and to disseminate it effectively and globally" [179]. A recent study of articles for the top 50 most proscribed drugs by volume found that the average Flesch Kincaid Grade Level [180] is 15.4, which correlates to requiring a high level of readability.

For our analysis, we used the Fathom library [181] to compute the Flesch Kincaid score of modified sentences. We did not consider only inserted text because the editor possibly did not insert enough text to qualify as a full sentence.

Common misspellings

Another hallmark of poor style (which we assume Wikipedia aims to avoid) is poor editing. We found that the grammar analysis is difficult because of Wikipedia article features

such as lists and templates where it is not uncommon for ungrammatical text to be inserted, but be perfectly appropriate. Therefore, honest editors who desire to meet Wikipedia's guidelines will most likely edit their revisions. We will assume that honest editors will also take some pains to ensure that there are no spelling mistakes. We use lists of common misspellings compiled from Wikipedia [182] and Oxford University Press [183].

In addition to actual spell checking, we identify words that are commonly misspelled. We hypothesize the existence of these misspellings indicate that the user is not as concerned with the integrity of the article and therefore the revision is more likely to be vandalism. We generate a feature that counts misspelled words where one of the top three suggestions from our spell checker contains a commonly misspelled word.

7.1.7 Character level Analysis

One common observation about vandalism is the vandal's disregard for properly written English. We address many aspects of these “silly vandalism” edits with features such as longest repeated pattern. But writing these features requires time and effort to identify interesting patterns and then write heuristics to recognize them which provide minimal false positives. In addition, edits that have been marked as vandalism (in our case, less than 2500) cannot simply capture the diversity in the expression of vandalism.

To address these two concerns, we hypothesize that character level language models (CLMs) can be trained from existing sources without intervention to learn many aspects of silly vandalism. CLMs build language models on characters and not words. One immediate benefit is space reduction, a trigram model has to account for less states than if we consider all potential

Text	YouTube CLM	Wiki CLM
p.s. i love you	**-2.99**	-5.23
and tony is awesome!!! remember that... all of you... teehee	**-3.09**	-4.45
Farnham Pottery, Wrecclesham, Surrey with the preserved bottle kiln on the right of photo	-3.89	**-3.65**

Table 11: Comparison of log probability on CLMs trained on YouTube comments and one on inserted text from Wikipedia revisions. The first two examples are vandalism and the last is not.

words. Therefore, it is more likely that a character level language model will not suffer from the same degree of sparseness as a normal word language model.

Our intuition is that CLMs can account for many different types of behavior that many of our existing features are designed to individually detect. In the work of [149], the authors use language models to capture the behavior of part of speech tags. This will presumably identify salient text like "!!!" or ":))" as vandalism. With CLMs, it is very likely that these examples will exist on any of our four co-training corpora and be identified as vandalism. We also hypothesize that garbage insertions (nonsense textual insertions) will also be identified as vandalism because the log probability generated by a character-level language model for regular edits will be quite low compared to the probability generated by one of our Internet corpora since the insertion does not conform to normal written English (the same may be true for insertions that contain non-English languages). Table 11 provides examples of the output for two CLMs trained on YouTube comments and Wikipedia articles respectively.

We trained several different CLMs. First, we trained a "regular" CLM based on 900K sentences from articles that contained one of the main categories of Wikipedia. In addition, we trained two other CLMs on non-vandalizing revisions with and without markup. We then trained

individual CLMs for each of our Internet co-training corpora (e.g. YouTube). We also trained one CLM sampling half the instances of the co-training at random. We also trained a special instance of the CLM based on the co-training samples called "internet.interesting". We found that the regular CLM on Wikipedia articles had an average log probability per character of -4.0 with a standard deviation of 0.75. We trained the "internet.interesting" CLM on only those samples from the co-training that had average character log probability of less than -6.25 (less than 3 standard deviations from the mean).

One other important aspect of the CLM computation is that we generate the probabilities for both inserted and deleted text. It is quite likely that in addition to inserting new text, an honest contributor will eliminate vandalism. We do not want to inadvertently suggest reverting an edit that does a service to Wikipedia by eliminating vandalizing text. Also, analysis of comments indicates that there exist several instances in our own training set where a user indicates that he is reverting vandalism.

Lastly, in addition to features that record the log probability for each of the CLMs, we have four nominal features. Two features record the name of the CLM that best models the inserted and deleted text, i.e., the CLM with the highest log probability. For revisions where the delete or inserted text is too short for measuring with the CLM, we then assign the instance with the nominal value of "None". The other features record the name of the Internet trained CLM that best models the inserted and deleted text respectively. Our last feature takes the difference between the best score from one of the Wikipedia trained CLMs and the best score from one of the Internet trained CLMs.

7.2 Handling of training corpus

It is didactic to explain the differences in the evolution of processing the Wikipedia markup from our previous work [184][185]. One main issue in feature extraction for WikiMarkup is what exactly to extract because the markup allows both visual and non-visual modifications. Another issue that is not addressed in much of the previous work on Wikipedia vandalism detection is what to do with templates. Templates allow markup reuse; they are an entity that takes a template name and arguments (either key-value pairs or ordered arguments) in the following format:

{template-name | arg1 | key1=value1 | arg2 | key2|value2 }

The Wikipedia rendering engine MediaWiki will look up the template name and apply the parameters to the markup. One difficulty is that it is not clear how the arguments are transformed visually. Determining this is non-trivial because we would need both the source of the template at the time of the revision plus analyzing the rendering of the attributes (for example, is the attribute transformed into a different representation). Being sensitive to the processing of the markup is important for the new features that we have computed beyond our original detector [185]. It is also important because we must clearly delineate what is a visible or invisible change and where content block boundaries exist. We fear that in some of the previous work that content boundaries were not taken into account. Therefore, this section will take the opportunity to describe our extraction process in great detail.

We first extracted templates from the source and computed the edit difference (diff) for these separately. Only the value of key arguments and un-typed arguments (e.g. arg1 above)

were considered “visual”. We wrote a link transformer that took Wikipedia link syntax (see [186] for a full description) that computed what text would actually be visible to the user from this transformation. We left a crumb in the transformed text to indicate the presence of a link. We then converted the markup without template data using the Bliki engine [187] to HTML. This allowed us to determine the boundary of content blocks so that we could determine the correct sentence boundaries (either terminated with punctuation or at the boundary of a content block). The three main content blocks rendered from the WikiMarkup are paragraphs, list items and table cells. In addition, code blocks were considered separate content blocks. Once the blocks were discovered, we then performed sentence splitting with LingPipe. We then computed the diff using Google diff-patch-match [188]. This allowed us to do word level diffing to construct which segments of text were inserted, deleted or unmodified. We then tagged sentences with which parts were inserted, deleted and unmodified. We denote **modified sentences** having either A) part of the sentence deleted or B) part-of or the entire sentence was inserted (including link text) or C) the sentence had a link inserted.

Stylometric features were computed in the following way. Stop word analysis was performed on all inserted text regardless if it was visual. Readability and PCFG scores were computed on modified sentences. The CLM log probability computations were computed for both the deleted and the inserted text segments. The segments for each insert/delete block were not concatenated together and all blocks of whitespace were converted to a single space.

7.3 Classifier choice

As more research is conducted on the application of machine learning to vandalism detection, researchers typically choose from two types of classifiers: boosting and random trees. As Velasco points out in his PAN 2010 Lab Report [189], classifiers must be able to cope with the severe class imbalance, require little or no processing of the data, perform implicit feature selection and require little parameter adjustment. We agree that these features must factor into classifier selection. Our experimentation has also concluded that discretization [190] of the features actually improves the performance for certain classification strategies.

From analyzing the PAN 2010 competition results, two classifiers appear to work well for this problem: RandomTrees and LogitBoost. We experimented with RandomTrees and did not observe similar performance relative to boosting as reported in [189]. We have found that LogitBoost still performs the best for our features with the base classifier set to a Decision Stump and iterated 500 times.

7.4 Evaluation

This section we will detail our evaluation set. In our evaluation, we use an unbalanced dataset similar to the ratio of vandalizing and regular edits that Wikipedia receives. We also focus on those modifications that either introduce new text or modify existing text. This evaluation will not evaluate the efficacy of the classifier on edits where only text is deleted because it only accounts for 0.43% of vandalism. Similarly, we will not attempt to classify purely template changes because these modifications only account for 8.79% of vandalism and our features are optimized for the most offensive type of vandalism, which is generally visible.

We will evaluate the classifier against many different metrics. We will first determine the precision, recall and the F1 measure (the harmonic mean of the precision and recall) of the classifier. We will also evaluate our classifier with respect to the area under the ROC curve (AUC). This metric was used for the PAN 2010 Wikipedia vandalism competition. This measure denotes the probability that a randomly chosen positive instance, e.g., vandalism, will rank higher than a randomly chosen negative instance. The metric has spurred some debate to its value [191], but we will use it as a means for comparison to our previous work.

We use the 2010 PAN Wikipedia vandalism corpus pan-overview to quantify the benefit of stylometric analysis to vandalism detection. This corpus comprises of 32444 edits on 28468 articles, with 2391 of the edits identified as vandalism by human annotators. Among the different types of vandalism (e.g. deletions, template changes), we focus only on those edits that inserted or modified text since stylometric features are not relevant to deletes and template modifications. Note that insertions and modifications are the main source for vandalism.

We randomly separated 15000 edits for training of $C_{regular}$ and $C_{vandalism}$, and 17444 edits for testing, preserving the ratio of vandalism to non-vandalism revisions. We eliminated 7721 of the testing edits to remove revisions that were exclusively template modifications or deletions and maintain an imbalanced ratio of vandalism and regular edits for a total of 9723 edits, of which 8548 are regular and 1175 are vandalism. For each edit in the test set, we compute the probability of each modified sentence for $C_{regular}$ and $C_{vandalism}$ and generate the statistics for the features described in Section 7.1 . We compare the performance of the stylometric features

Features	Precision	Recall	F1	AUC
Baseline	74.2%	43.0%	54.4%	91.7%
+SW	75.9%	48.9%	59.5%	92.5%
+PCFG	74.3%	48.9%	59.0%	93.0%
+CLM	76.7%	57.0%	65.4%	93.5%
+SW+PCFG+CLM	**78.5%**	**59.8%**	**67.9%**	**94.5%**

Table 12: Results on an unbalanced test data.

against a baseline classifier that is trained on meta data, lexical, sentiment and language modeling features using 10 fold stratified cross validation on the test set.

For the evaluation features for PCFG, we use the values of the scores computed in [185]. Surprisingly, although InfoGain indicates that the differences on the measures between the parser trained on regular edits and the composite feature that takes the best scores from the parsers trained on $D_{\{vandalism,\ twitter,\ YouTube\}}$, the combination of these features with the LogitBoost classifier actually performed worse than the originally computed features. This will be discussed further in Section 7.6.3 .

7.5 Results

Table 12 presents the results of stratified 10 fold cross validation on our testing set. We present the baseline results, with the addition of the CLM features (denoted +CLM), PCFG features (denoted +PCFG), stop word features (denoted +SW) and all the features combined. Table 13 presents the most informative features according to InfoGain.

Feature	Value
Total number of author contributions	0.107
Difference between the best CLM score from the Wiki trained CLMs and the best CLM score from the Internet trained CLMs	0.102
Which CLM best modeled the inserted text	0.101
How long the author has been registered	0.099
How frequently the author contributed in the training set	0.097
If the author is registered	0.089
Stop word Classifier feature: trained with non-empty instances, cost-sensitive SVM	0.049
Stop word Classifier feature: trained with empty instances, cost-sensitive SVM	0.048
Which Internet trained CLM best modeled the inserted text	0.039
Difference in the maximum PCFG score s	0.038

Table 13: Top 10 ranked features on the unbalanced test data by InfoGain.

7.6 Discussion

7.6.1 Evaluation results

We observe, for the first time, features that we have computed on the text difference of the revisions that are more informative than editor meta data including if the editor is registered, how long they have been registered and how frequently they revised articles in Wikipedia in the sampling period.

Our features improve the baseline performance. Stop words improve both the recall (48.9%), precision (75.9%) and F1 measure (59.5%) over the baseline classification. Similarly, adding only the PCFG features increases recall (48.9%), precision just slightly (74.3%) and F1 measure (59.0%). The improvements over the baseline suggest that stylometric analysis does provide us with a means to differentiate regular and vandalizing language. The most impressive gains of a single set of features is the CLM features. Adding the CLM features increases the recall (57.0%), precision (76.7%) and F1 measure (65.4%). Combining all the stylometric

features, we improve in all the measures over the addition of CLM features: roughly 2% for recall, precision and F-measure. From these results, we draw the following conclusions:

- There are indeed unique language styles in vandalism that can be detected with stylometric analysis .
- Rather unexpectedly, deep syntax oriented features based on PCFG bring a much more substantial improvement than language models that capture only shallow lexico-syntactic patterns.
- Stop words provide another stylometric analysis that differentiates between vandalism and regular edits. Training a cost-sensitive SVM classifier on the training data, we observe an increase in both precision and recall.
- CLMs appear to substantially increase the performance of our baseline classifier. This improvement appears to support our hypothesis that the authors of vandalism employ a language that is shared across different contexts in the Internet. Interestingly, we do not need to train a CLM specifically on vandalized text to achieve these gains in performance.

7.6.2 Stop words

Table 14 lists the InfoGain values for different combinations of classifiers (SVMs, BayesNet and NB), training set (with empty instances or no non-empty instances) and if the classifier was adjusted with a cost sensitive classifier. We choose only to apply the cost-sensitive approach to the SVM approach.

Training set	Classifier	Cost sensitive adjusted?	InfoGain
Non-empty	SVM	Yes	0.049
With empty	SVM	Yes	0.048
Non-empty	BayesNet	No	0.036
With empty	Naive Bayes	No	0.261
Non-empty	Naive Bayes	No	0.260
Non-empty	SVM	No	0.007

Table 14: InfoGain for various settings of our stop word classifier on our test set.

you ever hi so like really why thats shes know	0.023096 0.0156 0.008947 0.008482 0.008251 0.007831 0.007755 0.007651 0.007372 0.007235	Is you a he hi I like ever of so
Top ranked features correlated with vandalism according to a linear SVM	Top 10 features according to InfoGain for our stop word feature space	

Table 15: Salient features for stop word author identification.

From Table 14, we see a 7 fold improvement when applying a cost-sensitive classifier. The difference in InfoGain also correlates with the increase of the F-measure for the classifier on the non-empty training data.

The amount to weight mistakes for the cost sensitive classifier is not trivial. For certain classification schemes live Naive Bayes, the cost-sensitive approach will work better if the penalties for false positives are weighted more severely than the penalties for false negatives. Conversely, support vector machines are quite precise for this task, but have low sensitivity.

Results from an experiment conducted on different penalties indicate that penalizing false negative five times more than true positives yielded the highest F-measure. Interestingly, the effect of increasing the false positive rate did not diminish the information gained by the attribute, but rather, increased it. We hypothesize that classifiers can tolerate this increase in false positives because other features such as the editor's total contributions can prevent misclassification since the classifier can rely on the sensitivity of these features.

One last observation that explains the effectiveness of our stop word classification attribute is which features were most informative. Recall that the stop word classifier is trained on a feature space that entirely consists of stop words. Table 15 presents the ranking of the features according to the utility to an SVM and InfoGain. The rankings shed some insight into why this approach can improve vandalism detection. The words that correspond with the vandalism according to the SVM clearly do not belong in Wikipedia articles. Second person pronouns like "you" and seemingly informal words such as "hi", "really" are more appropriate for colloquial conversations, not for educational purposes. Many of the words in themselves are not offensive, and thus, do not immediately raise any obvious alarm in previous systems.

7.6.3 PCFG results

We found that the most informative features derived from PCFGs were computed taking by taking the difference between the probability scores generated by sentences from $D_{regular}$ and the best score from "best.vandalism" ($D_{vandalism}$, $D_{YouTube}$ or $D_{twitter}$). Including the parser predictions trained from MySpace and 4chan did not improve the InfoGain scores in our evaluation. We found that employing multiple parsers from treebanks that were built from

	Min	Max	Stdev	Sum	Mean
Diff between $D_{regular}$ and best.vandalism	**5.20%**	**4.80%**	1.50%	4.00%	**5.60%**
Diff between $D_{regular}$ and $D_{vandalism}$	4.70%	4.60%	1.30%	**5.40%**	5.40%
Original computation	2.50%	3.80%	**1.60%**	3.20%	3.80%

Table 16: InfoGain for diffs for different parsers across all our measures.

sentences deriving from the same category, i.e., categorical treebanks, was not as informative as simply comparing our "vandalizing" parsers with the parser trained from $C_{regular}$. Lastly, we did not gain any benefit from clustering and combining sentences from Internet sources with sentences from vandalizing edits.

The InfoGain presented in Table 16 shows an increase of at most 2.7% over our previous results. We hypothesize that the reason for this increase in the InfoGain is attributable to a new processing pipeline coupled with the Internet co-training. The original work computed the diff on the line level, not the word level. Manual inspection of these diffs show the introduction of sentences that were unaffected. Since our features represent a computation performed on all modified sentences, this may have adversely affected the features. For example, if the longest sentence in the line diff was not one that was modified, the score of the min feature would likely be the difference in probabilities of this sentence with respect to the parsers. We also recognized that the original software to do the WikiMarkup conversion to plain text appears to have computed the plain text incorrectly by misplacing the words. In particular, it had tremendous difficulty parsing templates and hence, why we went to great lengths to accurately pre-process the text and uncover the modified sentences.

Features	Recall	Precision	F1
“Best” PCFG features	**35.40%**	70.90%	**47.20%**
Original computed features	34.20%	**71.00%**	46.20%

Table 17: With the RandomForest classifier, we observe an improvement using the "best" PCFG features as ranked by InfoGain compared to the orginally computed features

We note in Section 7.4 that in fact, our original PCFG scores performed better than the best ones ranked by InfoGain when used by the LogitBoost classifier. The impact of higher InfoGain values generally imply improved performance in classification, but not always. For the LogitBoost classifier, it was unable to perform significantly better using the most informative feature for each measure according to InfoGain ranking from the new processing pipeline compared with our original scores. For other classifiers, such as the RandomForest classifier, we see an improvement (see Table 17) if we use the PCFG features for each measure with the best InfoGain instead of the values originally computed.

7.6.4 Character-level language models

The best new features for vandalism detection presented in this chapter are the CLM generated features. Error: Reference source not found shows the InfoGain values for the top 5 CLM features. This is the first set of features we produced that is more informative than if the editor is registered or not. The best performing features were the nominal feature of the model that best represented the inserted text and the feature that is the difference between the best probability parse from a CLM trained on Wikipedia articles and a CLM trained on Internet sources. The third most informative feature was which Internet trained CLM had best modeled the inserted text. Interestingly,

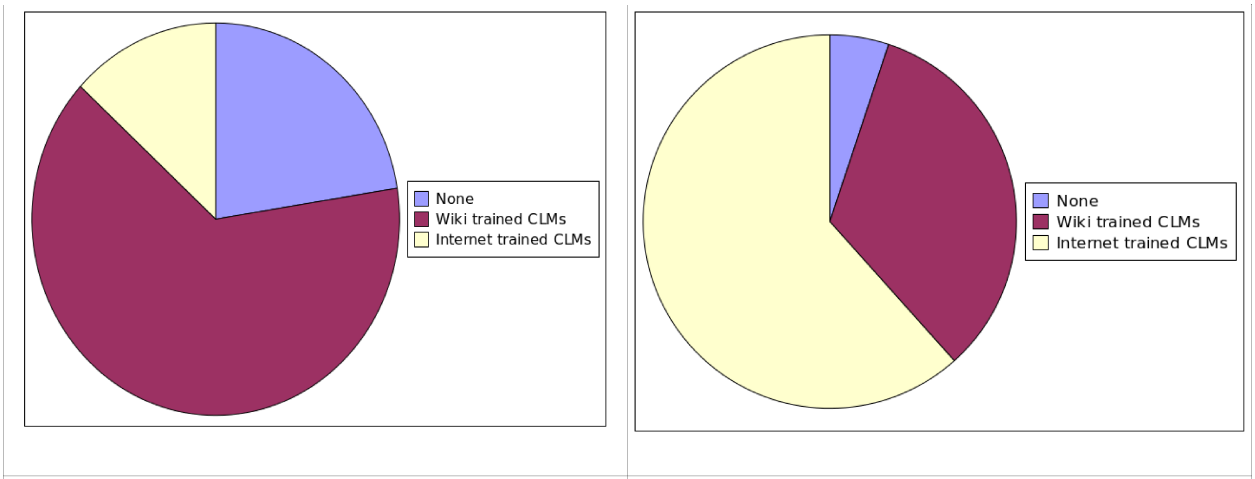
Difference between the Wiki and Internet/Vandalism trained CLMs	0.103
Which Wiki or Internet CLM best modeled the inserted text	0.102
Which Internet trained CLM best modeled the inserted text	0.039
The log probability of the Wiki trained CLM on deleted text	0.026
The log probability of the YouTube trained CLM on deleted text	0.026

Table 18: Top 5 InfoGain ranked attributes derived from CLMs.

although the top feature is derived from inserted text, the following most informative features focus on deleted text. The next two are specifically the features representing the log probability for the CLMs trained on Wikipedia and YouTube on deleted text.

Figure 14 shows the distributions for the nominal feature based on which CLM best modeled the inserted text performed. As we observe, only 14% of non-vandalizing edits were best modeled by Internet trained CLMs. Conversely, we see the Internet trained CLMs provided the best prediction for 61% of the vandalizing edits. Therefore, the classifier is able to effectively use the information gleaned from this attribute to better discriminate between the two classes.

Closer inspection of the distribution of values across the third most informative attribute (which Internet trained CLM best modeled the inserted text) are shown in Figure 15, the CLM trained on 4chan provided the best modeling for regular edits, whereas for the vandalizing edits, they are more evenly distributed across the different Internet trained CLMs. Although 4chan does not require its contributors to adhere to the strict guidelines of Wikipedia editors, after manually inspecting small samples from 4chan, YouTube and Twitter, 4chan comments tend to be better written (although the quality is much more crass and crude than what we expect from Wikipedia articles). With this observation, it is not surprising that the CLM trained from the



A) Which CLM best modeled inserted text for regular edits

B) Which CLM best modeled inserted for vandalizing edits

Figure 14: Which category of CLM (either trained from Internet sources or Wikipedia articles) best modeled the inserted text.

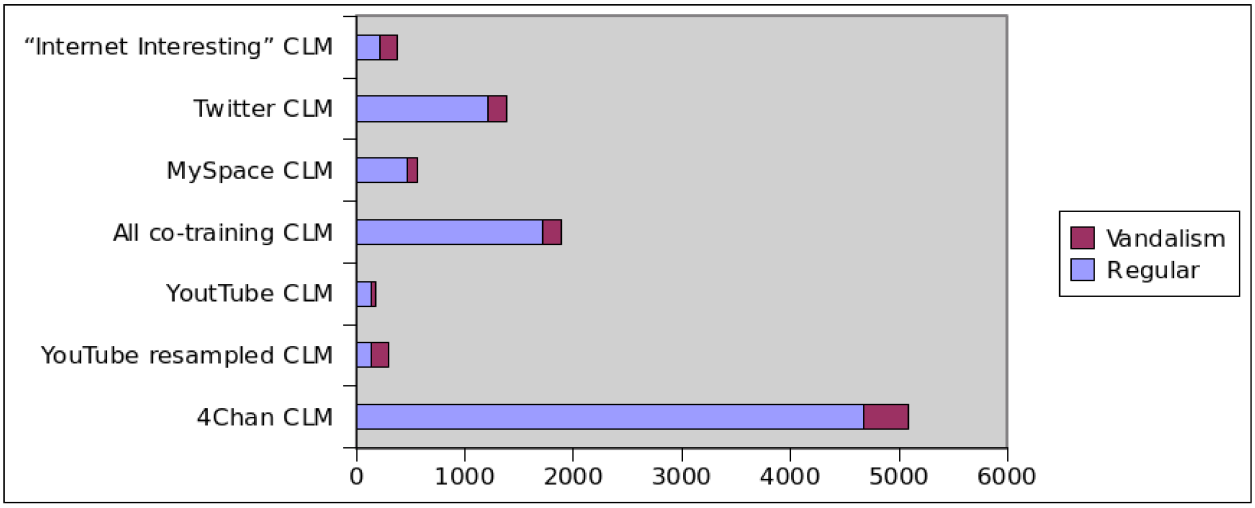


Figure 15: Which Internet trained CLM modeled inserted text the best.

4chan comments better model Wikipedia articles and provides some utility, although it is in essence a subset of the more informative nominal feature that is the name of the CLM that modeled the inserted text best.

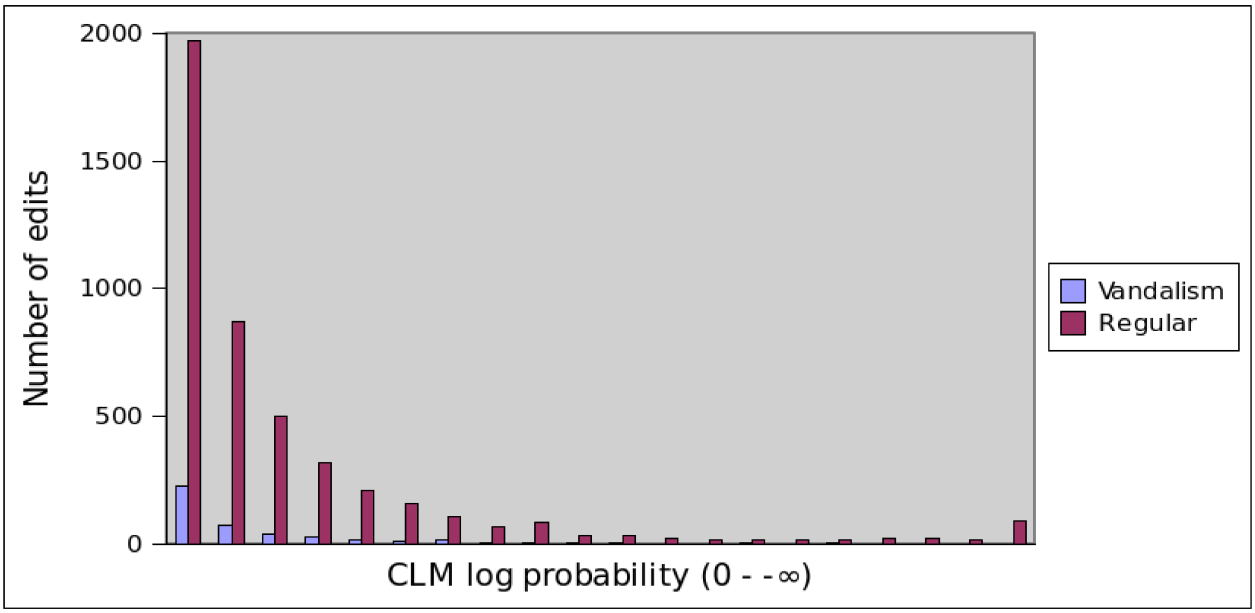


Figure 16: Distribution of values for the CLM trained on Wikipedia articles on deleted text. Note that the distribution for regular edits has a longer tail.

The small gain in adding the log probability from the CLMs for the deleted text is surprising since the best feature is derived from the diff of the CLMs for the inserted text. One possible explanation is that honest contributors who are contributing edits to Wikipedia will eliminate "silly vandalism". For vandalizing contributions, we observe that the proportion of edits that are best explained by Internet trained CLMs drops from 61% to 11%. This seems plausible because a vandal will not go out of his way to clean up articles only to vandalize them further. Rather, what he is deleting is probably "good" text that he wants to dispose of. Another explanation is that the correlation is spurious and honest editors simply delete more text. This observation is supported by our earlier work [184] where we observed 12.62% of non-vandalizing edits involve deletion. The distribution of CLM scores on deleted text will likely exhibit a more uniform distribution than vandals who are less apt to delete text unless they are

intent on “blanking” (removing all the contents of) the page. Figure 16 shows an the distribution of probabilities for the CLM trained on articles from main categories.

For the purposes of this evaluation, we did not train any CLM on the vandalizing text in our training set. The rationale for this decision came from the limited samples that we had in terms of character diversity. Rather, it appears that our hypothesis that A) vandals employ language that is expressed in many different places across the Internet and B) we can accurately model the character-level trigrams of articles from Wikipedia is suggested by our results.

7.6.5 Features and techniques that did not work well

This section will discuss different techniques and features that did not yield substantial value for the benefit of readers who may pursue similar ideas.

Certain hallmarks of style did not provide substantial benefit for classification. These include spelling errors, counting common misspelling mistakes and readability scores such as Flesch-Kincaid. Spelling errors are difficult to detect because building a comprehensive dictionary for Wikipedia is quite an undertaking considering how many words are simply not contained in standard dictionaries. In addition, the English language Wikipedia does contain articles with foreign language text, which is appropriate for the article. Common misspellings are not a hallmark of vandalism as is obscenities and slang speech. Also, readability scores like Flesch-Kincaid provide little benefit as well. We suspect this is due to the following features of revised edits. First, all inserted text is not necessarily sentences, but could be list or bulleted items. Second, vandalism that inserts long tracts of repeated letters may increase the syllable count artificially. Lastly, it is likely only looking at one sentence is simply not large enough for

a proper sampling. Not all non-vandalizing sentences are required to be of significant readability in order to be a worthy contribution.

Lastly, we experimented with different ways of structuring the classifier. We note that the primary profile of a vandal is an unregistered user or a newly registered user with little history. Cluebot [192] is an automated bot on Wikipedia that employs many different heuristics to identify and revert vandalism. It will score an edit and if it deems it to be vandalism, it will not revert a revision if either of the following two conditions apply: if the user is registered and has at least 50 edits or if the user is not registered but the IP address has logged over 250 edits. Presumably, the rational behind these rules is that if the person at the IP address or the registered user has vandalized Wikipedia regularly before, he would have been blocked by this point of time. We expanded on this heuristic in two ways motivated by analysis of our data. First, if the user was very active during our sampling period, we did not classify the edit as vandalism. Second, if the user or IP has contributed to the article 10 or more times, we did not label the revision as vandalism.

We applied the technique in two ways to our dataset. First, we applied the heuristic solely to our classifier decision: only labeling revisions as vandalism if it did not satisfy any of our four criteria. Secondly, we eliminated both testing and training examples from our dataset that satisfied at least one of the four criteria. The motivation here was that the population our classifier should focus on are the examples that do not meet any of our "Cluebot-esque" criteria. On average, this eliminated 68% of the training data. We found that for the baseline features, this yielded a 29% improvement in overall accuracy for the Naive Bayesian classifier. The

performance suffered, however, when we applied this technique with the LogitBoost classifier and the additional stylometric features. The LogitBoost classifier can leverage the new features and perform its own feature selection that it negates any utility provided by these simple heuristics and training methods.

7.7 Related work

[149] presents the first approach that is linguistically motivated. Their approach was based on shallow syntactic patterns, while ours explores the use of deep syntactic patterns, and performs a comparative evaluation across different stylometry analysis techniques. It is worthwhile to note that the approach of [149] is not as practical and scalable as ours in as it requires crawling a substantial number (150) of web pages to detect each vandalism edit. From our pilot study based on 1600 edits (50% of which is vandalism), we found that the topic-specific language models built from web search do not produce stronger result than PCFG based features.

The standard approach to Wikipedia vandalism detection is to develop a feature based on either the content or meta data and train a classifier to recognize it. A comprehensive overview of what types of features have been employed for this task can be found in [193]. Other approaches to vandalism detection include WikiTrust, a reputation system for Wikipedia authors, that focuses on determining the likely quality of a contribution, which is useful in identifying vandalism [194].

7.8 Conclusion and future work

This chapter presents new features that increase the performance of Wikipedia vandalism detection. The application of character-level language models trained from Wikipedia articles

and Internet sources had a substantial impact on the improvement of classification performance. What is encouraging about this particular set of features is that the training does not rely on examples of vandalism, but can be bootstrapped from existing Internet sources and article revisions from trustworthy editors. The models require significantly less space than language models trained on words and does not require per-article searches of [149]. We were also able to enhance the informative value of our PCFG features by training additional parsers on Internet sources. We were also able to create a salient feature based on stop word analysis, a technique used in author identification.

Where the "Got You!" project [149] was the first work to use Internet sources to enhance models for detecting what *should be* inserted into text, this work is the first to use Internet sources to identify what *should not* be inserted into Wikipedia articles. This work suggests that the language employed by vandals is also expressed across the Internet in different forums. The character-level language models successfully allow us to sample this text and learn from it. The usage of Internet co-training allowed us to also build syntax parsers that provide greater InfoGain than simply training on the vandalizing and non-vandalizing text. The main conclusion we draw is that with careful selection of Internet co-training, we can potentially anticipate vandalism.

The stop word analysis sheds some light into the language expressed by vandals. We observe that clearly some of the more salient stop words like "hi" and "obviously" are not particularly appropriate for Wikipedia, regardless of which article is being revised. This is not to say that previous approaches would not be able to identify such vandalism: for example, the "Got You!" system would likely assign low probabilities to these words. The evidence suggests that

there do exist words that vandals employ which are non-offensive but simply not appropriate for Wikipedia's stylistic guidelines. The question is how do we effectively identify these types of words? One approach could be to sample various sources to get an idea of the frequency of the word (optionally taking into account the part of speech for additional information). If the observed rate of the word is significantly higher in other sources than it is in Wikipedia, then this may suggest this word is not thematically appropriate for Wikipedia. Another technique may be to find the non-content bearing words that appear significantly more frequently in Wikipedia than in other contexts. We can then score revisions on how many "good" encyclopedic terms it contains. Like the Internet co-training, there may exist text sources that will provide adequate contrast to what is expected in Wikipedia.

Chapter 8 PhorceField

Passwords are by far the most common form of authentication in use today. Almost all online services, including social networking, email, e-commerce, and online banking, use passwords as their only authentication factor. Users often re-use passwords across multiple sites, so the exposure of a single password could compromise a user's entire financial identity, including bank account, credit card, e-commerce accounts, and investment accounts. Therefore, it is paramount to protect the secrecy of user passwords. The issue, however, is that nearly all password authentication schemes require user vigilance, something that users are largely *unable* to do [42].

Phishing attacks steal user passwords by tricking victims into entering their password into a prompt that the phisher controls [195]. A phisher lures victims to his web page, which mimics another web page trusted by the victims, and asks the victims to enter their personal information and login credentials for other web sites. Over 3.6 million people in the U.S. lost money to phishing attacks in 2007, losing over 3 billion USD [196].

Phishing attacks exploit users' inability to distinguish legitimate websites from fraudulent ones [197][42]. Researchers have proposed several "trusted password prompts" and usability aids to help users make this distinction, including secure attention keys [198], security indicators [42], and security skins [199]. None of these solutions is effective, though, because they all require users to take active measures, such as checking for the presence of a security indicator.

Even with these aids, the vast majority of users are not able to avoid phishing attacks [42].

Many factors impede the user's ability to recognize fraudulent websites: users expect software to be buggy and error-prone, they do not expect to be phished, and they are conditioned to only superficially observe browser security mechanisms [200]. As Karlof et al. summarized: "These tendencies suggest that we cannot rely on users' abilities to detect social engineering attacks and respond appropriately, and we must design defenses accordingly" [200].

This chapter presents PhorceField, a phishing-resistant password ceremony that relies on user laziness rather than vigilance. With PhorceField, a legitimate password prompt has access to a secret set of images that enables it to create an easy-to-use password prompt. Phishers do not have access to the secret images and hence must present victims with a fundamentally different and more difficult interface. Users cannot ignore the differences, as with previous schemes such as SiteKey, and hence cannot slip into dangerous click-whirr behaviors [200]. Even if users do attempt to interact with the phisher's page, though, the phishing page requires so much effort that the victims give up in frustration before they can reveal their password.

PhorceField exploits well-known strengths and weaknesses of human memory [201] [202]. During a normal PhorceField login, users must perform an image recognition task, e.g. discriminating the user's password images from a small set of candidate images, which is relatively easy for humans. During a phishing attack, though, victims must perform an image recall task, e.g. remember all images from the password without visual cues, which is substantially harder. Furthermore, PhorceField is designed so that victims will experience memory interference during a phishing attack, causing additional frustration and error. To our

Metric	SiteKey	Phorcefield
Percent users revealing their entire password	92%	**0%**
Average amount of password revealed	92%	**20%**

Table 19: Success rate of phishing attacks on PhorceField and SiteKey. The SiteKey statistics are derived from Schechter, et al.. Since SiteKey users reveal all-or-nothing of their password, the average amount of password entropy revealed is the same as the percentage of subjects who revealed their password. In our PhorceField study, all users were assigned passwords with 70 bits of entropy, and they revealed an average of 13.8 bits of information about their password.

knowledge, PhorceField is the first password ceremony to take advantage of these properties of human memory.

PhorceField is easy to use. The PhorceField user experience is identical to a cognometric graphical password [203], which have been shown to have high usability and good password memorability. Thus, this chapter focuses only on PhorceField's security against phishing attacks.

We have conducted a user study to evaluate the security of PhorceField against phishing attacks. Participants used PhorceField for one week and were then presented with a simulated phishing page that attempted to steal their PhorceField password. We made special effort to ensure the ecological validity of our study and to avoid participant bias. For example, participants worked on their own computers in their normal environments, and we told participants that the study was focused on "usability" instead of security.

As the user study results summarized in Table 19 show, PhorceField users presented with a phishing attack give up before they are able to reveal their entire password. Participants in our study revealed, on average, only 13.8 bits of information (out of 70 bits of entropy) about their password, and no participant revealed his entire password. This is a substantial improvement

over recent results on SiteKey, the current industry standard for preventing password phishing, that show that 92% of SiteKey users will reveal their password to a phisher [42].

PhorceField combines two existing technologies – client-side secrets (e.g. secure HTTP cookies) and graphical passwords – in a novel way to achieve a level of phishing-resistance that neither technology achieves on its own. Our results demonstrate that previous schemes based on client-side secrets, such as SiteKey [204] and Dynamic Security Skins [199], are not extracting the full benefit of the client-side secret. PhorceField could serve as a drop-in replacement for the login ceremony of either of these schemes, providing substantial improvement to their phishing-resistance.

Our study also offers upper-bounds on the amount of time and effort phishing victims will devote to a phishing web page. Since our phishing "attack" lacked all the cues of a normal phishing attack, and since we designed the experiment to give participants an incentive to cooperate, our results are quite conservative, but may still serve as guides for phishing defense designers. Participants in our study spent an average of 11.5 minutes on our phishing page, including some participants who spent over 25 minutes. Several participants visited the page more than once. These observations suggest that future phishing defenses should assume users will cooperate with a phisher for several minutes and execute several non-trivial tasks.

In summary, PhorceField demonstrates three new techniques for designing phishing-resistant password ceremonies:

- It forces phishers to present a fundamentally different interface to their victims. This gives users a better chance to detect the attack. With previous schemes, a phisher can present an interface that differs only superficially from a legitimate one.
- It forces phishers to present a much more difficult-to-use interface. Even if a victim is fooled by the phishers' attack, she is unlikely to succeed in communicating her password to the phisher.
- It exploits strengths and weaknesses of human memory to make phishing attacks difficult for their victims. Victims of a phishing attack experience memory interference while looking through hundreds of similar images, causing frustration and error.

We make the following additional contributions:

- We present results from a user study demonstrating that PhorceField successfully protected all participants against our simulated phishing attack.
- Our study also provides guideline parameters for designing future defense mechanisms based on user frustration. For example, users will devote an average of about 10 minutes, and up to 30 minutes, on a phishing page.
- We show how PhorceField can easily integrate into existing anti-phishing mechanisms, such as SiteKey and DSS, without introducing any new hardware or software requirements.

8.1 Background

Phishers steal user credentials by tricking victims into revealing private information, such as passwords. In this section, we review (1) why phishing is not solved by existing technologies, such as SSL or malware defenses, (2) that some proposed phishing solutions offer some protection but do not address the core problem of password disclosure, and (3) that previous solutions targeted specifically at preventing password disclosure are not effective.

8.1.1 Non-solutions

Phishing is a separate problem from malware (such as key-loggers), click-jacking, and other techniques that attackers may use to steal user secrets. Even if a user's computer is free of malware and has effective defenses against click-jacking, a phisher may still trick the user into divulging his password. Therefore, we need separate defense mechanisms to help users avoid phishing attacks.

Phishing cannot be solved solely with cryptography. SSL and PAKE protocols [205] cannot protect the user's password when she types it into a phisher's website. Therefore, we need separate defenses to help users avoid entering passwords into malicious prompts.

Password managers and one-time passwords do not prevent phishing, either. A phisher can defeat a password manager by convincing victims to disable it and type in their master password. Given the fallibility of users [42] and the usability problems with password managers [206], the ruse is likely to succeed. One-time password generators, whether implemented as a special-purpose token, software on a cell-phone, or delivered to the user's phone via SMS, all require users to manually copy the one-time password from the device to their computer.

Phishers can trick users into entering the password into their page instead of the user's intended website, as was done in a real phishing attack against Citibank [207].

Two-factor authentication can mitigate the damage of a stolen password, but it does nothing to protect the password itself. However, banks and other financial institutions clearly consider passwords worthy of protection in their own right, and for good reason. For example, SiteKey establishes a shared secret that subsequently acts as a second authentication factor, but SiteKey also contains a visual cue to help users avoid entering their passwords into malicious prompts, demonstrating that banks are invested in password security, too. Passwords are worth protecting primarily because a password stolen in one context can often be used in another. For example, most users share passwords across multiple sites, so a phisher that steals a user's password on one site can often log into several of the user's accounts. Phishers may also be able to use a stolen password to boot-strap social engineering attacks. For example, a phisher can use a stolen password to pose as the bank when attempting to extract more personal information from a victim. Finally, the same institution may allow users to log in using the same password in several different contexts, some of which do not require the user to present their stored secret. For example, adaptive authentication schemes grant users different levels of access depending on the number and nature of authentication factors presented [208]. For all these reasons, it is important to give web service providers and users tools to help them prevent accidental password disclosures.

Anti-phishing toolbars [209], spam detectors [210], phishing site blacklists and white-lists [211], and other reactionary mechanisms for fighting phishing offer some protection, but

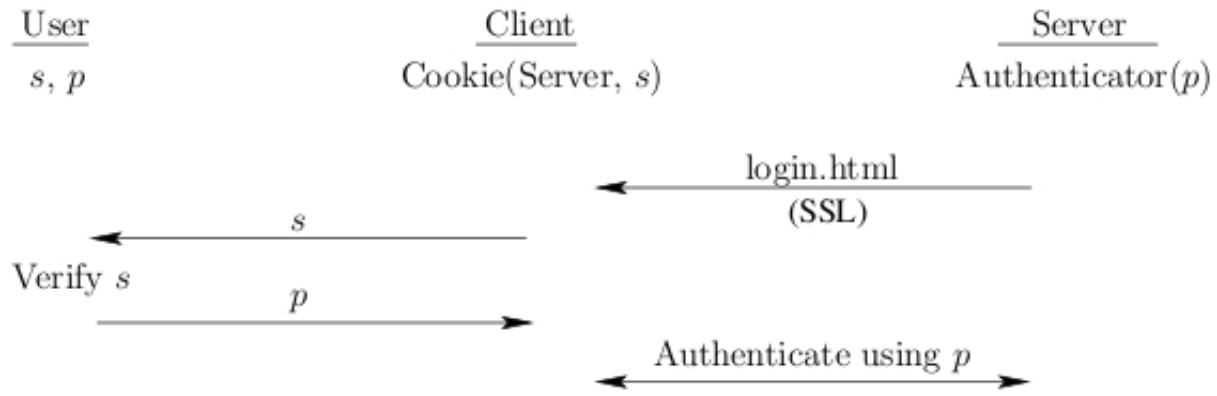


Figure 17: A generalized variant of the SiteKey and Dynamic Security Skins login ceremony. The user's secret image, s is stored securely on her machine and is only accessible to legitimate password prompts, i.e. prompts from "Server" in the case of SiteKey, or prompts presented by the DSS plug-in in the case of Dynamic Security Skins. Users are supposed to verify that s is displayed in the prompt before entering their password. Once the user has entered her password, p, any password-based authentication protocol can be used to authenticate the user to the remote server.

phishers have developed many techniques for evading these defenses [212], so millions of users fall victim to phishing each year [196].

The solutions above fail to prevent phishing because they do not address the core weakness exploited by phishers: users cannot determine whether a password prompt is legitimate or malicious.

8.1.2 Prior phishing solutions

Researchers have made several attempts to create password schemes that will help users distinguish legitimate prompts from malicious ones.

Secure attention keys allow users to summon a trusted password prompt by pressing a special sequence of keys – typically Ctrl-Alt-Del. Secure attention keys will not prevent phishing since phishers can easily convince users to skip the special key sequence.

Dynamic Security Skins (DSS) [199] and SiteKey [204] are conceptually similar schemes that use a client-side secret to help users recognize legitimate password prompts. In these schemes, a secret image known to the user is stored on the user's device. This image is presented to the user as part of the standard password prompt. The user is supposed to verify the image is present before entering her password. Figure 17 shows the protocol for logging in using these schemes. SiteKey and DSS differ in the location of the image: SiteKey displays it above the password entry field, DSS displays it behind the field.

Unfortunately, 92% of SiteKey users will ignore a missing image and enter their password anyway [42]. We could find no published user study evaluating phishing attacks against Dynamic Security Skins, but it also depends on user vigilance for security and so is likely to offer little protection against phishing.

8.1.3 Client-side secrets.

SiteKey and DSS also differ in their machine registration ceremonies, i.e. in the protocols they use to establish the client-side secret. When a SiteKey user attempts to login from a device without the secret, she is asked to answer a "security question", such as "What is your mother's maiden name? " Upon successfully answering the question, the remote website stores the secret cookie on the new device. Thus, the remote website and all of a user's devices share the same secret. In DSS, the secret background image is chosen randomly each time the DSS software is

installed on a new device. Thus devices do not share secrets with each-other or with any remote website.

SiteKey's machine registration ceremony is vulnerable to phishing attacks. Karlof, et al., found that over 92% of users would tell a phisher the answers to their security questions [200]. Even worse, phishers can often simply guess the answers to the security questions [213]. The answers to some questions can be guessed with over a 50% success rate [213].

Despite these problems, SiteKey and DSS suggest a strategy for solving the phishing problem: (1) Develop secure ceremonies for establishing a client-side secret, and (2) Create login ceremonies that prevent password phishing by leveraging the secret established during a machine registration ceremony.

Researchers have already begun to develop better machine registration ceremonies [200] [214]. Karlof, et al., proposed an email-based secure machine registration ceremony [200] that reduced the success of phishing attacks to about 47%. Parno, et al., proposed to use a trusted auxiliary device, such as a Bluetooth enabled cell phone, to establish a secure channel between the user's computer and the bank's servers [214]. After establishing the secure channel, the bank can store a cookie on the client computer to avoid the need for the auxiliary device during future logins. This machine registration scheme would be virtually immune to phishing attacks, but requires both additional hardware and software support. One could also create secure machine registration schemes based on USB tokens or biometrics and fuzzy extractors [215], although we are not aware of any specific proposal along those lines.

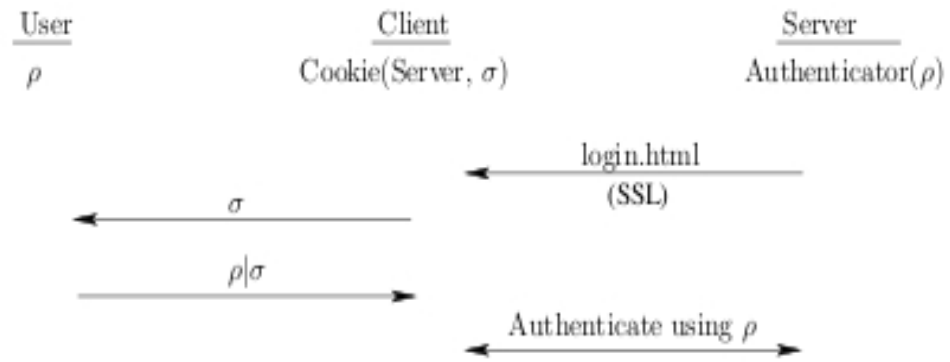


Figure 18: The PhorceField ceremony. The user's secret image set, σ, is stored on her device and inaccessible to phishers.

These results demonstrate that secure machine registration is acceptable to users, is an active area of research, and is likely to improve in the future. PhorceField provides the other half of the solution: a phishing-resistant password ceremony.

8.2 PhorceField

Our goal is to create a password ceremony such that attackers cannot present a usable password prompt to their victims. Therefore, legitimate prompts must have access to some secret that is not known to attackers. Furthermore, passwords cannot consist of letters, numbers, etc., since phishers can present easy interfaces for inputting those symbols. PhorceField satisfies the above two requirements by using a cognometric graphical password scheme and by storing the graphical password images securely on the user's device. Figure 18 shows the PhorceField login ceremony and Figure 19 shows an instance of a PhorceField password prompt.

Cognometric graphical passwords present users with a set, σ, of images and users log in by clicking on the images in a certain sequence or by clicking on a certain subset of images. We

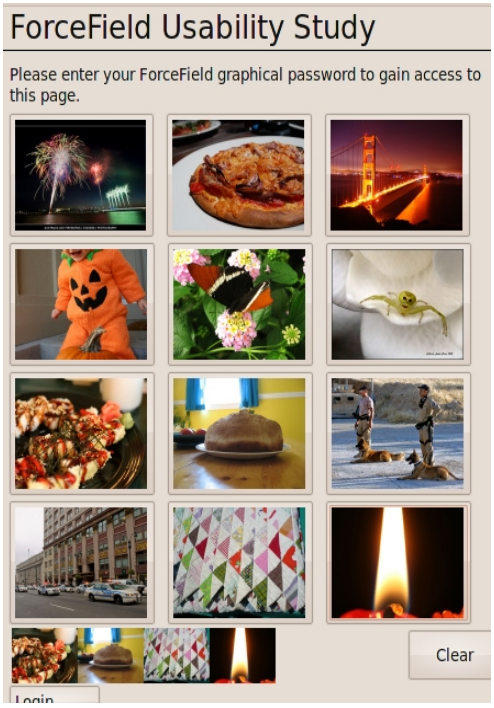


Figure 19: An example PhorceField password prompt using Creative Commons licensed images from the Flickr photo-sharing site.

refer to the user's password as ρ. In PhorceField, the set σ must be drawn from a much larger set, Σ. Thus, the user's password is a word in the language Σ^* but, since the prompt already has access to σ, the user only needs to communicate $\rho|\sigma$, which may only constitute 10-20 bits of information. A phisher that does not know σ, though, must trick the user into revealing ρ, which may require the user to communicate hundreds of bits of information, making the task much harder.

For security, Σ should be as large as possible and, for usability, σ should be small. An implementation could use random art images [216] generated using ℓ-bit seeds, in which case $|\Sigma| = 2^{\ell}$. The Déjà Vu graphical password uses random art images and has good usability [203].

Alternatively, Σ could consist of a large collection of photos or icons, similar to those used in SiteKey [204]. These photos could be drawn from a homogeneous set, such as pictures of faces or landscapes, or could be more diverse, such as pictures of everyday objects. The Passfaces scheme uses only pictures of faces and also has good usability [217], but research has shown that users do not pick good Passfaces passwords – they must be assigned to users to ensure they are random [218]. The choice of images in Σ will affect the usability and security of a PhorceField deployment. This chapter evaluates a PhorceField implementation using photos of everyday objects.

For our prototype implementation, we chose $|\sigma|=12$, since this makes it easy to enter passwords on cell phones and other mobile devices with the standard phone keys 0-9, "*", and "#", and Σ consisted of 188218 creative-commons licensed images collected from the Flickr photo service [219]. There are over 100 million such images on Flickr, so our prototype could be easily scaled up for real-world deployment [220]. We collected images by searching for 193 different concrete nouns, such as "cow", "flowers", "sky", and "tree". We constructed each participant's σ by selecting 12 concrete nouns and then selecting an image from each noun's image set. Passwords in our implementation consist of a sequence of 4 distinct images (order matters). Our system randomly generated passwords for participants. This yields an entropy of 181 bits for σ, 70.0 bits for ρ, and 13.5 bits for $\rho|\sigma$. Our scraper also downloaded the description, tags and titles for each image, which we use later to develop the phishing attack interface in our user study.

The exact details of passwords are orthogonal to the basic design of PhorceField. For example, the user's password could consist of a subset of the displayed images, and the user would log in by clicking on the images in her subset in any order, as in Déjà Vu [203]. Alternatively, the password could consist of an arbitrary word in . The password system could impose constraints, such as a minimum length, or a minimum number of distinct images, in order to help users avoid bad passwords. The service could also choose ρ for the user, which would guarantee passwords are chosen randomly, albeit at a potential cost in memorability.

Each user's σ must be stored on her device. As described in Section 8.1 , there are multiple ways to do this, each with their own advantages and disadvantages. For the purposes of our analysis and user study, we assume that σ has been securely stored on the user's computer.

8.3 Phishing attacks against PhorceField

We argue in this section that phisher's only have two possible strategies for extracting ρ from their victims: brute force attacks and search attacks.

Brute force attacks. A phisher that does not know σ may attempt to learn ρ by presenting the user with an invalid password prompt. If the invalid prompt contains some images from the user's set, σ, then the user may click on them. Note that in many cases, the user may refuse to interact with a prompt unless it contains exactly the images in her σ but, for this analysis, we pessimistically assume that the user will select all the images in her ρ, even for a badly malformed prompt. By performing this attack repeatedly with different images each time, the attacker may eventually recover all of ρ. The attacker can reduce the number of repetitions

required to complete the attack by placing more images into each prompt. However, if the number of pictures in the invalid prompt is too large, then users will give up in frustration, and the attacker will learn nothing. If we assume that users will give up if asked to examine more than *m* pictures in one prompt, then the probability that a phisher can recover σ after repeating the attack *r* times is $\frac{(mr|\sigma|)}{|\Sigma||\sigma|}$. For example, our user study described in the next section used $|\sigma| = 12$ and $|\Sigma| \approx 2*10^5$. From the results of our study, where we were able to observe how many images a victim will look at and how many times he will return to the attack page. we can estimate that $m \leq 3500$ and $r \leq 10$. With these parameters, the probability a phisher can recover σ with this attack is less than 2^{-30}.

Search attacks. Since brute force attacks will not work, a phisher must trick a user into communicating the images in her ρ to the phisher. This is a search problem: the phisher can iteratively make adaptive oracle queries to the victim in order to narrow down the search space for ρ. The phisher's oracle queries can contain text, pictures, video, sound, etc., and the phisher may request user input in the form of text, selections, images, audio, or video. For example, the phisher could ask the user to provide a textual description of an image in ρ, as is commonly done with current search engines. He could also provide a set of words or phrases from which the user can select the most descriptive, leading to either a tag-oriented search tool or a hierarchical search tool. The phisher could also provide an image-similarity search tool, in which users are repeatedly presented with a set of images and must click on the image most similar to their target image. The phisher could ask users to sketch an approximation of their target image. Naturally,

phishers could provide multiple search modalities to help their victims communicate as much as possible about their password.

We designed PhorceField to resist search attacks in several ways. PhorceField forces attackers to present victims with an interface that is substantially different from a legitimate PhorceField password prompt. Victims of phishing attacks therefore cannot slip into a "click-whirr" response mode [200]; they must actively evaluate the situation, giving them a much better chance of detecting the attack. This benefit will grow as more users become educated about phishing. With SSL, SiteKey, and DSS security indicators, even an educated user may forget to check for an indicator before entering her password. This cannot happen with PhorceField.

PhorceField trains users to distinguish their password images from a small set of distractor images – they do not need to remember every detail of their password images. A phisher, however, must coax enough information out of his victims in order to identify their images within a set of thousands or even millions of candidate images. In many cases, users will be unable to provide more than the most prominent features of their password images, leaving the phisher with potentially thousands of possible matches. He can try to get the victim to sift through the matches, but our user study shows that victims will often give up before finding their password images.

We constructed the database of images used in our PhorceField prototype so that logging in is easy but cooperating with a phisher is difficult. Recall that Σ contains 188218 images representing 193 different concrete nouns, or about 975 images per noun. Constructing the database in this way causes users to suffer from memory interference while looking through

PhorceField parameters	Bits	Text (letters)	Hierarchical (steps)	Iterative (steps)
Prototype ($\|\Sigma\| = 2 \times 10^5, \|\rho\| = 4$)	48	32	12	10
Large, Pictures ($\|\Sigma\| = 2^{27}, \|\rho\| = 6$)	162	108	41	33
Random Art ($\|\Sigma\| = 2^{128}, \|\rho\| = 6$)	768	512	192	154

Table 20: The amount of information a user must communicate to a phisher, and the theoretical minimum amount of effort required to do so through a textual description, through a hierarchical search interface in which the user is offered 16 choices at each step, and through an iterative search interface where the user is shown 32 images at each step and clicks on the image most similar to her target image.

candidate matches for their password images [202][201]. As the phisher presents the user with more and more candidate images that are similar to the victim's target image, the victim loses her ability to precisely identify her target image because it becomes blurred together with the candidates. This leads to errors, as victims may accidentally select a wrong image, and frustration within the victims, causing them to quit cooperating with the phisher before they succeed in finding their password images.

We can also derive conservative lower bounds on the security of PhorceField by analyzing the amount of information a victim must communicate to a phisher. Table 20 shows the entropy of ρ in three different PhorceField instantiations: our prototype implementation, an expanded version that uses 134 million pictures and passwords of length 6, and a version using random art images and passwords of length 6. From these entropy values, we can derive lower bounds on the effort users must make to reveal their password. For example, using the standard upper bound of 1.5 bits/character for English text [221], we can estimate that a phisher would have to convince his victims to type over 500 characters of text to fully describe their passwords in a random-art instance of PhorceField. If the phisher provided his victims with a hierarchical

search interface, i.e. in which users navigate a taxonomic tree of the images in Σ, with a branching factor of 16, it would still take users dozens of steps to find all their images. Similarly, if the phisher developed an image-similarity based search in which users are repeatedly shown 32 images and click on the image most similar to their target image, users would have to click on dozens of images and would end up looking at hundreds or even thousands of images. These figures assume the phisher is able to build a perfect search interface that extracts the maximum amount of information from each user input, and hence these numbers are very conservative lower bounds. Given that users will not likely write the perfect description of their password images, given the difficulty in building a perfect hierarchy of images in Σ, and given that building effective image-similarity search mechanisms is still an open problem, the actual work factor required is likely several times higher than estimated here.

For large image collections, e.g. with , building the search database required for a phishing attack may be prohibitively expensive. Although we used pre-tagged photos from Flickr in our prototype implementation, a real implementation could collect photos from a variety of sources that did not provide tags or titles for each image, since PhorceField does not use the tag information. In this case, the phisher would have to tag the photos or construct an image similarity index himself. As mentioned before, image similarity search schemes have yet to be shown effective, so a phisher is not likely to pursue that approach. For a textual or hierarchical search interface, the phisher cannot tag (or taxonomize) the images automatically, since this would require image recognition software that simply does not exist, so he would have to farm out the task to humans. Currently, this kind of work costs about 0.01 USD/image on Amazon's

Mechanical Turk, giving an overall cost of over 10^6 USD [222]. By increasing the up-front costs and lowering the yield of phishing attacks, PhorceField can render them unprofitable.

The preceding arguments show that brute force attacks are impractical and that search attacks, no matter how clever, will require significantly more time and mental effort from victims than is required to log in. However, we cannot analytically determine how users will react when presented with a search attack on their password; for that, we need a user study.

8.4 User study

The purpose of our user study is to measure the success rate of phishing attacks against PhorceField passwords. This study does not attempt to measure the usability of the PhorceField prompt or the memorability of graphical passwords – those topics have been explored elsewhere [203]. Our study includes numerous conservative design decisions, so we believe the results of this study represent an upper bound on the success probability a phisher can achieve with this attack.

Participants in our study were told that we were conducting an experiment to evaluate the usability of graphical passwords. After consenting to the study, participants were shown their set σ and password ρ and required to practice entering their password five times. They then downloaded a Firefox plug-in that randomly prompted them to enter their PhorceField password up to 4 times per day. We chose to prompt participants randomly because it was the simplest way to ensure participants logged in often enough to become familiar with their passwords. We needed participants to be familiar with their passwords in order to get meaningful data during the subsequent phishing attack.

Participants were told that the study would last two weeks and that they would receive $20 if they entered their password at least 14 times over the entire study period, $10 if they entered their password between 7 and 13 times, and no compensation if they entered their password fewer than 7 times. After about one week, they received an email from us indicating that we had lost their password and requesting them to visit the study website to help us recover it. We then measured how much time they spent at the site and how much information they were able to divulge about their σ and ρ. Participants were then presented with a debriefing questionnaire.

We were careful to avoid priming the participants about security or phishing in particular. Subjects were told that the study was about usability, not security. All study materials used the name "ForceField" instead of "PhorceField". The attack email clearly came from us and directed the users to the same website that they visited to sign up for the study. Thus our attack was missing all the cues of a real phishing attack and hence we believe that our results are an upper bound on the success rate a real attacker would experience with this attack.

Although usability is not the subject of our user-study, we followed usability best-practices when designing our password prompt. We based our implementation on the results of Moncur et al. [223] Our password prompt requires users to complete a cognometric task, not a drawmetric [224]or locimetric [225] task. Cognometric tasks are generally easier than the alternatives. Furthermore, a phishing attack against PhorceField would require users to select their images from thousands of similar images and is therefore closer to an image recall task than

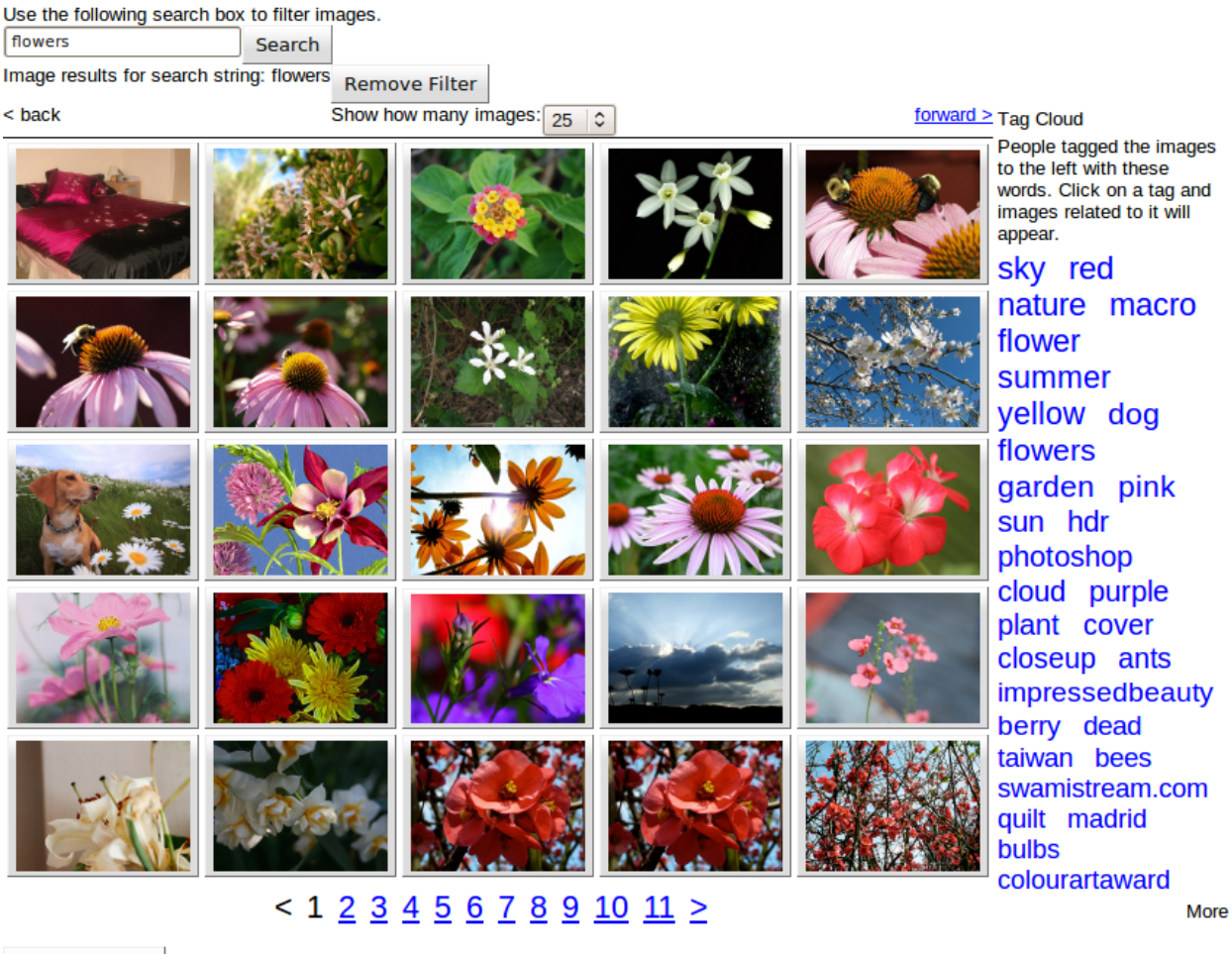


Figure 20: Our phishing attack website. Users could search for images using the search box at the top of the page and could click on tags in the tag cloud on the right-hand side.

an image recognition task. We designed the PhorceField prompt so that users would be trained to recognize, but not recall, their images.

Figure 20 shows a screen-shot of the password recovery page. Participants could use the search box to enter queries and could click on words in the tag cloud on the right-hand side of the page. As is demonstrated in the screen-shot, the search results were quite good because the Flickr photos were so well tagged. The interface defaulted to showing participants 25 images in

a 5×5 grid because previous research has shown that users can search most efficiently with such an interface [226]. However, the success of a phishing attack depends on keeping users on the phishing page as long as possible, so participants were able to choose a different presentation that suited them. The interface never indicated the number of images that participants had to search through, since we felt that would only discourage them. The pagination buttons along the bottom behaved analogously to Google's image search interface [227]: when the user was on results page n>5, the interface provided links to pages n–5 through n+5.

The attack page recorded every user action, so we could tell how long they spent on each page, the search terms they entered, and which images they clicked on. Users could also navigate away from the page and return later. In that case, their exact state would be restored.

We conducted a pilot study to determine the number of participants needed to estimate the mean number of σ images revealed by a user. During our pilot study, thirteen subjects used our graphical password prompt for seven days and received the phishing attack page under the guise that the experimenters lost their password. The subjects revealed on average 0.35 images in their σ (SD =0.60). Therefore, we require 20.7 subjects to determine the average number of σ images revealed in a phishing attack with a 99% confidence interval ±0.34 images [228].

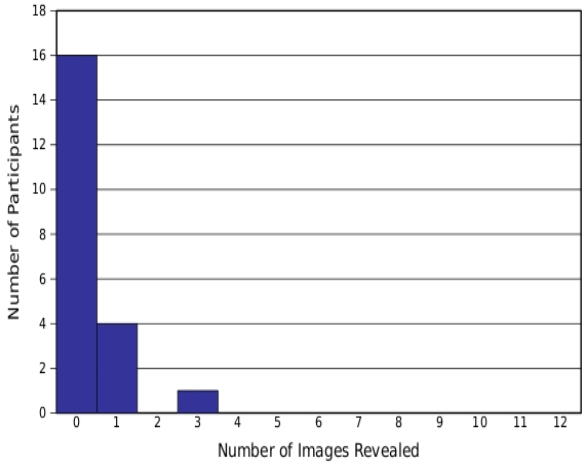
8.5 Results

We conducted our user-study over the course of three weeks. We recruited 45 participants from a Craigslist posting. To ensure that subjects had sufficient exposure to their password, we only considered subjects that successfully logged in at least 4 times and at least

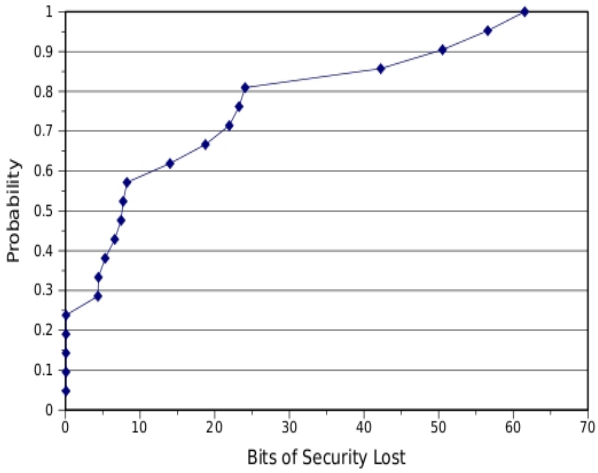
once in the two days prior to receiving the attack email. 23 subjects met this criteria. This retention rate is comparable to other graphical password user studies [223]. We also eliminated from the analysis two subjects who thought our attack email was a real phishing attack. Participants ranged in age from 18 to 39 years and were varied in race (including Asian, African-American, Caucasian and Hispanic) and profession (including students, actors, IT professionals and HR representatives).

We evaluate both the explicit and implicit password information revealed by participants. Participants explicitly revealed part of their password if they found a password image and clicked on it. They revealed information implicitly by searching for images on the phishing site, even if their search was unsuccessful. For example, if a user spends a long time looking through pictures of dogs on the phishing site, then the phisher can infer that one of the user's password images is of a dog.

Explicitly-revealed information about σ. Figure 21 shows the number of σ images our participants were able to find and click on. On average, participants clicked on 0.3 images of their σ. No participant clicked on an image that was in their σ but not in their ρ. Furthermore, as Figure 21 shows, 76% participants did not click on any of their σ images, and the others were only able to find at most three of their σ images, implying that PhorceField offers strong protection for almost everyone. During the attack, several participants clicked on random images out of frustration, but doing so provides no useful information to a phisher.



The number of σ images clicked by participants in our study. No user clicked on an image in $\sigma \setminus \rho$.



The cumulative distribution function of entropy loss for σ from participants in our study.

Figure 21: Explicit and implicit information revealed about σ.

Implicitly-revealed information about σ. Even if users fail to find one of the images in their σ, they may still reveal information about that image through their search activities. For example, if a victim searched for the term "flowers" during a phishing attack on our system, then the phisher could reduce the search space for one image from 188218 to 3117 images.

Furthermore, if that user looks at 10 pages of results without clicking on any of them, then the phisher can conclude that the user's image is not amongst those images. If the user performs a second search on the term "plant", the phisher can intersect the two results sets to further narrow the candidate set.

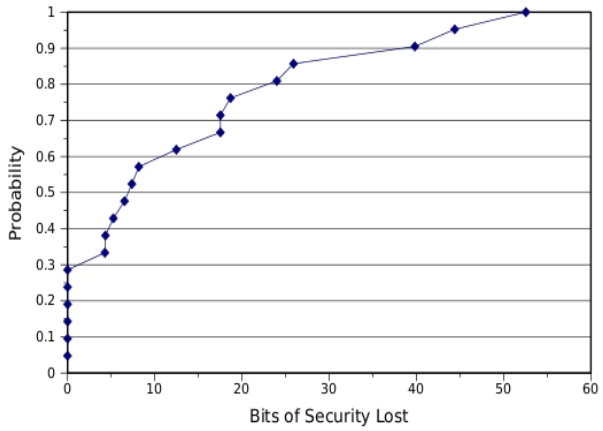
Users may also click on images that are not in their σ but are visually or semantically similar to images that are in their σ. We used the tags on the images users clicked in order to approximate this information, since the tags assigned by Flickr users cover both visual and

semantic aspects of the images. Therefore, we took the tags on each clicked image and added them as search queries during the analysis described below. However, we discovered that many of the tags conveyed contextual information, such as the type of camera used to create the photograph, that would not be apparent to a phishing victim and therefore would not contribute useful information to a phisher. A real phisher could clean the image tags to avoid this problem, but we took a shortcut: For each image a participant clicked on, we computed the set of tags on that image that also occur on an image in the participant's σ and treat those tags as additional search queries performed by that participant. In reality, a phisher would not know which tags occur in the user's σ, so this simplification grossly over-estimates the information gained by a phisher.

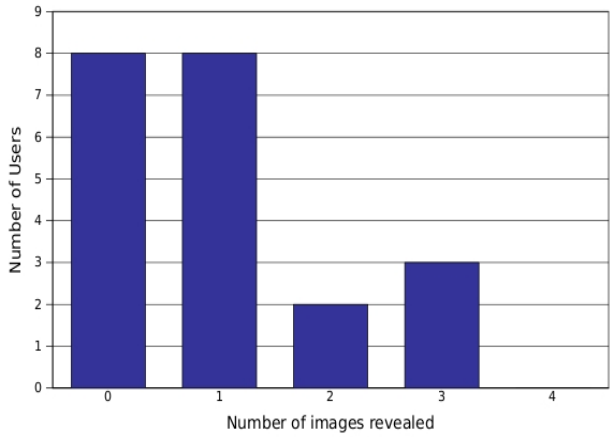
Given the set of search queries performed by a participant, we upper bound the information gained by the phisher as follows. For each search query, q, let S_q be the set of images in the search result set, and let $U_q \subseteq S_q$ be the set of images in S_q that the user never looked at. For search queries derived from tags on clicked images, $U_q = S_q$. For each image $\iota \in \sigma$, let

$$C_\iota = \begin{cases} \{\iota\} & \text{if user clicked on } \iota \\ \Sigma & \text{if } \iota \text{ is not in any } S_q \\ \bigcap_{q:\iota \in U_q} U_q \cap \bigcap_{q:\iota \in S_q \setminus U_q} S_q & \text{otherwise} \end{cases}$$

In other words, for each image ι in σ , we take all the search results that contained that image and intersect them. We also remove any images the user looked at in that search set,



The cumulative distribution function of entropy loss for ρ from participants in our study.



The number of password images implicitly revealed by participants in our study.

Figure 22: The implicitly-revealed information about ρ. In (b), we assume the attacker steals σ after interacting with the victim. Even in that case, he is not able to learn the victim's entire password.

unless the user overlooked ι, in which case we take the whole result set. We then used the values to bound the entropy loss for each user's σ (see Section 8.9 for details). Note this estimate is conservative because it assumes the phisher knows which search queries contained each image and whether the user overlooked an image in each results set.

Figure 22 presents the cumulative distribution function of entropy loss experienced by participants in our study. No participant revealed more than 62 bits of information about their σ, giving them a residual security of at least 119 bits. Furthermore, more than 80% of participants lost less than 25 bits of σ security.

Implicitly-revealed information about ρ. To estimate the amount of information a phisher can gain about a user's password, we compute as above for each $\iota \in \rho$, and compute an

upper bound on the entropy loss as described in Section 8.9 . Note that this analysis is conservative for the same reasons as before, plus it conservatively assumes the phisher can infer which candidate sets, correspond to images in ρ, and the position of each image in ρ. Figure 22 shows the CDF of bits lost on ρ. On average, participants revealed only 13.8 bits of information about their password. No participant revealed more than 52.6 bits of information (out of 70 bits) about her password, and 90% of our participants revealed less than 30 bits of information. PhorceField provided strong protection for all our participants' passwords.

Implicitly-revealed information about $\rho|\sigma$. Finally, to demonstrate the resiliency of PhorceField, we assume the attacker gains access to σ after conducting the phishing attack. Although a real attacker could attempt to conduct a second phishing attack in this case, the analysis below is intended to show that phishing attacks without σ against PhorceField reveal very little information about user passwords. In this case, we consider an image $\iota \in \rho$ to be completely revealed if the attacker gained any information about it during the phishing attack, i.e. if $C_\iota \neq \Sigma$. Note this assumes the attacker can tell which images are in ρ versus $\sigma \setminus \rho$. Figure 22 shows the distribution of the number of password images that would be revealed in this scenario. Over 75% of the participants revealed fewer than 2 of their password images, and no one revealed all 4 of their password images. If we assume the phisher knows the location of each revealed image within the user's password, then our participants revealed, on average, 3.52 bits of information about $\rho|\sigma$.

Phishing suspicion. The pre-debriefing questionnaire asked participants, "What was your reaction when you received the lost password email? " Of the 23 original participants, only

		Suspected Phishing	
		Yes	No
Familiar with Phishing	Yes	4	**13**
	No	0	4

Table 21: Responses to the questions, "Were you familiar with the concept of phishing before participating in this study?" and "At any point during the study, did you suspect that the study was about phishing?".

5 gave answers that mentioned '''not safe", "protected", "scam", "hoax" or other phrases related to security and phishing. As mentioned above, we ruled out 2 of these 5 participants because they refused to visit the attack page. Therefore, of the 21 participants whose data are reported here, 18 made no indication before the debriefing that they suspected a phishing attack. As a double-check, we asked participants after the debriefing, "At any point during the study, did you suspect that the study was about phishing? ", to which 4 subjects answered "Yes". Of the 21 participants, 13 answered "Yes" to the question, "Were you familiar with the concept of phishing before participating in this study? " Based on these results, we believe that the attack success observed in our study is a good upper bound on the success a phisher would experience with this attack in the real world because participants primarily believed the deception given to them in the email. Table 21 summarizes the results of our post-debriefing questionnaire.

Pre-debriefing σ test. Before reading the debriefing statement, we presented the user with 24 images, including their entire σ, and asked them to recall as many images possible from their graphical password prompt. On average, users were able to recall 3.9 images in their ρ (all but one subject remembered all 4) and only 1.1 images from $\sigma \setminus \rho$. This demonstrates that our participants knew their passwords but were unable to reveal them to our simulated phishing attack.

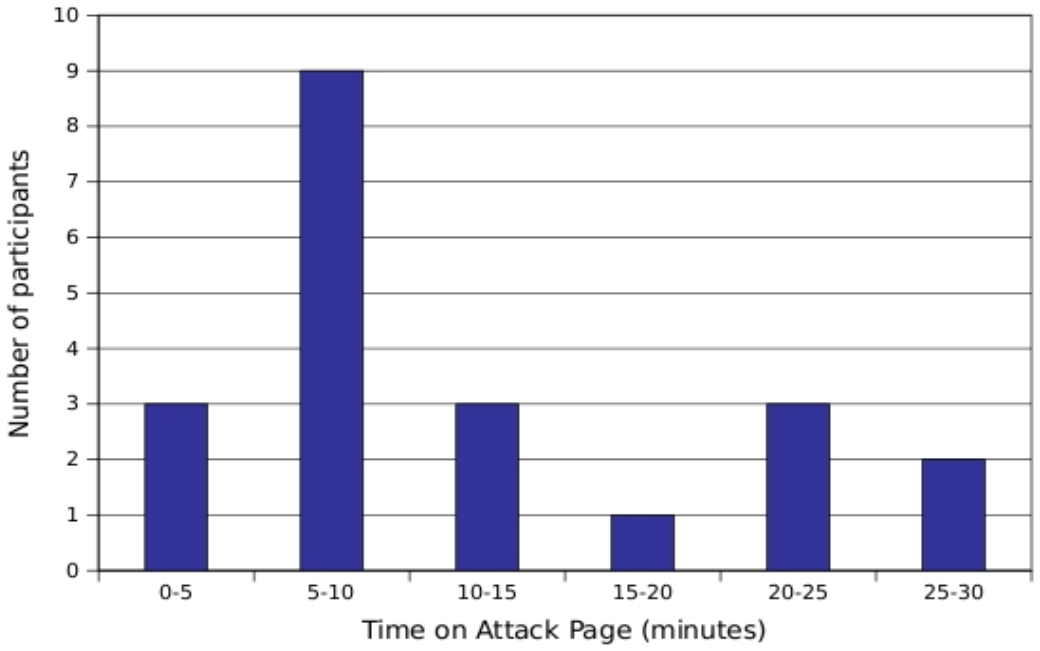


Figure 23: The amount of time participants spent on the phishing page.

Time spent on attack page. Figure 23 shows how long participants spent on the attack page, including revisits. Thirteen participants gave up in under 10 minutes, but five tried for over 20 minutes to find their password. From these data, we conclude that anti-phishing schemes should make users work for at least half an hour to divulge their private information.

Other observations. Before and after reading the debriefing statement, subjects filled out a questionnaire. The questionnaire asked the participants to explain why they left the attack page. Subjects grasped that there were indeed thousands of images (one subject suggested there must be at least 400000) and that it would take them a long time to search them. Subjects mentioned that they were confused by similar images or could not precisely recall the details (e.g. color) in their σ images. Subjects could formulate search queries based on their images (particularly those

in ρ), but the search results were too numerous to find the specific image. Put succinctly by one subject, it was "too much of a hassle".

The median value for the subjects' belief that we had actually lost their password was three on a five point Likert scale, suggesting that some users truly believed us and some were very skeptical. During the course of the study, we received five emails from subjects after closing the attack page concerned about how to continue despite not being able to retrieve their password. Several subjects also mentioned that they did not attempt to "write down" or otherwise save their password.

8.6 Discussion

The results of our user study imply that PhorceField will prevent most password phishing attacks from succeeding. 76% of our participants failed to find even a single image in their password, and no participant found his entire password. Even assuming the phisher later gained access to σ, no participant entirely compromised his password. This compares quite favorably to passive phishing defenses such as SiteKey [204]. Previous studies have shown that 92% of users will reveal their entire text password to a phisher, even if their SiteKey is missing. In our study, 0% of participants revealed their entire password despite the extra information gathered during the attack.

Our results suggest reasonable security parameters for a real-world PhorceField deployment. Our choice of was sufficiently large to make searching for images difficult. Likewise, selecting about a thousand images for each concrete noun made examining search results a tedious task, improving phishing-resistance. Future deployments can increase security

without harming usability by choosing a larger Σ. For example, there are over 100 million Creative Commons licensed photos on Flickr [220], so it would be straightforward to construct a Σ with over 100 million images representing over 10000 concrete nouns. Given that some participants revealed 3 of their password images, $|\rho|$ should be at least 6 and possibly higher. Passwords should not allow repeated images, since this forces users to find the maximum possible number of images during a phishing attack.

Ours is the first study to estimate bounds on how much time and effort users will spend on a phishing page. These measurements should guide researchers designing future phishing defenses. Participants spent an average of 11.4 minutes on the attack page, and one participant spent 26 minutes on the attack page. Participants looked through an average of 3500 images, and one participant looked at 13875 images. Participants entered 6 search terms on average, and one participant conducted 25 searches. Several participants visited the attack page twice, and one participant visited 5 times. Security researchers should create defenses for the most vulnerable users, so they should design for the upper bounds presented here.

Most users gave up after trying to find 1 or 2 images, so their search queries only revealed information about 1.33 images in their σ on average. Phishers could design attacks to avoid this bottleneck, but the success rate may or may not improve. For example, a phisher could present victims with 12 text boxes and ask the victim to describe each of the images in their σ. By forcing victims to describe their entire σ instead of allowing them to focus on one image at a time, the attacker may gain information about more images in σ. However, the information gained about each image would probably be less specific, e.g. the victim might simply enter

"bird" instead of performing several searches such as "bird", "flying", and "flock", which together reveal much more information about the given image. We could render this attack ineffective by using images, such as random art images, that admit no easy description.

PhorceField has two limitations that we should note. First, PhorceField is ineffectual in preventing phishers from soliciting personal information from victims such as Social Security numbers or credit card information. This work focuses exclusively on authentication. We believe that a two prong approach may be useful in defeating this social networking. First, services must try to minimize the collection of these personal identifiers so that the request for them is the exception not the rule. This may assist users in detecting these attacks, but ultimately, it will require user vigilance. Another potential avenue to aide users in detecting malicious intent is for the browsers themselves to identify when this type of information is being input into a form. DLP systems, as discussed in Chapter 6 , do successfully identify these types of Personally Identifiable Information.

The second limitation is that PhorceField does require the installation of client-side secrets. We do not see this as an impediment to deployment since many anti-phishing strategies employ client-side secrets in their defenses including Phoolproof [214] and Dynamic Security Skins [199]. In fact, there are many methods available to install client-side secrets such as Dynamic Security Skins where the secret is installed on the client-side or Phoolproof where the secret is installed onto a device that the user owns and presumably will not loose, e.g., their cellular telephone.

8.7 Related work

PhorceField touches upon several areas of research: secure password prompts, password-based authentication protocols, anti-phishing tools, graphical passwords, and secure machine registration protocols. We now discuss related work that hasn't already been covered in Section 8.1 .

Password-Authenticated Key Exchange protocols. PAKE protocols enable computers to perform mutual authentication and establish a session key based on a password provided by a user [229][205]. PAKE protocols protect passwords once they are entered into a secure password prompt, they do not prevent users from entering passwords into malicious prompts.

Anti-phishing tools. In general, anti-phishing tools identify phishing attempts either in email or when a user loads a web page. Most phishing attempts are instigated by spam emails in which an attacker uses social engineering to encourage the victim to navigate to a phishing page [230]. Much research has been focused on spam detection. The main thrust of this research focuses on statistical methods and machine learning techniques to identify spam messages. There have been successful applications of Bayesian classifiers [210] and support vector machines [231]. Many solutions based on these techniques are widely available [232][233].

Commercial anti-phishing tools use heuristics, community ratings, black-lists and white-lists [234]. Many simplistic phishing attacks can be detected using heuristic evaluation. Criteria for rejection include the geographical origin of the website, the owner, the age of the website and possible URL obfuscation. These programs, usually in the form of tool bars or plug-ins, indicate to the user the level of confidence that the current website is indeed genuine. The SpoofGuard

project [235] takes this one step further by also evaluating the images on the web page. Other commercial products maintain community ratings of websites where votes determine if the site is legitimate or fraudulent [236]. These ratings may also be used to construct black-lists, or dynamically changing lists of sites that contain phishing or malware. White-lists are the opposite: these are genuine websites that do not attempt to steal credentials from users. Some attempts have been made to construct user-generated white-lists [211].

Machine learning has also been employed to detect phishing websites [237]. These techniques use classifiers based on the features of the email or website to determine whether it as a phishing attempt. Several different machine learning classifiers have been used, the most successful being support vector machines and random forests. In addition, researchers have utilized Fuzzy Sets on similar features of websites to determine if they should be included in the set of authentic or phishing sites [238].

Graphical passwords. Suo, et al., provide an extensive review of graphical password systems and group them into three categories [239]. Cognometric graphical password systems are the most prevalent. The Déjà Vu system [203] asks users to select a subset of images from a collection of random art images. Weinshall, et al., have an interface that shows users 100-200 sets of image where the user password consists of selecting a predetermined image from each of the sets [240]. Systems like Passfaces take advantage of users' innate ability to recall faces to improve recall [217]. Some systems use thumbnails generated from a single image rather than utilizing image sets [241]. Shoulder resistant schemes for graphical password systems have been developed as well. One such interface asks users to click within a convex hull formed by the

objects consisting of their passwords [242]. Another shoulder-surfing resistant strategy includes using an eye tracker to determine the sequence of images the user looked upon [243]. Drawmetric graphical password systems require users to reproduce a shared secret to authenticate. One such technique has the user draw a secret on a 2-D grid and reproduce it for authentication. Other similar techniques include presenting a signature provided by either a stylus [244] or mouse [245]. Locimetric graphical passwords have users repeat a sequence of actions. These systems have users click on a series of interesting and meaningful points in an image in a predetermined sequence [246][247]. Builders of these systems argue that a large password space can be constructed from a single image since images can contain hundreds of memorable points.

8.8 Conclusion

In this chapter, we presented PhorceField, a password ceremony designed to depend on human laziness rather than vigilance. PhorceField uses a client-side secret and graphical password scheme to make it effectively impossible for a user to provide her password to a phisher. As long as the phisher does not know the client-side secret, he can only present the user with a non-standard and difficult-to-use password interface. We also designed PhorceField to exploit weaknesses of human memory to make phishing attacks even less likely to succeed.

We conducted a user study to verify that users will be unable to comply with a phisher's requests. No participant in our study successfully entered his password into our phishing web page, even though some participants spent several minutes trying to do so.

Our user study also suggests conservative bounds for the amount of time and effort users will spend cooperating with a phisher. Participants executed several searches and looked through thousands of images in their attempts to help the phisher. Several of our participants visited the phishing page more than once. Researchers should keep these facts in mind when designing future phishing defenses.

8.9 Computing Entropy Loss

Given C_ι for each $\iota \in \sigma$, we wish to compute the entropy loss experienced by each participant in our study or, equivalently, the residual entropy of σ given the C_ι values. If each image in σ were chosen independently, the entropy would be $\sum_{i \in \sigma} |C_\iota| - \log_2 12!$. However, the images in σ were chosen so that each represents a distinct concrete noun, so they are not independent.

procedure enumerate-sigmas($C_1,\ldots,C_k,F,i,(\iota_1,\ldots,\iota_{i-1})$)
if $i = k + 1$
  output $(\iota_1,\ldots,\iota_k)$
for $\iota \in C_i \setminus F$
  enumerate-sigmas($C_1,\ldots,C_k$, $F \cup N(\iota)$, $i+1$, $(\iota_1,\ldots,\iota_{i-1},\iota)$)

Figure 24: Pseudo-code for enumerating all possible values for σ given C_1 , . . . , C_k . In this code, k is the number of images in σ, i.e. 12, and N(ι) is the set of images derived the same concrete noun as ι.

To derive a lower bound on the number of possible values for σ given the C_ι sets, we analyze the pseudo-code in Figure 24 for enumerating all possible σ values. In this code, we order the sets C_ι from smallest to largest and define $N(\iota)$ as the set of images in Σ that represent the same concrete noun as the image ι. In our implementation, $|N(\iota)| \approx 975$ for all ι. Thus, in the

pseudo code presented in Figure 24, $|F| \approx 975i$, so the for-loop in the ith recursive call will execute at least $|C_i|$- 975i times. Thus this code will generate output at least $\prod_{i=1}^{k}(|C_i|-975\,i)$ times. The same σ value may occur more than once, though. However, each σ value can occur at most k! times, giving us the lower bound:

$$\frac{\prod_{i=1}^{k}(|C_i|-975\,i)}{k!}$$

The only wrinkle in this analysis is that, if C_i is small, $|C_i|$- 975i may be zero or negative. However, we know from construction that there is at least one σ consistent with the sets C_i, so in this case we replace $|C_i|$- 975i with 1 in the first few terms:

$$\frac{\prod_{i=k_0+1}^{k}|C_i|-975\,i}{(k-k_0)!}$$

where k_0 is the smallest integer such that $|C_i|$- 975i > 0 for all $i > k_0$. In this case, we can also reduce the degree of symmetry among the solutions to (k - k_0)! since we're only counting solutions that have a common prefix of length k_0.

We count possible values for ρ in the same way, except that order matters in passwords, so we don't divide by (k - k_0)!.

Chapter 9 Conclusion

This dissertation has presented Content-based Access Control (CBAC): a paradigm for access control where access to an object is contingent on the content of objects in the system. CBAC is designed to address to grave concerns with respect to privacy in a networked world: users are either *unable* or *unwilling* to use existing access controls. By integrating content recognition into policy enforcement, designing more usable interfaces and expediting policy acquisition, we are able to make access control easier and more intuitive as well as provide users protection where little was afforded before. We present three different systems that employ CBAC: Privacy/Policy-aware blogging (PLOG), text classification for DLP and Wikipedia vandalism detection.

CBAC is not intended to replace existing access controls but to enhance it. We also cannot guarantee that it will arrive at the correct policy. Therefore, privacy in a CBAC cannot be analyzed in terms of a zero-sum analysis, but rather as a probabilistic assessment of the likelihood of a policy infraction. For example, in Data Loss Prevention, the risk of an accidental disclosure is correlated with the false negative rate. Therefore, for certain situations where any intrusion or infraction cannot be tolerated, CBAC will not be appropriate. But for many domains, such as the enterprise and the Web, a CBAC system can provide a great utility even if it is not perfect.

In addition to our CBAC, we presented PhorceField, a phishing resistant password system that addresses a major issue in existing authentication schemes: the fallibility of user vigilance. A PhorceField-based system imbues passwords with inherent phishing resistance. By making the device and not the user responsible for the password alphabet, users cannot mistakenly input their password into a phishing prompt. Phishers then must engage victims in a laborious task of searching for the images in their password from a set of hundreds of thousands. PhorceField can be deployed in a single or multi-factor authentication scheme and serve as a drop in replacement to many existing systems such as SiteKey or Dynamic Security Skins.

Bibliography

1. Facebook. Timeline. http://www.facebook.com/press/info.php?statistics#!/press/info.php?timeline.

2. Facebook. Statistics. http://www.facebook.com/press/info.php?statistics#!/press/info.php?statistics.

3. Claudine Beaumont. New York plane crash: Twitter breaks the news, again. http://www.telegraph.co.uk/technology/twitter/4269765/New-York-plane-crash-Twitter-breaks-the-news-a.

4. About.com. What is a Tweet?. http://webtrends.about.com/od/glossary/g/what-is-a-tweet.htm.

5. ABC News. Pop Star Justin Bieber Is on the Brink of Superstardom. http://abcnews.go.com/GMA/Weekend/teen-pop-star-justin-bieber-discovered-YouTube/story?id=9068403.

6. Barbara Ortutay. Facebook to end Beacon tracking tool in settlement. http://www.usatoday.com/tech/hotsites/2009-09-21-facebook-beacon_N.htm.

7. Charlotte Observer. Kirsten Valle. http://www.charlotteobserver.com/2008/11/16/354729/bosses-checking-up-on-workers.html.

8. Applicant.com. How To Lose a Job Via Facebook In 140 Characters or Less. http://applicant.com/how-to-lose-a-job-via-facebook-in-140-characaters-or-less/.

9. ESPN. Facebook post gets worker fired. http://sports.espn.go.com/nfl/news/story?id=3965039.

10. Erik Palm. Facebooking while out sick gets employee fired. http://news.cnet.com/8301-1023_3-10228434-93.html.

11. Ellen Simonetti. I was fired for blogging. http://news.com.com/I+was+fired+for+blogging/2010-1030_3-5490836.html.

12. C. Hopkinsurt. Statistics on Fired Bloggers. http://morphemetales.blogspot.com/2006/10/statistics-on-fired-bloggers.html.

13. David G. Savage. Blogger beware: Postings can lead to lawsuits. http://www.latimes.com/news/nationworld/nation/la-na-blogger-suits-20100823,0,5604043.story?track=rs.

14. Russell Goldman. Teens Indicted After Allegedly Taunting Girl Who Hanged Herself. http://abcnews.go.com/Technology/TheLaw/teens-charged-bullying-mass-girl-kill/story?id=10231357.

15. Abby Goodnough and Anahad O'Connor. Details Released About "Craigslist" Suspect. http://www.nytimes.com/2009/04/22/us/22boston.html.

16. Tom O'Reilly. What Is Web 2.0. http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html.

17. D. Gurzick and W. Lutters, From the Personal to the Profound: Understanding the Blog Life Cycle. *CHI '06 extended abstracts on Human factors in computing systems,* 827-832, Montreal, Quebec, 2006, ACM.

18. B. Nardi and D. Schiano and M. Gumbrecht. Blogging as social activity, or, would you let 900 million people read your diary? *Computer Supported Cooperative Work,* 222-231, Chicago, Illinois, 2004, ACM.

19. J. Boase and J. Horrigan and B. Wellman and L. Rainie. The Strength of Internet Ties. http://www.pewinternet.org/pdfs/PIP_Internet_ties.pdf.

20. D. Boyd. Why Youth <3 Social Network Sites. Youth, Identity, and Digital Media. The MIT Press, 2007.

21. wikiHow. wikiHow: the how to manual that you can edit. http://www.wikihow.com/Main-Page.

22. BBC Staff Writer. Wikipedia survives research test. http://news.bbc.co.uk/2/hi/technology/4530930.stm.

23. E. Hansen. Wikipedia Founder Edits Own Bio. http://www.wired.com/culture/lifestyle/news/2005/12/69880.

24. N. Cohen. After False Claim, Wikipedia to Check Degrees. http://www.nytimes.com/2007/03/12/technology/12wiki.html

25. E. Lee. CBS says YouTube partnership a success. http://www.sfgate.com/cgi-bin/blogs/sfgate/detail?blogid=19&entry_id=11166.

26. J. Davis. The Secret World of Lonelygirl. http://www.wired.com/wired/archive/14.12/lonelygirl_pr.html.

27. N. Anderson. YouTube's copyright conundrum. http://arstechnica.com/business/news/2006/10/7982.ars.

28. K. Lorenz. Avoid getting fired for blogging. http://www.cnn.com/2005/US/Careers/04/05/blogging/index.html.

29. C. Negroni, Fired Flight Attendant Finds Blogs Can Backfire. http://www.nytimes.com/2004/11/16/business/16pose.html.

30. K. Poulsen. MySpace Predator Caught by Code. http://www.wired.com/news/technology/0,71948-0.html.

31. D. Rowse. Blog Stalkers - Personal Safety for Bloggers. http://www.problogger.net/archives/2006/02/07/blog-stalkers-personal-safety-for-bloggers/.

32. S. Barnes, Privacy Paradox: Social Networking in the United States. *First Monday*, 11(9), 2006.

33. J. Cheng, Google search by employer not illegal, say judges. http://arstechnica.com/tech-policy/news/2007/05/google-search-by-employer-not-illegal-say-judges.ars.

34. COPPA. Children's Online Privacy Protection Act of 1998. http://www.ftc.gov/ogc/coppa1.htm.

35. DMCA. Digital Millennium Copyright Act. www.copyright.gov/legislation/dmca.pdf.

36. H. Nissenbaum. *Privacy as Contextual Integrity. Washington Law Review*,79(1), 2004.

37. Schneier, B. A Taxonomy of Social Networking Data. *Security Privacy, IEEE*, 8(4), July 2010.

38. Julia Angwin and Tom McGinty. Sites Feed Personal Details To New Tracking Industry. http://online.wsj.com/article/SB10001424052748703977004575393173432219064.html?mod=WSJ_hps_MIDDLETop.

39. Wikipedia. Semi-protected articles. http://en.wikipedia.org/wiki/Wikipedia:Protection_policy.

40. BBC staff writer. MySpace tightens age restrictions. http://news.bbc.co.uk/2/hi/technology/5101942.stm.

41. L. Chenaurel. Facebook's feeds cause privacy concerns. http://halogen.note.amherst.edu/~astudent/2006-2007/issue02/news/01.html.

42. Stuart E. Schechter and Rachna Dhamija and Andy Ozment and Ian Fischer. The Emperor's New Security Indicators. 2007. *Proceedings of the 2007 IEEE Symposium on Security and Privacy,* 51-65, Oakland, CA, IEEE Computer Society.

43. Ellen Nakashima. Harsh Words Die Hard on the Web. http://www.washingtonpost.com/wp-dyn/content/article/2007/03/06/AR2007030602705_pf.html.

44. Oasis. A Brief Introduction to XACML . http://www.oasis-open.org/committees/download.php/2713/Brief_Introduction_to_XACML.html.

45. P. Ashley and S. Hada and G. Karjoth and C. Powers and M. Schunter. Enterprise Privacy Authorization Language (EPAL 1.1). 2003

46. M. Lorch and S. Proctor and R. Lepro and D. Kafura and S. Shah. First Experiences Using XACML for Access Control in Distributed Systems . *ACM workshop on XML security*, 25-37, Fairfax, Virginia, 2003, ACM.

47. M. Verma. XML Security: Ensure portable trust with SAML. http://www.ibm.com/developerworks/xml/library/x-seclay4/.

48. A. Anderson. XACML References and Products. http://www.oasis-open.org/committees/download.php/27298/xacmlRefs-V1-84-1.htm.

49. Tanzia Vega. 'Do Not Track' Privacy Bill Appears in Congress. http://mediadecoder.blogs.nytimes.com/2011/05/06/do-not-track-privacy-bill-appears-in-congress/.

50. R. Wenning and M. Schunter. The Platform for Privacy Preferences 1.1 (P3P1.1). http://www.w3.org/TR/2004/WD-P3P11-20040427/.

51. L. Cranor and M. Arjula and P. Guduru. Use of a P3P User Agent by Early Adopters. *ACM workshop on Privacy in the Electronic Society*, 1-10, Washington, 2002, ACM.

52. Rudiger Grimm and Alexander Rossnagel. Can P3P help to protect privacy worldwide?. *Proceedings of the 2000 ACM workshops on Multimedia*, 157-160, Los Angeles, California, 2000. ACM.

53. JJ. Rousseau. The Social Contract, or Principles of Political Right. http://etext.lib.virginia.edu/toc/modeng/public/RouSoci.html.

54. J. Kaufman and S. Edlund and D. Ford and C. Powers, The Social Contract Core. *Electronic Commerce Research.* Springer Netherlands, 2005.

55. D. Weitzner and J. Hendler and T. Berners-Lee and D. Connolly. Creating a Policy-Aware Web: Discretionary, Rule-based Access for the World Wide Web. *Web and Information Security*, E. Ferrari and B. Thuraisingham (eds), Idea Group Inc., Hershey, PA

56. W3C. Cwm. http://www.w3.org/2000/10/swap/doc/cwm.html.

57. G. Zarri and P. Wu and P. Stockinger and E. Berino and E. Ferrari and A. Perego and P. Abreu and F. Pires and D. Allen and A. Maxwell. EUFORBIA, Advanced Indexing and Filtering of Questionable WebSites, Results and Final Remarks. http://semioweb.msh-paris.fr/euforbia/ http://ec.europa.eu/information_society/activities/sip/docs/pdf/projects/eurofibia_final_report.pdf .

58. Hal Abelson and Lalana Kagal. The Accountability Perspective. *W3C Workshop on Privacy and data usage control*, Cambridge, MA, 2010, W3C.

59. Wikimedia Foundation. Wikipedia. http://en.wikipedia.org.

60. Martin Potthast. Crowdsourcing a wikipedia vandalism corpus. *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, 789-790, Geneva, Switzerland, 2010, ACM.

61. L. Qin. and V. Atluri. Concept-level access control for the semantic web. *Proceedings of the 2003 ACM workshop on XML security,* 94-103, Fairfax, Virginia, 2003, ACM.

62. P. Samarati and E. Bertino and S. Jajodia. An authorization model for a distributed hypertext system. *IEEE Transactions on Knowledge and Data Engineering* , 8(4), 555-562, 1996.

63. E. Bertino and J. Fan and E. Ferrari and M. Hacid and A. Elmagarmid and X Zhu. A hierarchical access control model for video database systems. *ACM Transactions on Information Systems*, 21(2), 151-191, 2003.

64. C Karat. and J Karat. and C Brodie. and J Feng. Evaluating interfaces for privacy policy rule authoring. *Proceedings of the SIGCHI conference on Human Factors in computing systems,* 83-92, Montreal Quebec, 2006, ACM.

65. C. Manning and H. Schutze., Foundations of Statistical Natural Language Processing. MIT Press, 1999.

66. C. Brodie and C. Karat and J. Karat. Intelligible access control: An empirical study of natural language parsing of privacy policy rules using the SPARCLE policy workbench. *Proceedings of SOUPS,* 8-19, Pittsburgh, Pennsylvania, 2006, ACM.

67. Jonathan Karl. WikiLeaks Releases Confidential Diplomat Cables. http://abcnews.go.com/Politics/wikileaks-releases-classified-diplomat-cables-us-state-department/story?id=12260376.

68. moveon.org . Petition: Facebook, stop invading my privacy!. http://www.facebook.com/group.php?gid=5930262681.

69. Rob Pegoraro. Facebook Backs Into a 'Bill of Rights'. http://www.washingtonpost.com/wp-dyn/content/article/2009/02/18/AR2009021803121.html.

70. Jon Kleinberg and Éva Tardos. Algorithm Design. Addison Wesley. 2005.

71. Automattic. Wordpress. http://wordpress.org.

72. Mark Brownlow. Email and webmail statistics. http://www.email-marketing-reports.com/metrics/email-statistics.htm.

73. Debbie Stone and Caroline Jarrett and Mark Woodroffe and Shailey Minocha. User Interface Design and Evaluation (The Morgan Kaufmann Series in Interactive Technologies). Morgan Kaufmann, 2005.

74. D Ferraiolo. and R Kuhn. Role-based access controls. *Proceedings of the 15th National Computer Security Conference,* 554-563, 1992, Gaithersburg, Md.

75. William H. Winsborough and Ninghui Li. Towards Practical Automated Trust Negotiation. *Proceedings of the Third International Workshop on Policies for Distributed Systems and Networks (Policy 2002)*, 92-103, 2002, IEEE Computer Society.

76. Wolfgang Nejdl and Daniel Olmedilla and Marianne Winslett. PeerTrust: automated trust negotiation for peers on the semantic web. *In Workshop on Secure Data Management in a Connected World (SDM 2004)*, 118-132, 2004.

77. Robert W. Reeder and Lujo Bauer and Lorrie Faith Cranor and Michael K. Reiter and Kelli Bacon and Keisha How and Heather Strong. Expandable grids for visualizing and authoring computer security policies. *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems,* 1473-1482, Florence Italy, 2008, ACM.

78. Lujo Bauer and Lorrie Faith Cranor and Robert W. Reeder and Michael K. Reiter and Kami Vaniea. A user study of policy creation in a flexible access-control system. *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems* 543-552, Florence Italy, 2008, ACM.

79. Martin Fowler. Domain Specific Languages. Addison-Welesly Professional. 2010

80. Henry A. Rowley and Yushi Jing and Shumeet Baluja. Large Scale Image-Based Adult Content Filtering. *International Conference on Computer Vision Theory and Applications,* 290-296, 2006.

81. E. Bertino and E. Ferrari and E. Perego, Content-based filtering of Web documents: The MaX system and the EUFORBIA project. *International Journal of Information Security*, 2(2), 45-48, 2003.

82. James Ze Wang and Gio Wiederhold and Oscar Firschein. System for Screening Objectionable Images. *Computer Communications*, 21(15), 1355-1360, 1998.

83. Travis D. Breaux and Annie I. Anton. Mining rule semantics to understand legislative compliance. *WPES '05: Proceedings of the 2005 ACM workshop on Privacy in the electronic society,* 51-54, 2005, Alexandria, Virginia, ACM.

84. G. Yee and L. Korba, Semi-Automatic Derivation and Use of Personal Privacy Policies in E-Business. *International Journal of E-Business Research*, 1(1), 54-69, 2005.

85. W3C. Resource Description Framework. http://www.w3.org/RDF/.

86. Paul Heymann and Daniel Ramage and Hector Garcia-Molina. Social tag prediction. *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information,* 531-538, New York, NY, 2008, ACM.

87. Christopher H. Brooks and Nancy Montanez. Improved annotation of the blogosphere via autotagging and hierarchical clustering. *WWW '06: Proceedings of the 15th international conference on World Wide Web*, 625-632, New York, NY, USA, 2006, ACM.

88. Money Magazine. Get Rich Slowly. http://www.getrichslowly.org/.

89. Gilad Mishne. AutoTag: a collaborative approach to automated tag assignment for weblog posts. *WWW '06: Proceedings of the 15th international conference on World Wide Web*, 953-954, New York, NY, USA, 2006, ACM.

90. Sanjay Sood and Sara Owsley and Kristian Hammond and Larry Birnbaum. TagAssist: Automatic Tag Suggestion for Blog Posts. *Proceedings of the International Conference on Weblogs and Social Media (ICWSM 2007)*, 2007.

91. Automattic. Growing Blogs -- Wordpress.com. http://botd.wordpress.com/growing-blogs.

92. Apache Software Foundation. Apache Lucene. http://lucene.apache.org.

93. Ciro Cattuto and Vittorio Loreto and Luciano Pietronero. Collaborative Tagging and Semiotic Dynamics. *PNAS*, 104(5), 1461-1464, 2007.

94. Ian H. Witten and Eibe Frank, Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann, 2005.

95. K. Burton and A. Java and I. Soboroff. The ICWSM 2009 Spinn3r Dataset. *Proceedings of the Third Annual Conference on Weblogs and Social Media (ICWSM 2009)*, 2009.

96. Poneman Institute. Fourth Annual US Cost of Data Breach Study. http://www.ponemon.org/local/upload/fckjail/generalcontent/18/file/2008-2009%20US%20Cost%20of%20Data%20Breach%20Report%20Final.pdf.

97. Privacy Rights Clearinghouse. Chronology of Data Breaches: Security Breaches 2005--present. http://www.privacyrights.org/data-breach.

98. McAfee. Data Loss Prevention. http://www.mcafee.com/us/enterprise/products/data_loss_prevention/.

99. Symantec. Data Loss Prevention Products & Services. http://www.symantec.com/business/theme.jsp?themeid=vontu.

100. Trend Micro. Trend Micro Data Loss Prevention. http://us.trendmicro.com/us/products/enterprise/data-loss-prevention/.

101. RSA. Data Loss Prevention. http://www.rsa.com/node.aspx?id=1130.

102. proofpoint. Unified Email Security, Email Archiving, Data Loss Prevention and Encryption. http://www.proofpoint.com/id/outbound/index.php.

103. Frederick Hitz, *Why Spy?: Espionage in an Age of Uncertainty.* Thomas Dunne Books, 2008

104. T. Joachims. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. *European Conference on Machine Learning (ECML)*, 137-142, 1998, Berlin, Springer.

105. Trustwave. Global Security Report 2010. https://www.trustwave.com/whitePapers.php.

106. Christopher D. Manning and Prabhakar Raghavan and Hinrich Schutze. Introduction to Information Retrieval. Cambridge University Press, 2008.

107. Mark Hall and Eibe Frank and Geoffrey Holmes and Bernhard Pfahringer and Peter Reutemann and Ian H. Witten, The WEKA data mining software: an update. *SIGKDD Explorations Newsletter,* 11(1), 10-18, 2009.

108. Lewis, David. Reuters 21578. http://www.daviddlewis.com/resources/testcollections/rcv1/.

109. Trevor Hastie and Robert Tibshirani. Classification by pairwise coupling. *Proceedings of the 1997 conference on Advances in neural information processing systems 10*, 507-513, Cambridge, MA, USA, 1998, MIT Press.

110. Hearst, Marti. Teaching applied natural language processing: triumphs and tribulations. *TeachNLP '05: Proceedings of the Second ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language*, 1-8, Morristown, NJ, USA, 2005, ACM.

111. Internet Archive. WayBack Machine. http://www.archive.org.

112. Internet Archive. Brown Corpus. http://www.archive.org/details/BrownCorpus.

113. Proofpoint. Outbound Email Security and Data Loss Prevention. http://www.proofpoint.com/id/outbound/index.php.

114. M. E. Maron. Automatic Indexing: An Experimental Inquiry. *Journal of ACM*, 8(3), 404-417, 1961.

115. Harold Borko and Myrna Bernick. Automatic Document Classification. *Journal of ACM*, 10(2), 151-162, 1963.

116. Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys.* 34(1), 1-47, 2002.

117. N. Fuhr and G. E Knorz. Retrieval test evaluation of a rule based automatic indexing (AIR/PHYS). *Proc. of the third joint BCS and ACM symposium on Research and development in information retrieval*, 391-408, New York, NY, USA, 1984, Cambridge University Press.

118. Philip J. Hayes and Steven P. Weinstein. CONSTRUE/TIS: A System for Content-Based Indexing of a Database of News Stories. *IAAI '90: Proceedings of the The Second Conference on Innovative Applications of Artificial Intelligence*, 49-64, 1991, AAAI Press.

119. William W. Cohen. Learning Rules that Classify E-Mail. *In Papers from the AAAI Spring Symposium on Machine Learning in Information Access*, 18-25, 1996, AAAI Press.

120. Ion Androutsopoulos and John Koutsias and Konstantinos V. Chandrinos and Constantine D. Spyropoulos. An experimental comparison of naive Bayesian and keyword-based anti-spam filtering with personal e-mail messages. *SIGIR '00: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information*, 160-167, New York, NY, USA, 2000, ACM.

121. Giuseppe Attardi and Antonio Gullì and Fabrizio Sebastiani. Automatic Web Page Categorization by Link and Context Analysis. *Proceedings of THAI-99, 1st European Symposium on Telematics, Hypermedia and Artificial Intelligence*, Varese, 1999.

122. Soumen Chakrabarti and Byron Dom and Rakesh Agrawal and Prabhakar Raghavan. Scalable feature selection, classification and signature generation for organizing large text databases into hierarchical topic taxonomy. *The VLDB Journal* , 7(3), 163-178, 1998.

123. Tun Thura Thet and Jin-Cheon Na and Christopher S. G. Khoo. Filtering product reviews from web search results. *DocEng '07: Proceedings of the 2007 ACM symposium on Document engineering*, 196-198, New York, NY, 2007, ACM.

124. D. Koller and U. Lerner and D. Angelov. A general algorithm for approximate inference and its application to hybrid Bayes nets. *Proceedings of the Fifteenth Annual Conference on Uncertainty in Artificial Intelligence*, 324-333, 1999.

125. L. Breiman and J. Friedman and R. Olshen and C. Stone. *Classification and Regression Trees.* Wadsworth,1984

126. T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory* ,13(1), 21-27, 1967.

127. Kenney Ng, A Comparative Study of the Practical Characteristics of Neural Network and Conventional Pattern Classifiers. *Proceedings of the 1990 conference on Advances in neural*

information processing systems 3, 970-976, Denver Colorado, 1990, Morgan Kaufmann Publishers Inc.

128. David Freedman. Statistical Models: Theory and Practice. Cambridge University Press, 2005.

129. T. Joachims. Learning to Classify Text Using Support Vector Machines -- Methods, Theory, and Algorithms. Kluwer/Springer, 2002.

130. Robert E. Schapire. Theoretical Views of Boosting and Applications. *ALT '99: Proceedings of the 10th International Conference on Algorithmic Learning Theory*, 13-25, London, UK, 1999, Springer-Verlag.

131. Corinna Cortes and Vladimir Vapnik. Support-Vector Networks. *Machine Learning,* 20(3), 273-297, 1995.

132. Thorsten Joachims. Transductive Inference for Text Classification using Support Vector Machines. *International Conference on Machine Learning (ICML)*, 200-209, Bled, Slowenien, 1999, Morgan Kaufmann Publishers Inc.

133. Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. *COLT' 98: Proceedings of the eleventh annual conference on Computational learning theory*, 92-100, New York, NY, USA, 1998, ACM.

134. Kristina Toutanova and Frannie Chen and Krist Popat and Thomas Hofmann. Text classification in a hierarchical mixture model for small training sets. *CIKM '01: Proceedings of the tenth international conference on Information and knowledge management*, 105-113, New York, NY, USA, 2001, ACM.

135. Nitesh Chawla and Nathalie Japkowicz and Aleksander Kolcz. Editorial: Special Issue on Learning from Imbalanced Data Sets. *SIGKDD Explorer Newsletter*, 6(1), 1-6, 2004.

136. Nitesh Chawla and Kevin W. Bowyer and Lawrence O. Hall and W. Philip Kegelmeyer. SMOTE: synthetic minority over-sampling technique. *Journal of Artificial Inteligence Research*, 16(1), 321-357, 2002.

137. Nathalie Japkowicz. Supervised versus unsupervised binary-learning by feedforward neural networks. *Machine Learning*, 42(1-2), 97-122, 2001.

138. Dunja Mladenic and Marko Grobelnik. Feature Selection for Unbalanced Class Distribution and Naive Bayes. *Proceedings of the Sixteenth International Conference on Machine Learning*, 258-267, San Francisco, CA, USA, 1999, Morgan Kaufmann Publishers.

139. Evgeniy Gabrilovich and Shaul Markovitch. Computing semantic relatedness using Wikipedia-based explicit semantic analysis. *Proceedings of the 20th international joint conference on Artifical intelligence*, 1606-1611, San Francisco, CA, USA, 2007, Morgan Kaufmann.

140. Xuan-Hieu Phan and Le-Minh Nguyen and Susumu Horiguchi. Learning to classify short and sparse text & web with hidden topics from large-scale data collections. *Proceeding of the 17th international conference on World Wide Web*, 91-100, New York, NY, USA, 2008, ACM.

141. Jian Hu and Lujun Fang and Yang Cao and Hua-Jun Zeng and Hua Li and Qiang Yang and Zheng Chen. Enhancing text clustering by leveraging Wikipedia semantics, *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, 179-186, 2008, ACM.

142. Zsolt Minier and Zalan Bodo and Lehel Csato. Wikipedia-Based Kernels for Text Categorization. *Proceedings of the Ninth International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, 157-164, Washington, DC, USA, 2007, IEEE Computer Society.

143. Michael Strube and Simone Paolo Ponzetto. WikiRelate! computing semantic relatedness using wikipedia. P*roceedings of the 21st national conference on Artificial intelligence - Volume 2*, 1419-1424, Boston, Massachusets, 2006, AAAI Press.

144. Alexa. Top Sites. http://www.alexa.com/topsites/global.

145. Wikipedia. Daily edit statistics. http://stats.wikimedia.org/EN/PlotsPngDatabaseEdits.htm.

146. R. Stuart Geiger and David Ribes. The work of sustaining order in wikipedia: the banning of a vandal. *Proceedings of the 2010 ACM conference on Computer supported cooperative work*, 117-126, Savannah, Georgia, 2010, ACM.

147. Martin Potthast and Benno Stein and Robert Gerling. Automatic vandalism detection in Wikipedia. *ECIR'08: Proceedings of the IR research, 30th European conference on Advances in information retrieval*, 663-668, Berlin, Heidelberg, 2008, Springer-Verlag.

148. Andrew G. West and Sampath Kannan and Insup Lee. Detecting Wikipedia vandalism via spatio-temporal analysis of revision metadata. *EUROSEC '10: Proceedings of the Third European Workshop on System Security*, 22-28, Paris, France, 2010, ACM.

149. William Yang Wang and Kathleen R. McKeown. "Got You!": Automatic Vandalism Detection in Wikipedia with Web-based Shallow Syntactic-Semantic Modeling. *23rd International Conference on Computational Linguistics (Coling 2010)*, 1146-1154, Beijing, China, 2010, Association for Computational Linguistics.

150. Si-Chi Chin and W. Nick Street and Padmini Srinivasan and David Eichmann. Detecting Wikipedia vandalism with active learning and statistical language models. *WICOW '10: Proceedings of the 4rd Workshop on Information Credibility on the Web*, 3-10, Raleigh, North Carolina, USA, 2010, ACM.

151. B. Thomas Adler and Luca de Alfaro and Santiago M. Mola-Velasco and Paolo Rosso and Andrew G. West. Wikipedia Vandalism Detection: Combining Natural Language, Metadata, and Reputation Features. *CICLing '11: Proceedings of the 12th International Conference on Intelligent Text Processing and Computational Linguist,* 277-288, Tokyo, Japan, 2011, Springer.

152. K. Smets and B. Goethals and B. Verdonk. Automatic Vandalism Detection in Wikipedia: Towards a Machine Learning Approach. *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI) Workshop on Wikipedia and Artificial Intelligence: An Evolving Synergy (WikiAI08),* 43-48, 2008, AAAI Press.

153. PAN Workshop 2010. Evaluation Campaign on Plagiarism Detection and Wikipedia Vandalism Detection. http://pan.webis.de/.

154. Martin Potthast. Crowdsourcing a Wikipedia Vandalism Corpus. *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, 789-790, Geneva, Switzerland, 2010, ACM.

155. Alias-i. LingPipe 4.0.0. http://alias-i.com/LingPipe/.

156. Incava.org. Difference algorithm for Java. http://incava.org/projects/1042574828.

157. SoftCorporation LLC. English Dictionary. http://softcorporation.com/products/spellcheck/.

158. ExpressionEngine. English obscene wordlist. http://expressionengine.com/?ACT=51&fid=8&aid=1830_mvLZ2WkoRucNvRegThbL&board_id=1.

159. Dennis Grinberg and John Lafferty and Daniel Sleator. A Robust Parsing Algorithm for Link Grammars. *In Proceedings of the Fourth International Workshop on Parsing Technologies*, Prague, 1995.

160. Cheezburger, Inc. Internet Meme Database. http://www.knowyourmeme.com/.

161. Christopher Poole. 4chan website. http://www.4chan.org.

162. YouTube, Inc. YouTube website. http://www.YouTube.com.

163. MySpace, Inc. MySpace website. http://www.myspace.com.

164. Twitter, Inc. Twitter. http://www.twitter.com.

165. Sean Michaels. Taking the Rick. http://www.guardian.co.uk/music/2008/mar/19/news.

166. Twitter. Trending Topics. http://twitter.com/trendingtopics.

167. Efstathios Stamatatos, A survey of modern authorship attribution methods. *Journal of the American Society for Information Science and Technology.* 60(3), 538-556, 2009.

168. Shlomo Argamon and Moshe Koppel and James W. Pennebaker and Jonathan Schler. Automatically profiling the author of an anonymous text. *Communications of the ACM,* 52(2), 119-123, 2009.

169. Polina Panicheva and John Cardiff and Paolo Rosso. Personal Sense and Idiolect: Combining Authorship Attribution and Opinion Analysis. *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10),* Valletta, Malta, 2010.

170. M. Teresa Turell, The use of textual, grammatical and sociolinguistic evidence in forensic text comparison. *International Journal of Speech Language and the Law*, 17(2), 211-250, 2010.

171. Sindhu Raghavan and Adriana Kovashka and Raymond Mooney. Authorship Attribution Using Probabilistic Context-Free Grammars. *Proceedings of the ACL*, 38-42, Uppsala, Sweden, 2010, ACL.

172. Wikipedia. Main categories in Wikipedia. http://en.wikipedia.org/wiki/Category:Main_topic_classifications.

173. Wikipedia. Database download. http://en.wikipedia.org/wiki/Wikipedia:Database_download.

174. J. B. MacQueen. Some Methods for Classification and Analysis of MultiVariate Observations. *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, 281-297, Berkeley, CA, 1967, University of California Press.

175. Tin Kam Ho, Stop Word Location and Identification for Adaptive Text Recognition. *International Journal on Document Analysis and Recognition*, 3(1), 16-26, 2000.

176. R. Arun and V. Suresh and C.E. Veni Madhavan. Stopword Graphs and Authorship Attribution in Text Corpora. *IEEE International Conference on Semantic Computing*, 192-196, Berkley, CA, 2009, IEEE.

177. Suresh Venkatasubramanian and Ashok Veilumuthu and Avanthi Krishnamurthy and Veni Madhavan and Kaushik Nath and Sunil Arvindam. A Non-Syntactic Approach for Text Sentiment Classification with Stopwords. *Proceedings of the 20th international conference companion on World wide web,* 137-138, Hyderabad, India, 2011, ACM.

178. WEKA. Cost Sensitive Classifier. http://weka.sourceforge.net/doc/weka/classifiers/meta/CostSensitiveClassifier.html.

179. Wikimedia Foundation. Mission Statement. http://wikimediafoundation.org/wiki/Mission_statement

180. Devin Pelcher. Medication Information in Wikipedia is Accessible, but is it Readable?. http://www.slideshare.net/DPelcher/medication-information-in-wikipedia-is-accessible-but-is-it-readable-3171639.

181. Larry Ogrodnek. Fathom Java Library. http://www.representqueens.com/fathom/.

182. Wikipedia. Lists of common misspellings. http://en.wikipedia.org/wiki/Wikipedia:Lists_of_common_misspellings.

183. Oxford University Press. Common spelling errors. http://www.oup.com/uk/booksites/content/0199296251/essentials/commonspellingerrors/.

184. Manoj Harpalani and Thanadit Phumprao and Megha Bassi and Michael Hart and Rob Johnson. *The 4th International Workshop on Uncovering Plagiarism, Authorship, and Social Software Misuse (PAN-10)*, Padua, Italy, 2010.

185. Manoj Harpalani and Michael Hart and Sandesh Singh and Yejin Choi and Rob Johnson, Language of Vandalism: Improving Wikipedia Vandalism Detection via Stylometric Analysis. *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (to appear),* Portland, Oregon, 2011.

186. Wikipedia. Help:Formatting. http://www.mediawiki.org/wiki/Help:Formatting.

187. axelclk. gwtwiki - Java Wikipedia API (Bliki engine). http://code.google.com/p/gwtwiki/.

188. Neil Fraser. Google diff-patch-match. http://code.google.com/p/google-diff-match-patch/.

189. S.M. Mola Velasco. Wikipedia Vandalism Detection Through Machine Learning: Feature Review and New Proposals. *Notebook Papers of CLEF 2010 Labs and Workshops,* 22-23, Padua, Italy, 2010.

190. Petra Perner and Sascha Trautzsch. Multi-interval discretization methods for decision tree learning. *Advances in Pattern Recognition*, 475-482, 1998, Springer Berlin / Heidelberg.

191. Jorge M. Lobo and Alberto Jiménez-Valverde and Raimundo Real. AUC: a misleading measure of the performance of predictive distribution models. *Global Ecology and Biogeography*, 17(2), 145-151, 2008.

192. Cobi Carter. ClueBot. http://en.wikipedia.org/wiki/User:ClueBot.

193. B. Thomas Adler and Luca de Alfaro. A content-driven reputation system for the Wikipedia. *Proceedings of the 16th international conference on World Wide Web*, 261-270, New York, NY, USA, 2007, ACM.

194. IMDb. Lincoln (2012). http://www.imdb.com/title/tt0443272/.

195. Jason Milletary. Technical Trends in Phishing Attacks. www.us-cert.gov/reading_room/phishing_trends0511.pdf.

196. Rick Hodgin. Phishing Cost The U.S. $3.2 Billion In 2007. http://www.tgdaily.com/trendwatch-features/35326-phishing-cost-the-us-32-billion-in-2007.

197. Rachna Dhamija and J. D. Tygar and Marti Hearst. *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems,* 581-590, New York, NY, USA, 2006, ACM.

198. Microsoft. How Interactive Logon Works. http://technet.microsoft.com/en-us/library/cc780332.aspx.

199. Rachna Dhamija and J. D Tygar. The battle against phishing: Dynamic Security Skins. *SOUPS '05: Proceedings of the 2005 symposium on Usable privacy and security*, 77-88, 2005, ACM.

200. Chris Karlof and J.D. Tygar and David Wagner. Conditioned-safe Ceremonies and a User Study of an Application to Web Authentication. *Sixteenth Annual Network and Distributed Systems Security Symposium (NDSS 2009),* 2009.

201. M. C. Anderson and J. H. Neely. *Memory. Handbook of Perception and Cognition.* Academic Press, 1996

202. Kenneth A. Deffenbacher and Thomas H. Carr and John R. Leu. Memory for Words, Pictures, and Faces: Retroactive Interference, Forgetting, and Reminiscence. *Journal of Experimental Psychology: Human Learning and Memory*,7(4), 299-305, 1981.

203. Rachna Dhamija and Adrian Perrig. Déjà Vu: a user study using images for authentication. *SSYM'00: Proceedings of the 9th conference on USENIX Security Symposium*, 4-4, Berkeley, CA, USA, 2000, USENIX Association.

204. Bank of America. SiteKey at Bank of America. http://www.bankofamerica.com/privacy/sitekey/.

205. Mihir Bellare and David Pointcheval and Phillip Rogaway, *Authenticated Key Exchange Secure against Dictionary Attacks.* Springer, 2000.

206. Sonia Chiasson and P. C. van Oorschot and Robert Biddle. A usability study and critique of two password managers. *USENIX Security '06: Proceedings of the 15th conference on USENIX Security Symposium*, 1-15, Berkeley, CA, USA, 2006, USENIX Association.

207. Krebs, Brian. Citibank Phish Spoofs 2-Factor Authentication. http://voices.washingtonpost.com/securityfix/2006/07/citibank_phish_spoofs_2factor_1.html.

208. Verdasys, Inc. SiteTrust Adaptive Authentication. http://www.sitetrust.net/technology/adaptiveauth.aspx.

209. Netcraft. Anti-Phishing Toolbar. http://toolbar.netcraft.com/.

210. Mehran Sahami and Susan Dumais and David Heckerman and Eric Horvitz. A Bayesian Approach to Filtering Junk E-Mail. *Learning for Text Categorization: Papers from the 1998 Workshop*, AAAI Technical Report, 1998.

211. Yue Wang and Agrawal, R. and Baek-Young Choi. Light Weight Anti-Phishing with User Whitelisting in a Web Browser. *Proceedings of the 2008 IEEE Region 5 Conference*, 1-4, Kansas City, MO, 2008, IEEE.

212. Tyler Moore and Richard Clayton. An empirical analysis of the current state of phishing attack and defence. *In Proceedings of the 2007 Workshop on the Economics of Information Security (WEIS)*, 7-8, 2007, Pittsburgh PA.

213. Stuart Schechter and A. J. Bernheim Brush and Serge Egelman. It's No Secret. Measuring the Security and Reliability of Authentication via "Secret" Questions. *SP '09: Proceedings of the 2009 30th IEEE Symposium on Security and Privacy*, 375-390, Washington, DC, USA, 2009, IEEE.

214. Bryan Parno and Cynthia Kuo and Adrian Perrig. Phoolproof Phishing Prevention. *Financial Cryptography and Data Security*, 1-19, 2006.

215. Yevgeniy Dodis and Jonathan Katz and Leonid Reyzin. Robust fuzzy extractors and authenticated key agreement from close secrets. *In Advances in Cryptology - CRYPTO 2006*, 232-250, 2006, Springer.

216. Andrej Bauer. random art. http://www.random-art.org/.

217. Real User Corporation. The Science Behind Passfaces. http://www.realuser.com/published/The%20Science%20Behind%20Passfaces.pdf

218. Darren Davis and Fabian Monrose and Michael Reiter. On User Choice in Graphical Password Schemes. *Proceedings of the 13th USENIX Security Conference,* 151–164, 2004.

219. Yahoo!. Welcome to Flickr. www.flickr.com.

220. Michelle Thorn. Analysis of 100M CC-Licensed Images on Flickr. https://creativecommons.org/weblog/entry/13588.

221. Claude Shannon, Prediction and Entropy of Printed English. *Bell Systems Technical Journal*, 30(1), 50-64, 1951.

222. Amazon Mechanical Turk. Amazon Mechanical Turk - Welcome. https://www.mturk.com/mturk/welcome?variant=worker.

223. Moncur, Wendy and Grégory Leplâtre. Pictures at the ATM: exploring the usability of multiple graphical passwords. *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, 887-894, New York, NY, USA, 2007, ACM.

224. Ian Jermyn and Alain Mayer and Fabian Monrose and Michael K. Reiter and Aviel D Rubin. The design and analysis of graphical passwords. *SSYM'99: Proceedings of the 8th conference on USENIX Security Symposium*, 1-14, Berkeley, CA, USA, 1999, USENIX.

225. Susan Wiedenbeck and Jim Waters and Jean-Camille Birget and Alex Brodskiy and Nasir Memon. PassPoints: design and longitudinal evaluation of a graphical password system. *International Journal of Human-Computer Studies*, 63(1-2), 102-127, 2005.

226. Tomas Lindberg and Risto Nasanen. The effect of icon spacing and size on the speed of icon processing in the human visual system. *Displays*, 24(3), 111-120, 2003.

227. Google Inc. Google Images. http://www.google.com/imghp.

228. John Eng. Sample size estimation: how many individuals should be studied? *Radiology*, 227(3), 309-313, 2003.

229. Thomas Wu. The Secure Remote Password Protocol. *NDSS 1998: Proceedings of the 1998 Internet Society Network and Distributed System Security Symposium*, 97-111, San Diego, 1998.

230. Tom N. Jagatic and Nathaniel A. Johnson and Markus Jakobsson and Filippo Menczer. Social phishing. *Communications, ACM*, 50(10), 94-100, 2007.

231. H. Drucker and Donghui Wu and V.N. Vapnik., Support vector machines for spam categorization. *IEEE Transactions on Neural Networks*, 10(5), 1048-1054, 1999.

232. Burton Computer Corporation. SpamProbe. http://spamprobe.sourceforge.net/.

233. Apache. SpamAssasin. http://spamassassin.apache.org/.

234. Yue Zhang and Serge Egelman and Lorrie Cranor and Jason Hong. Phinding Phish: Evaluating Anti-Phishing Tools. *NDSS '07: Proceedings of the 14th Annual Network and Distributed System Security Symposium,* 2007.

235. Neil Chou and Robert Ledesma and Yuka Teraguchi and Dan Boneh and John C Mitchell. Client-Side Defense against Web-Based Identity Theft. *NDSS '04: Proceedings of the 11th Annual Network and Distributed System Security Symposium,* 2004.

236. GeoTrust Inc. TrustWatch Search By GeoTrust. https://addons.mozilla.org/en-us/firefox/addon/trustwatch-search-by-geotrust/.

237. Saeed Abu-Nimeh and Dario Nappa and Xinlei Wang and Suku Nair. A comparison of machine learning techniques for phishing detection. *eCrime '07: Proceedings of the anti-phishing working groups 2nd annual eCrime researchers summit*, 60-69, 2007, ACM.

238. M. Aburrous and M.A. Hossain and F. Thabatah and K Dahal. Intelligent Phishing Website Detection System using Fuzzy Techniques. *Information and Communication Technologies: From Theory to Applications, 2008*, 1-6, Damascus, 2008.

239. Xiaoyuan Suo and Ying Zhu and G. Scott Owen. Graphical Passwords: A Survey. *ACSAC '05: Proceedings of the 21st Annual Computer Security Applications Conference*, 463-472, Washington, DC, USA, 2005, IEEE Computer Society.

240. Daphna Weinshall and Scott Kirkpatrick. Passwords you'll never forget, but can't recall. *CHI '04 extended abstracts on Human factors in computing systems*, 1399-1402, Vienna, Austria, 2004, ACM.

241. Wayne Jansen. Authenticating Mobile Device Users Through Image Selection. http://csrc.nist.gov/groups/SNS/mobile_security/documents/mobile_devices/PP-VisualAuthentication-rev-DS04.pdf.

242. Leonardo Sobrado and Jean-Camille Birget. Graphical Passwords. *The Rutgers Scholar, An Electronic Bulletin of Undergraduate Research,* 2002.

243. Manu Kumar and Tal Garfinkel and Dan Boneh and Terry Winograd. Reducing shoulder-surfing by using gaze-based password entry. *SOUPS '07: Proceedings of the 3rd symposium on Usable privacy and security*, 13-19, 2007, ACM.

244. Joseph Goldberg and Jennifer Hagman and Vibha Sazawal. Doodling our way to better authentication. *CHI '02 extended abstracts on Human factors in computing systems*, 868-869, Minneapolis, Minnesota, 2002, ACM.

245. Ian Jermyn and Alain Mayer and Fabian Monrose and Michael K. Reiter and Aviel D Rubin. The design and analysis of graphical passwords. *SSYM'99: Proceedings of the 8th conference on USENIX Security Symposium*, 1-14, Berkeley, CA, USA, 1999, USENIX Association.

246. L.D. Paulson. Taking a graphical approach to the password. *Computer*, 35(7), 19-19, 2002.

247. Susan Wiedenbeck and Jim Waters and Jean-Camille Birget and Alex Brodskiy and Nasir Memon. Authentication using graphical passwords: Basic Results. *Human-Computer Interaction International (HCII 2005)*, 1-12, 2005, Springer.