NTNU

Norwegian University of
Science and Technology

Identity Management with Petname Systems

Md. Sadek Ferdous

Master in Security and Mobile Computing

Submission date: June 2009

Supervisor: Øivind Kure, ITEM

Co-supervisor: Audun Jøsang, University of Oslo/UNIK
Peeter Laud, Institute of Computer Science, University of Tartu

Norwegian University of Science and Technology
Department of Telematics

Problem Description

Phishing, one type of social engineering attacks, poses a great security threat to users. Phishing is commonly used to obtain sensitive and secure information (username, password) which might include financial data too, like credit card numbers, by exploiting the trust relationship between a user and a web site. The most used method for Phishing is to masquerade as a web site which the user trusts. In 2007 phishing caused a loss of US $3.2 billion with assumption that the financial loss might escalate in the upcoming years. This large number of phishing attacks has been successful mainly because of users' inability to differentiate between the genuine, trusted and a fake web site, that is, because of mistaken identity. Petname systems have been proposed to offer better identity management. A Petname system consists of Keys which can be used to uniquely identify a web site on a global scale, Nicknames which, supplied by the web site or Identity Provider, represent a global but possibly ambiguous (i.e. non-unique) association with a web site, and Petnames which are created by the user to uniquely refer to that web entity and to provide a bi-directional mapping with keys within the personal user domain. These three fundamental elements of a Petname system represent the three corners (Secure, Memorable and Global) of Zooko's triangle and provide a solid basis for identity management. This project aims at studying ways by which Petname systems can be applied in various applications related to identity management. It also explores the usability issues of applying Petname system in identity management.

Assignment given: 20. January 2009
Supervisor: Øivind Kure, ITEM

Acknowledgements

This dissertation has been written as a part of fulfillment of ERASMUS MUNDUS MSc in Security and Mobile Computing (NordSecMob) and carried out at the University Graduate Center in Kjeller (UniK) in between January, 2009 – June, 2009.

I would like to express my sincere gratitude to my supervisor, Professor Audun Jøsang, for his continuous support, guidance and valuable feedback at every stage of my work with this thesis. Without his continuous guidance it would be very difficult to complete.

I would also like to thank my other supervisors, Professor Øivind Kure of Norwegian University of Science & Technology (NTNU) and Peeter Laud of University of Tartu, for providing me with valuable feedback time to time.

Finally, I would like to thank my wife, Farida Chowdhury, for supporting me consistently and encouraging me throughout my thesis.

Kjeller, June 2009

Md. Sadek Ferdous

Abstract

To have certainty about identities is crucial for secure communication in digital environments. The number of digital identities that people and organizations need to manage is rapidly increasing, and proper management of these identities is essential for maintaining security in online markets and communities. Traditional Identity Management Systems are designed to facilitate the management of identities from the perspective of the service provider, but provide little support on the user side in managing service provider's identity. The difficulty of managing identities on the user side causes vulnerabilities that open up for serious attacks such as Phishing. The main reason for which attacks like Phishing have been so successful is that it exploits the users' inability to differentiate between the genuine trusted and a fake web site, that is, because of mistaken identity. Therefore an aid to help users in managing service provider's identity is required. Petname Systems have been proposed to provide such user friendly and secure identity management of the service provider on the user side. This thesis aims at studying ways by which Petname systems can be applied in application related to identity management of the service provider on the user side.

The thesis will be mainly divided into two parts: the first part will provide an analysis of the Petname model by describing its history and background, properties, application domains and usability issues and in the second part, a Firefox extension based on Petname model will be developed and analyzed against a set of desirable properties. The aim is to show how such application can help users in managing service provider's identity in a convenient and secure way and thus presenting a defense mechanism against Phishing attacks.

Contents

List of figures

List of tables

Abbreviations

API	Application Programming Interface
CA	Certificate Authority
CN	Common Name
CSS	Cascading Style Sheet
DBMS	Database Management System
DHTML	Dynamic HyperText Markup Language
DOM	Document Object Model
DTD	Document Type Definition
GUI	Graphical User Interface
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
IdM	Identity Management
IdMS	Identity Management System
MSC	Message Sequence Chart
PAD	Personal Authentication Device
PC	Personal Computer
PIN	Personal Identification Number
PKI	Public Key Infrastructure
RDF	Resource Description Framework
SA	Security Action
SAML	Security Assertion Markup Language
SC	Security Conclusion
SDL	Specification and Description Language
SP	Service Provider
SSL	Secure Socket Layer
SSO	Single Sign On
SVG	Scalable Vector Graphics
TLD	Top Level Domain
TLS	Transport Layer Security
URL	Uniform Resource Locator
WLAN	Wireless Local Area Network
WWW	World Wide Web
XHTML	Extensible HyperText Markup Language
XML	Extensible Markup Language
XPCOM	Cross Platform Component Object Model
XPConnect	Cross Platform Connect
XPI	Cross Platform Installer
XUL	XML User Interface Language

Chapter 1

INTRODUCTION

1.1 Problem statement

The purpose of digital communication protocols is to exchange information as efficiently and reliably as possible. Originally, these protocols were designed without authentication because the identities of communicating parties could be assumed, and did not have to be formally verified. Authentication was subsequently added for verifying the correctness of claimed and assumed identities. Authentication requires prior registration of identities, and is based on a set of security mechanisms combined with a credential or security token. As authentication became necessary for accessing many online services, more and more identities and credentials were issued, and their management became problematic, both for service providers and for users. Identity Management (IdM, in short) was introduced by the industry to facilitate server-side management of user identities. Initially, client-side management of user identities was not considered to be an issue. However, many people currently feel overloaded with identities and passwords that security policies require them to memorize. The growing number of identities that users need to handle and the inability of users to comply with credentials management policies now makes client side IdM a critical issue. It is also important to consider that SP (Service Provider) also have identities and that also need to be managed by the users. However, this aspect of IdM has received very little attention and deserves more consideration at the alarming escalating rate of identity theft attacks, e.g. Phishing. In 2007, phishing caused a loss of US $3.2 billion in only US with the assumption that the financial loss might escalate in the upcoming years [1] and thereby imposing as one of the main sources for Internet Crime. Phishing has been mostly successful because of the user's inability of managing and identifying SP identities properly and securely on their side. Therefore wide-scale adoption of mechanisms for managing SP identities is timely. Petname Systems, as we will see, precisely focus on such client-side management of SP identities.

1.2 Contribution of the thesis

This thesis is mainly divided into two parts. At the first part we thoroughly analyze the Petname Model beginning with its history. It should be noted here that we will use the term Petname Model to denote the abstract properties of Petname Systems. An implementation of the Petname Model will be denoted as a Petname System. It is our belief that to understand the Petname Model it is essential to understand why Petname Systems were proposed in the

first place. The Petname Model was formally described by Marc Stiegler in his 2005 paper [2]. The potential of the Petname Model, however, was discovered by several people in several successive steps. Elements of the idea behind Petname Systems are scattered among several papers and web articles, and the combined efforts of these authors have shaped the formulation of the Petname Model. We aim to summarize the existing literature at first.

We, then, define the Petname Model by outlining its different components and establishing the relationship among them. A Petname System can have several properties and its potential applications can span over several disciplines of computing and networking. A long list of properties as well as several application scenarios were listed in [2]. However, we formalize the properties in a more systematic way by dividing them into two broad categories: 1) Functional properties and 2) Security Usability properties, and also by adding novel usability requirements. Then, Security Usability of Petname Systems based on those Security Usability properties will be analyzed and different applications of the Petname Model are explained. The current thesis introduces some new application scenarios other than those discussed in [2]. We will also analyze the Security Usability issues of two applications that utilize the Petname Model. Security Usability of Petname Systems was never explored and all these analyses are novel addition to the concept of Petname Systems.

At the second part of the thesis, we develop a Firefox extension based on the Petname Model as an aid for client-side management of SP identities and analyze its Security Usability issues. We will show how this software is a significant improvement over other similar Firefox extensions and how it provides a better and improved user-experience. Then we conclude with some hints on potential future work and research direction on Petname Systems.

1.3 Thesis structure

The rest of the thesis is outlined as follows:

- Chapter 2 explains briefly some of the concepts that will be used frequently in this thesis and that are necessary for understanding Petname Systems. The relevant terms are entity, identity, identifier, digital identity, Identity Management, Identity Theft and Phishing.
- Chapter 3 describes the Petname Model in details starting with its history and continuing with its components, properties and application domains.
- Chapter 4 analyzes the Security Usability issues regarding the Petname Model and then analyzes Security Usability of two applications that utilize Petname Model using Cognitive Walkthrough method.
- Chapter 5 discusses the developed Firefox extension and the related technologies that have been used to develop it in a detailed fashion.
- Chapter 6 analyzes the Security Usability of the developed application and shows its improvement on Security Usability issues over other available extensions.
- Chapter 7 summarizes our thesis and provides our concluding remarks on the thesis with hints on potential future work and research direction on Petname Systems.

Chapter 2

PRELIMINARIES

2.1 Entity

An entity is a physical or logical object which has a separate distinctive existence either in a physical or a logical sense [3]. Both animate and inanimate objects can be denoted as entity. Different disciplines employ the term *Entity* in different ways and also with different meaning. For example, in Database Management System (DBMS), each drawing element in an Entity-Relationship-Diagram is referred to as an entity, in HTML and XML, entity refers to a specific keyword to display special character in the web browser, etc [3]. In the scope of this paper, a person, an organization or a machine (computer) operated by any person or organization will be denoted as an entity.

2.2 Identity

Different disciplines (Philosophy, Social Science, etc.) interpret identity in different ways. There are also different definitions of identity which can be quite complex to understand and sometimes even contradictory. By putting aside the philosophical debates and contradictory arguments, a simple but intuitive definition can be provided [4]: Identity is the fundamental property of any entity that declares the uniqueness or sameness of itself and makes it distinctive from other entities in a certain context. In general, an entity can have multiple identities, but an identity cannot be associated with more than one entity. For example, one person can be a client of a bank, hence may carry the identity given by the bank and he can also be a player of a football team and may carry the identity of a player of that team. However, there may exist shared identities in a specific context. For example, in a group of football team all the players may be identified with the same identity.

Each identity may consist of a set of characteristics. Such characteristics are denoted as the identifier, name, label or designator. In general, the most representative characteristic of an entity within a specific context is denoted as the identifier for that entity. Other than being a unique identifier, those characteristics can have several properties [4, 5]. They can be intrinsic, meaning inherited naturally, e.g. hair or eye color, height or other physical attributes

or they can be extrinsic, meaning provided to an entity by other entities, e.g. name, address, social security number, etc. Characteristic can also be persistent, e.g. date of birth, biometrics, etc. or temporary, e.g. address, role in organization, etc. Such other properties can be: self-selected or given by an authority, suitable for human or suitable for computers only and so on. Figure 2.1 illustrates the conceptual relationship among identities, the entities they correspond to and the attributes that each of the identities may consist of.

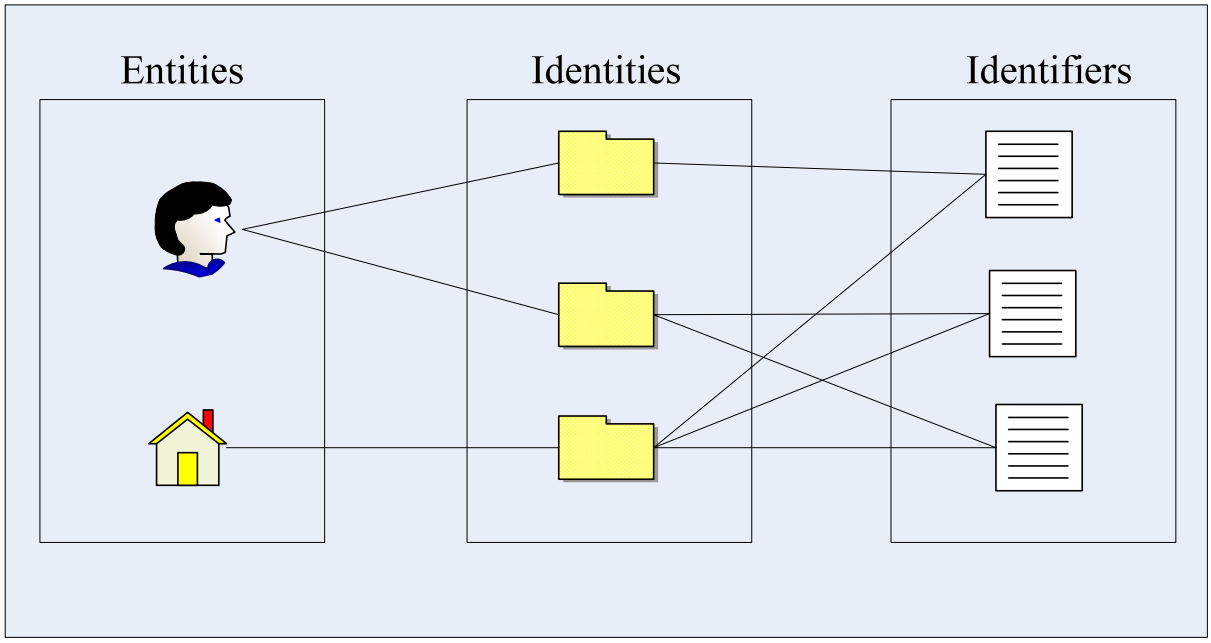

Figure 2.1: Relationship among Entities, Identities and Identifiers

It is crucial to note the degree of confusion between identity and identifier in common language usage. The term "*identity*" and "*identifier*" are often used interchangeably, though a fine line of separation exists between them in reality. However, we will use the terms with their distinctive meanings throughout this thesis.

The domain in which each identity is unique can be regarded as the Identity Domain. Each identity domain may have different type of namespaces. There exists a unique one-to-one relationship between identities and their corresponding identifiers in a namespace if that namespace only consists of unique identifiers. It is rather a challenging task to design a suitable namespace, and generally the larger the domain, the trickier it is to define such suitable namespace of unique identifiers.

Human beings are equipped with the ability to intuitively identify an entity based on an ad hoc set of characteristics and also in varying contexts, but a machine is not. To enable a machine to identify other entities, Digital Identity is required.

2.3 Digital Identity

The digital encoding of an identity can be defined as a Digital Identity. It is the representation of an identity in a form that is suitable for representation and processing in computer systems.

In all types of digital communication (Internet, telecommunication) digital identifiers are being used in the form of URL, user-id, phone number, etc. While some digital identifiers (e.g. URL and phone number) are strongly resistant against forgery, other such as user-id is not. They can be easily duplicated. Therefore another additional attribute, known as credential, has been introduced to accompany user-id. Such credential is assumed to be secret to others except to the valid user and sometimes to the issuing or authenticating authority. The simplest and also the weakest form of such credential is a password. For more secure authentication, biometrics such as fingerprint, retina scan could be used. Like other identifiers, digital identifiers (and also their corresponding credential) could be issued by a third party such as government or financial organization or it could be self-created. The former identifier is used for highly sensitive data transmission that requires higher degree of secrecy while the latter identifier is used for light-weight communication such as social communication, fan group, hobby club, etc. One of the serious problems the users are facing is the constantly increasing number of digital identifiers and their corresponding credentials as their interaction with digital world is increasing. The term Identity Management (IdM, in short) has emerged to aid the user in reducing the cognitive load of managing different identities.

For a good introduction to the concepts of identity and digital identity, see [4, 5].

2.4 Identity Management

A large number of service providers combined with a large number of users that access each service results in an even larger number of digital identifiers with their corresponding credentials that need to be managed. Formally Identity Management consists of technologies and policies for representing and recognizing entities as digital identities [6]. There are basically four types of identity management:

i. Managing user identities on the server side,
ii. Managing user identities on the client side,
iii. Managing server identities on the server side, and
iv. Managing server identities on the client side.

Traditionally, IdM refers to the Type 1 IdM, and the so-called Identity Management Systems are designed mainly for the purpose of managing user identities on the server side. In the next few sub-sections, we will take a brief look at the traditional model for identity management [5]. Each of these models involves several parties that include:

- **Service Provider:** A service provider usually provides service to the clients or to the other service providers. Examples include mobile phone operators, different web service providers, etc [7]. In its simplest form, service provider may also include identity and credential provider.
- **Identity Provider:** An identity provider provides digital identity to the clients to enable them to receive service from a service provider. In its simplest form, it may also include a credential provider.
- **Credential Provider:** A credential provider provides related credential for a digital identity issued by an identity provider.

- **Client/User:** A client/user receives service from a service provider. To receive the service, the client usually needs a digital identity and related credential to authenticate him as the valid user of that service.

2.4.1 Isolated User Identity Model

This model represents the most common and the simplest IdM model. There are only two parties involved in the scenario: service provider and its clients. Each service provider provides the identity and the corresponding credential to the clients who wish to receive its services. That means that each service provider has its own identity domain and manages its own namespace locally. Though this model represents the simplest form of IdM from the service provider's perspective, it creates immense burden for the clients as they need to remember literally hundreds of user-ids and passwords and soon those become unmanageable. Figure 2.2 illustrates this model.

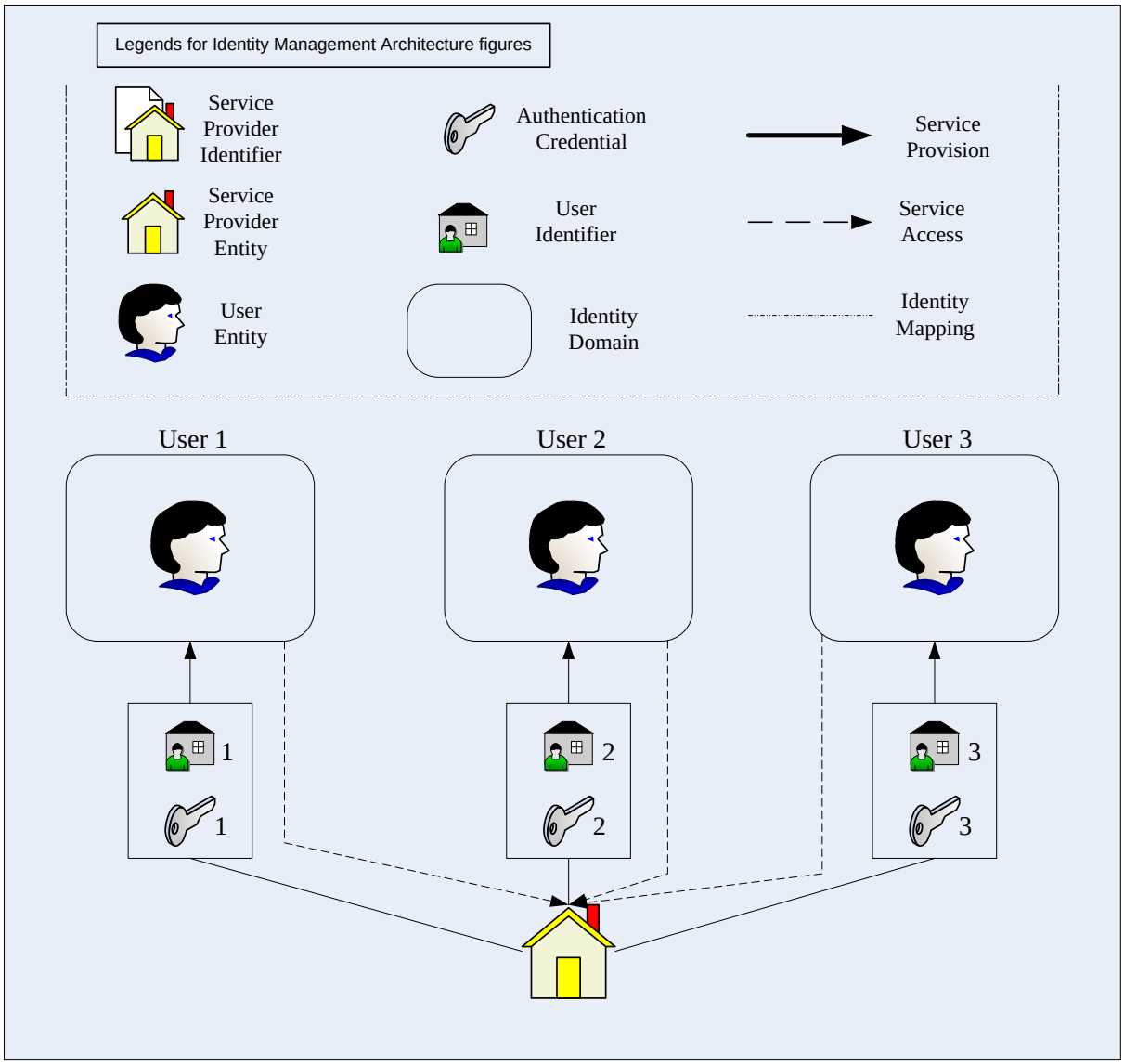

Figure 2.2: Isolated User Identity Model

2.4.2 Federated User Identity Model

In this model each service provider has their own identity domain but each domain is shared among the previously agreed service providers and thus providing a unified namespace among all the service providers. Such identity domain is commonly known as Federated Domain. Client of one service providers can authenticate himself with his corresponding service provider and then can enjoy the service of all the service providers if there is a federated agreement between them. Users can, of course, have multiple identities issued by multiple federated service providers, but usually one identity is sufficient to avail all the services. Industry has responded very well with this model as the users don't have to remember a large number of identities and credentials and several technological standards have been proposed that include OASIS Security Assertion Markup Language (SAML) [8], Liberty Alliance Architecture [9], Shibboleth [10], etc. Figure 2.3 illustrates the concept of this model.

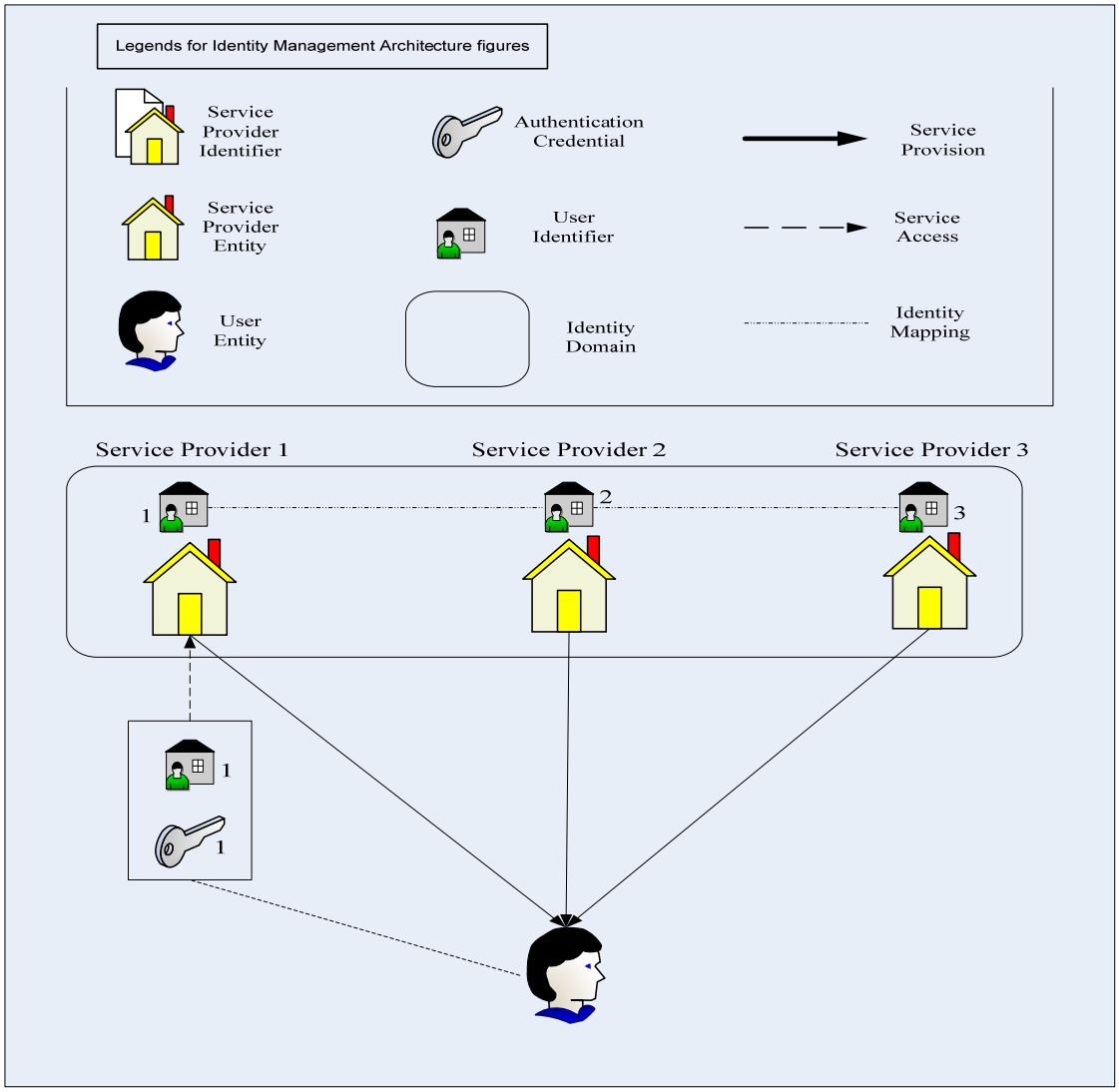

Figure 2.3: Federated User Identity Model

2.4.3 Common User Identity Model

In this model there is a common identity and credential provider for all service providers. The common identity and credential provider issues identifier and related credential to the clients and clients can use them to access the services provided by the service provider. Usually clients require one identifier and its credential to access services of all service providers. The main challenge of this model is to establish a common identity domain and maintain the common namespace when it reaches a global scale. Figure 2.4 illustrates the concept of this model.

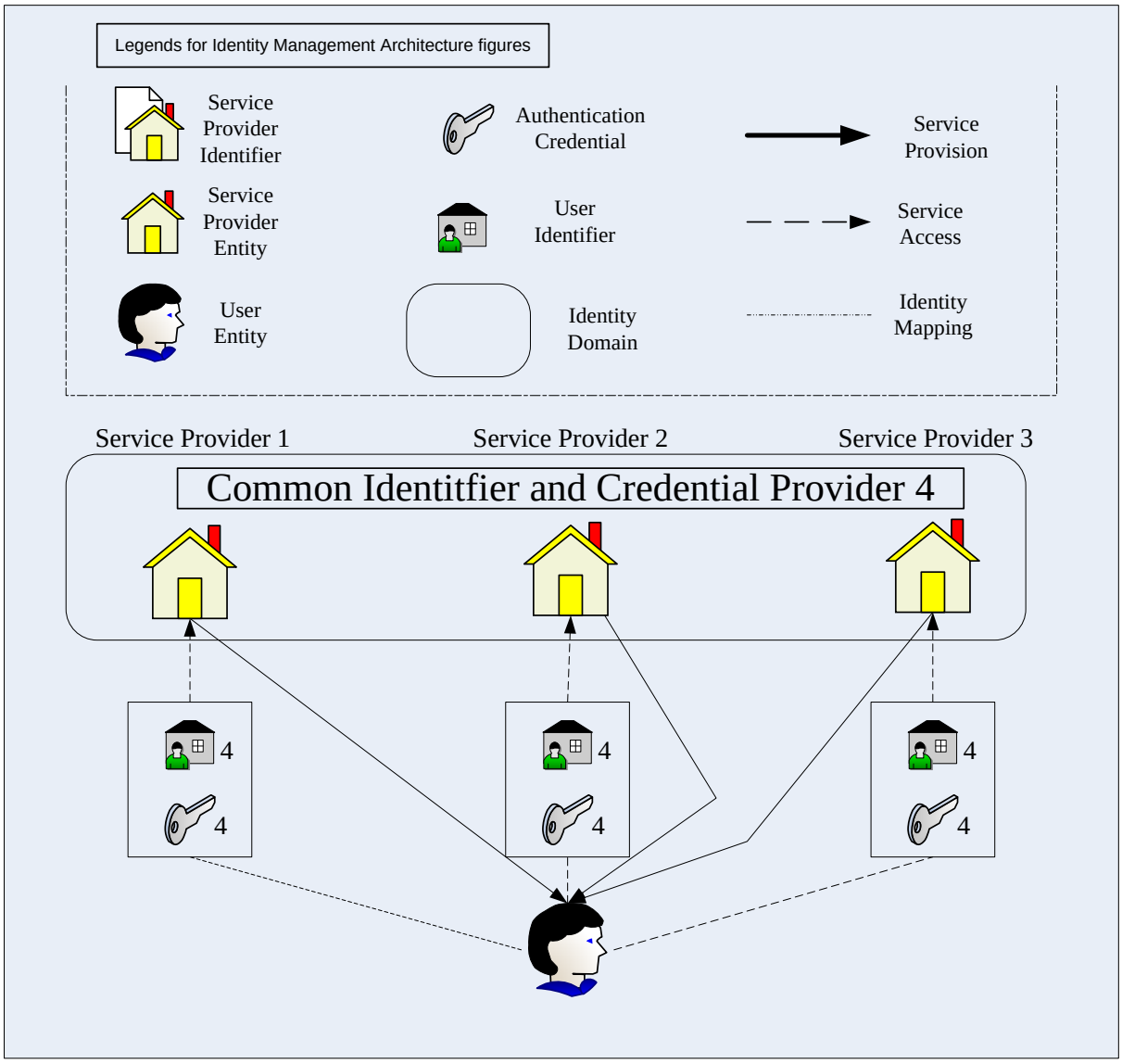

Figure 2.4: Common User Identity Model

2.4.4 Meta User Identity Model

In this model all the service providers share the identity related data on a Meta level by mapping all service providers' identifiers into a unique Meta-identifier and then issuing a credential related to that identifier (Figure 2.5). Each service provider maintains their own

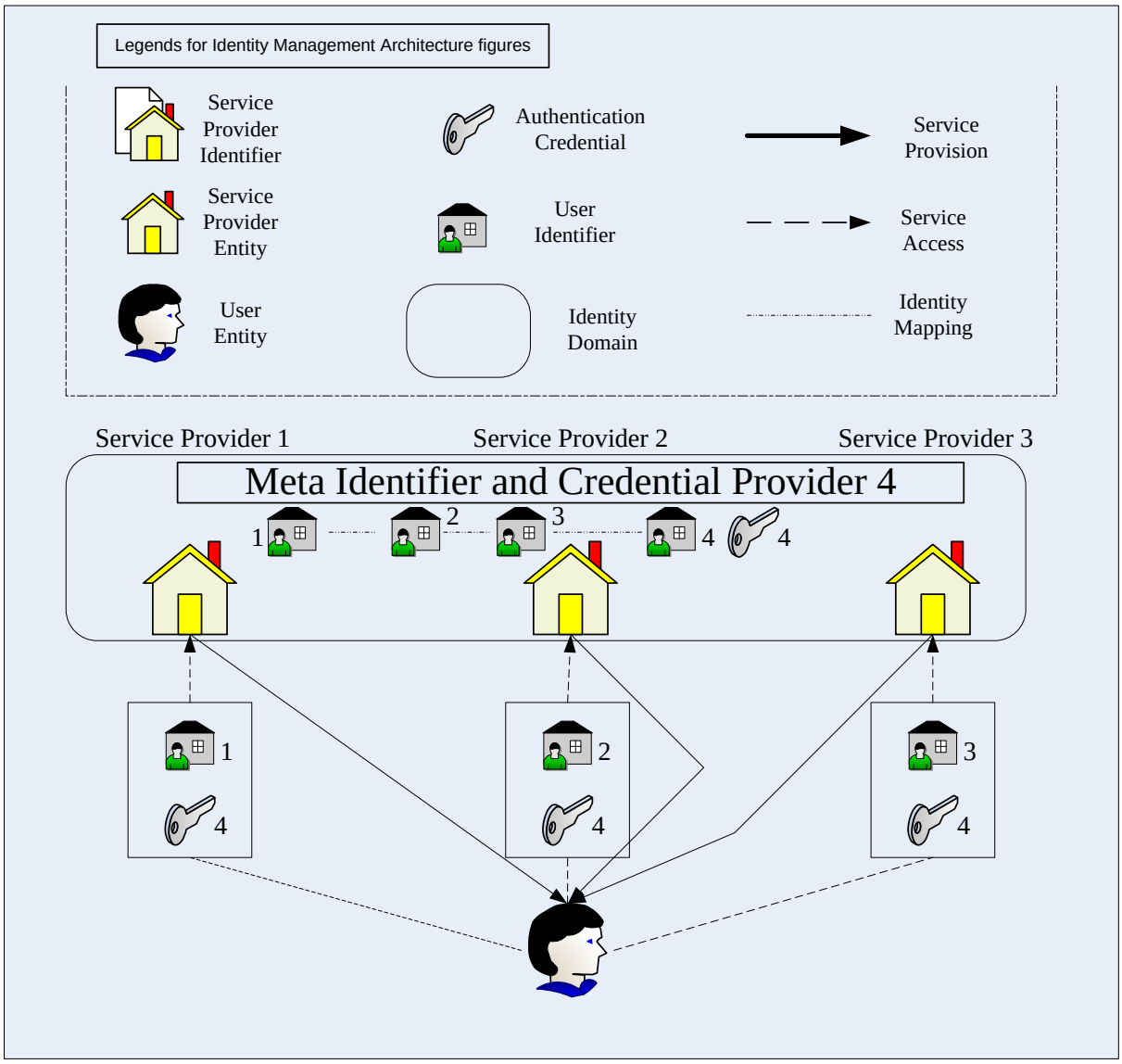

Figure 2.5: Meta User Identity Model

identity domain and issues identifier and credential to their clients. Clients only need the single identifier and credential to access service of all providers and normally not aware of the Meta-level. The Meta-identifier is used internally by the service providers to manage identity across different identity domains.

2.4.5 Single Sign On Identity Model

In this model, there is a single identity and credential provider for all service providers. The identity and credential provider issues identifier and the related credential to the clients on behalf of the service provider. To access service, clients authenticate himself to the identity and credential provider and once authenticated, he is redirected to the service provider's domain to access the service. Once a client is authenticated he can access the service of all service providers. Microsoft.Net Passport is an example of such model. Figure 2.6 illustrates the concept of this model:

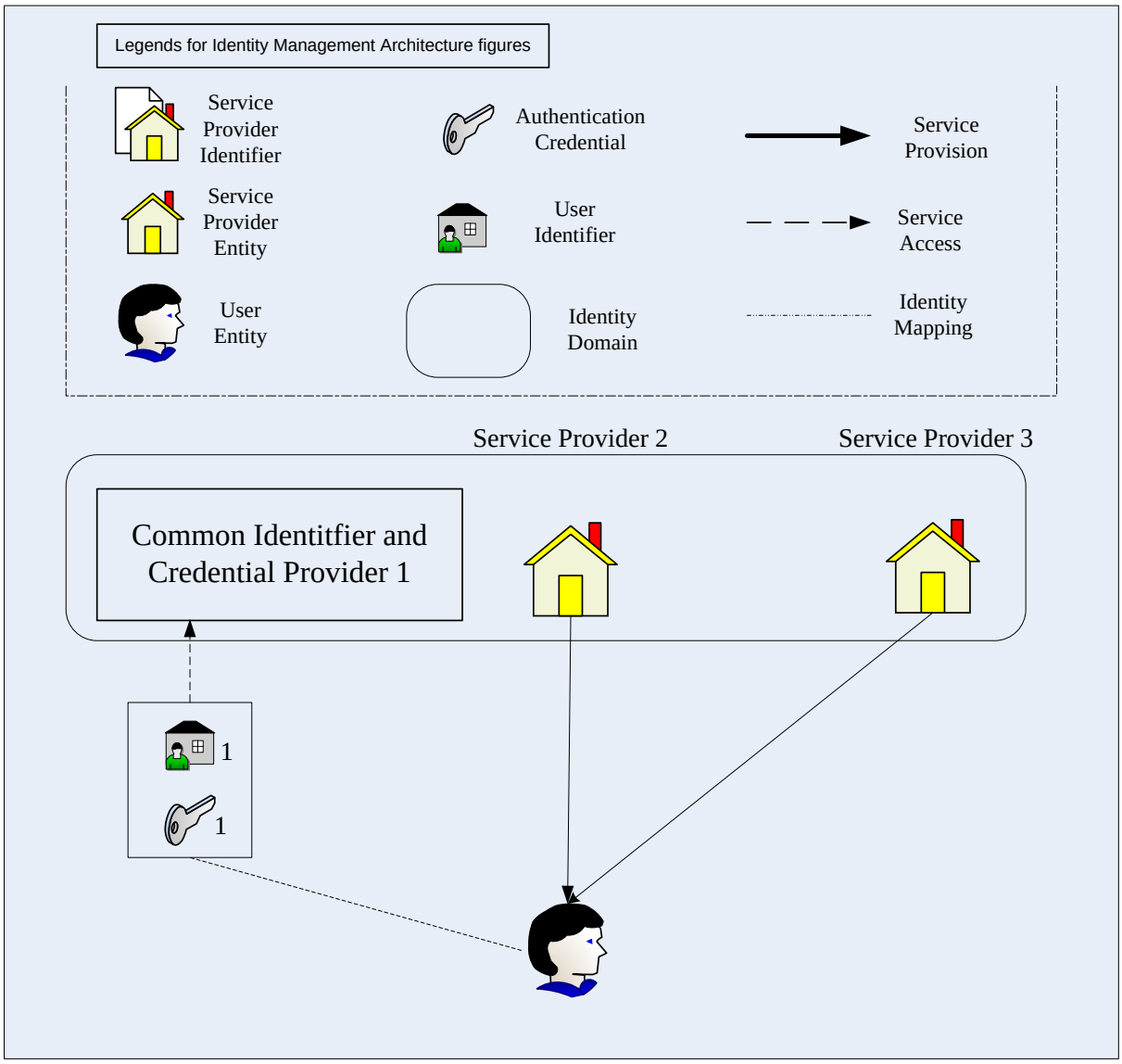

Figure 2.6: SSO Identity Model

2.4.6 User Centric Identity Management Model

The traditional Identity Management Systems that are discussed above are designed mainly for the purpose of managing client identities on the server side. Typically, an identity management system is implemented in software on the server side. However, clients also need to manage their own identities, and service providers also have identities that need to be managed. Unfortunately, these issues are mostly overlooked, and clients currently have little support in the form of software solutions on the client side to manage their identities and to manage service provider identities. Highlighting these points, a user-centric Identity Management model was introduced in [5] to manage client identities. In this model, different service providers will issue identifiers and the corresponding credentials to the clients like some previously discussed models. However, unlike them, these identifiers will not be handled manually by the client; instead a Personal Authentication Device (PAD) could be used to store and manage these identities and their related credentials. The client will authenticate to the PAD using PIN or other mechanisms and PAD then either can interact with the client machine (e.g. PC) using Bluetooth or WLAN or can interact with the service

provider directly using WLAN. Once this double authentication is done, the client can directly access the services from the provider. A conceptual diagram of this model is given in Figure 2.7.

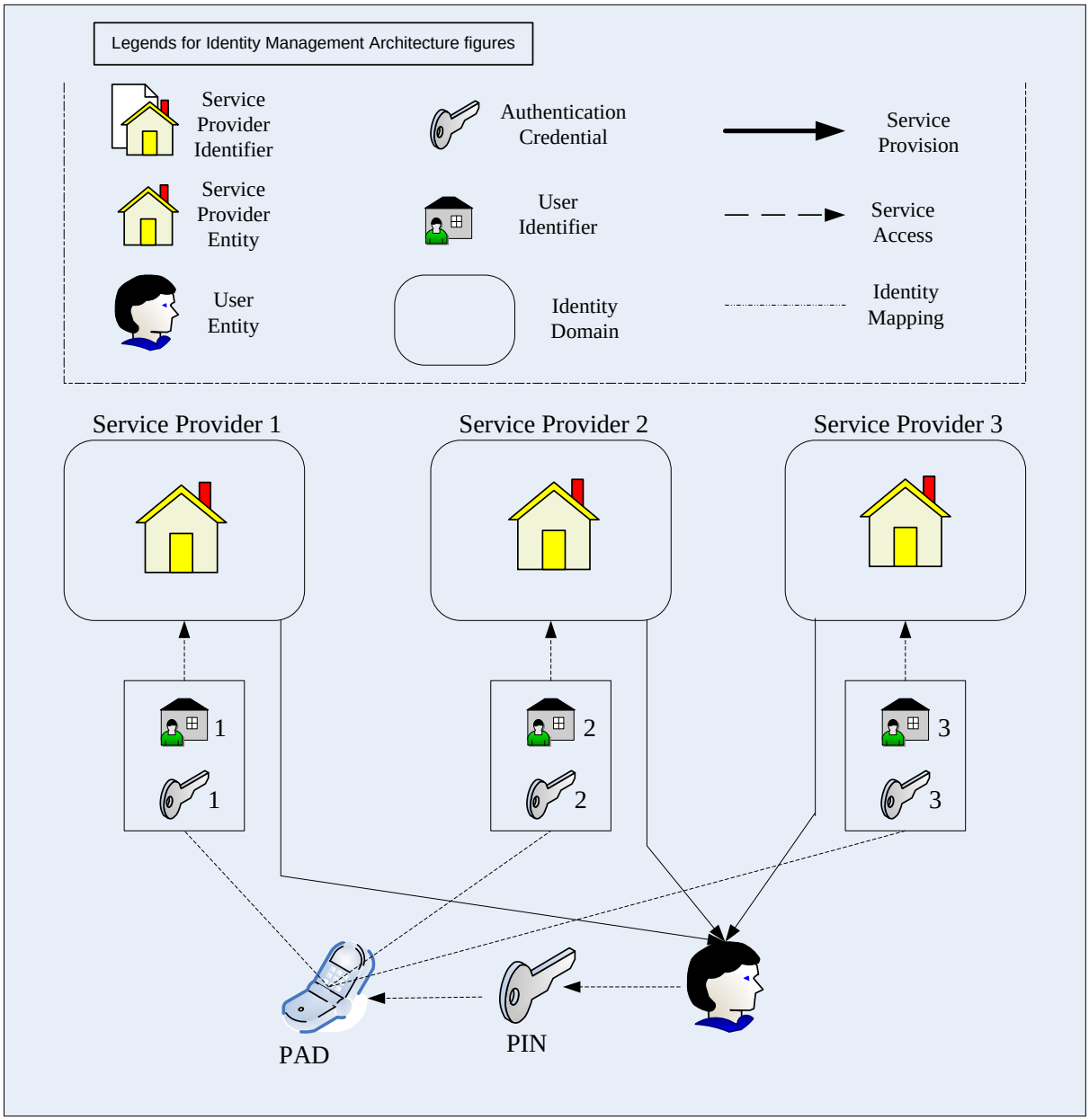

Figure 2.7: User Centric Identity Management Model

The next three models will concentrate on managing service provider identities on the client side.

2.4.7 Common SP Identity Model

In this model, the identity of the service provider is received from a third party and is managed manually by the client. This model is widely deployed in the Internet communication using Web PKI combined with SSL/TLS Security protocol. The client's browser receives the identity of the service provider from a certificate in a secure

communication using SSL/TLS protocol and validates on behalf of the client and informs the client about this validation using different techniques. This model has one major drawback: the identity validated by the web browser is not always the identity the user wishes to access. Using the technique of Typo Squatting, a technique in which similar domain names that only vary in one or two letters are utilized (e.g. *www.paypal.com* and *www.paypa1.com*, the second is 1 not small 'L'), an attacker can sometimes trick a user and the web interface of the fraudulent site may be an exact copy of the real one and even can be validated by the web browser and thus convincing the user to believe that he is using the real site. Therefore, a certificate based on SSL/TLS is cryptographically sound, however, they even fail to consider whether the website is the one that the users intend to access. Figure 2.8 illustrates the concept of this model.

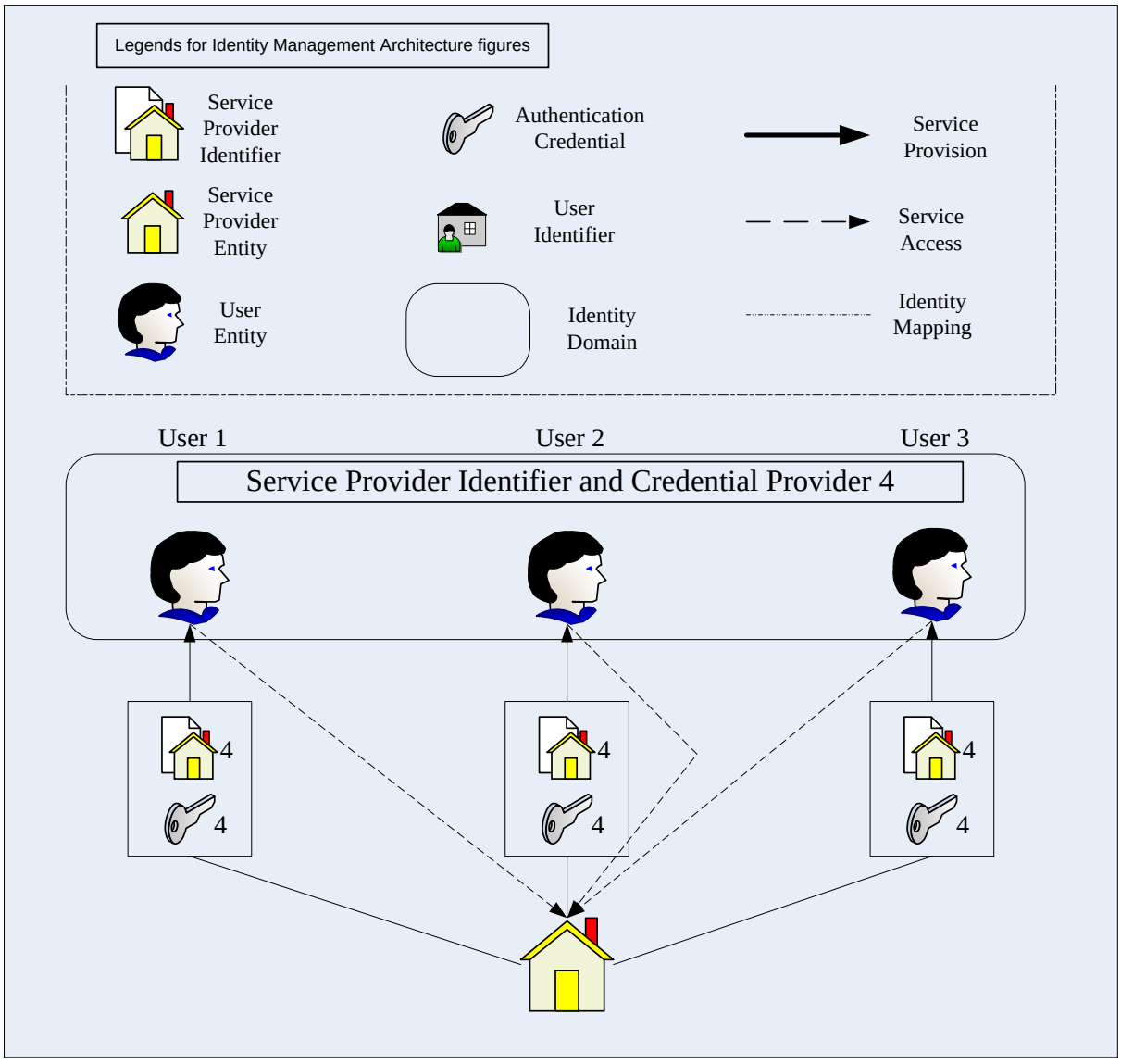

Figure 2.8: Common SP Identity Model

2.4.8 Isolated SP Identity Model

In this model, each client utilizes a personal namespace to assign a unique identifier and the related credential for each service provider. The service has to present that unique identifier and the corresponding credential to the client it is serving, meaning a service provider has to provide different identifiers and credentials to different users. Figure 2.9 illustrates the concept of this model.

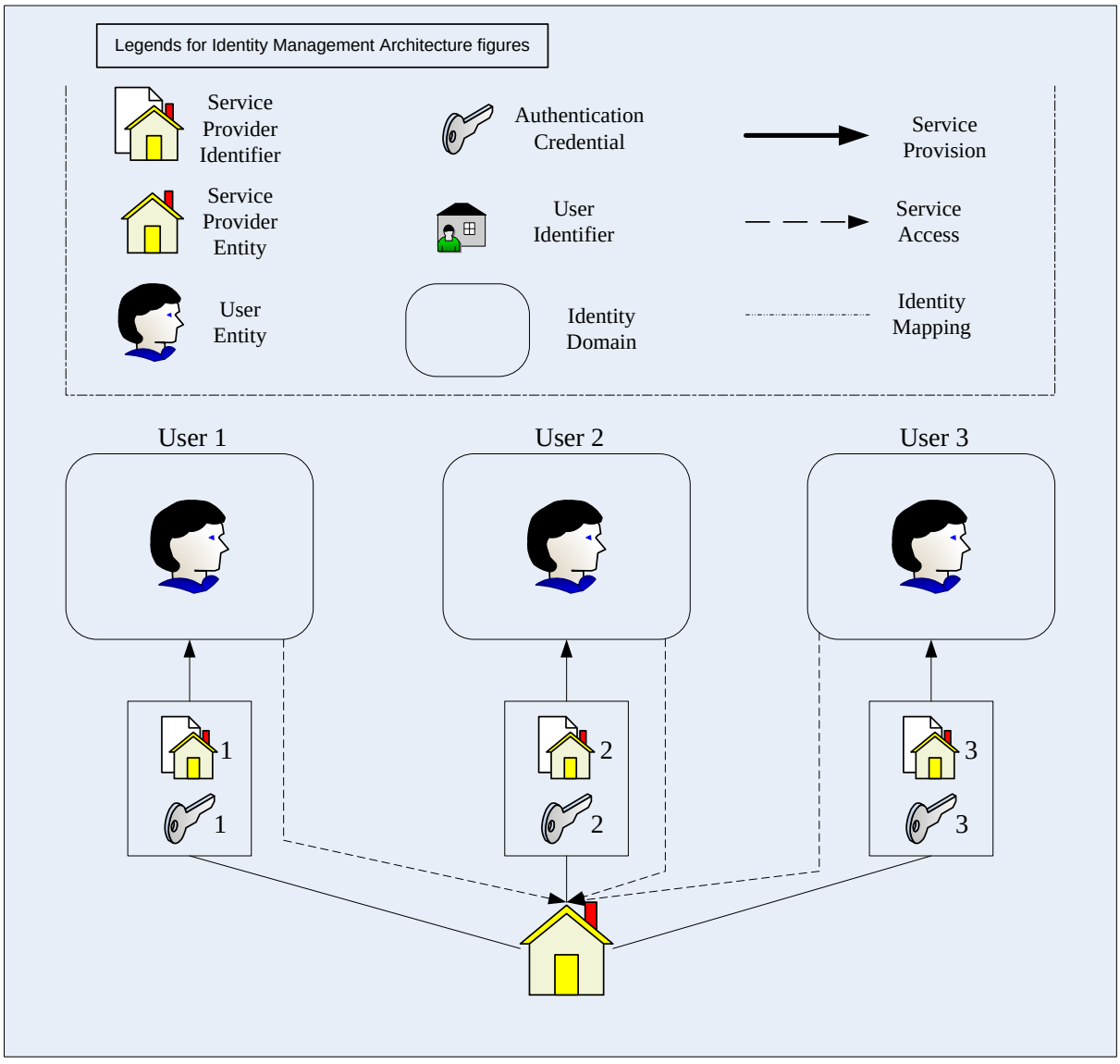

Figure 2.9: Isolated SP Identity Model

As observed from the figure, this model is the upside-down implementation of the Isolated User Identity Model.

2.4.9 Personal SP Identity Model

This model is the combination of the Common and Isolated SP Identity models. Here, each user utilizes a personal namespace to assign a unique identifier for each service provider. This can be done by providing a local mapping between the user created identifier and the real identifier of the service provider within each user domain. The user created identifier can consist of Text, Image, Sound or any combination of several of them. However, the identity of the service provider is validated formally. For such validation, the approach of the Section 2.4.7 is used, that is, Web PKI Certificate with SSL/TLS protocol is used for validating the identity of the service provider. The Petname Model is actually the advanced implementation of this model with some additional features. Figure 2.10 illustrates the concept of this model.

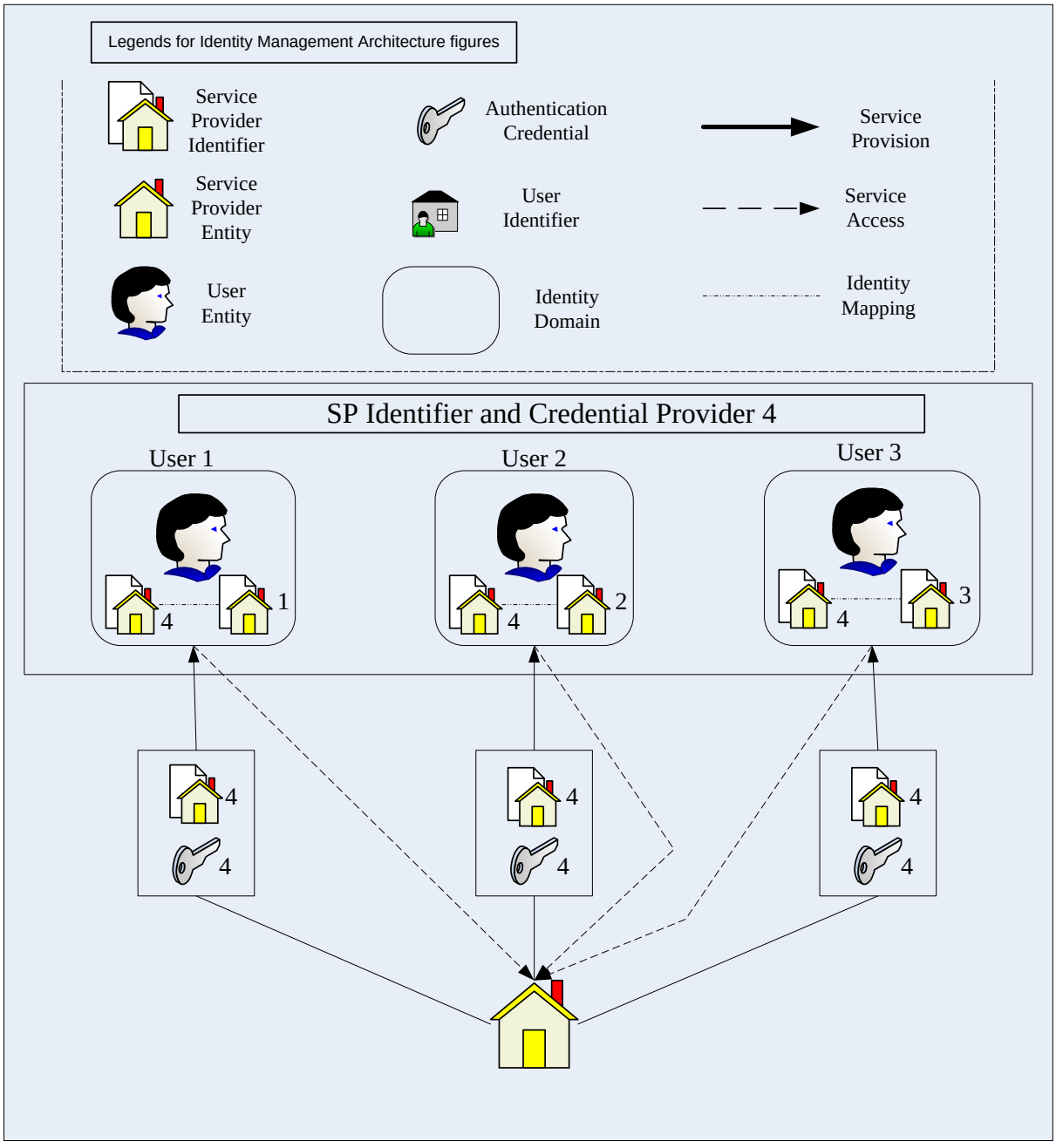

Figure 2.10: Personal SP Identity Model

2.4.10 Other Standards

There are some other standards, other than those described in the previous sections, available in the market that more or less deploy the similar or slightly modified models. These include: Microsoft's Card Space [11], Eclipse's Higgins [12], OpenID [13], SourceID [14], DotGNU Virtual Identities [15], etc.

2.5 Identity Theft

Identity theft can be defined as a crime in which someone (perpetrator) pretends to be someone else (victim) in order to attain financial or other benefits [16]. The victim will be held responsible for whatever the perpetrator does and may endure severe consequences for something that he did not do. However, the term is actually a misnomer, since it is not actually possible to steal someone's identity, only can be misused. In the scope of the thesis, Identity theft will usually refer to the theft of digital identity. Identity theft is one of the major sources of financial losses over Internet and the financial loss due to the Identity theft was estimated around $56 billion dollars in 2005 [17]. There are many techniques that perpetrators can deploy to execute an identity theft attack and Phishing is one of the mostly used.

2.6 Phishing

Phishing can be defined as an attack in which people are tricked to provide their identities and related credentials (e.g. user-id and password, respectively) or other valuable financial information, e.g. credit card details, to a fraudulent entity that masquerades as a valid entity in a communication that usually takes place over Internet. Etymologically, the term is believed to derive from the term Phreaking, a hacking art in which the phone network is cracked to make free long-distance call [18]. The reason behind such naming is simple: attackers try to catch fish (victim) over Internet and it is also believed that substituting 'f' with "ph", the attacker could exchange different tricks without being noticed [19]. Attackers who are involved in phishing are sometimes known as 'Phishers' [20]. Phishing is considered one of the major sources of financial losses over Internet and caused a loss of US $3.2 billion in only US in 2007 [1]. Phishing exploits different techniques as well as takes advantage of the poor security usability issues of the existing web security technology [21]. Different phishing techniques and the current practices to defend against Phishing will be briefly explored in the next two subsections.

2.6.1 Phishing Techniques

Attackers have utilized different Phishing techniques that have changed over time. We list some of them here [20]:

- **Social engineering:** Social engineering is one of the widely deployed techniques used for tricking people to reveal their sensitive information. It is used not only for Phishing also for many other attack forms. For Phishing, the most common method is to send convincing-looking e-mails to the individuals stating something that might catch their

immediate attention. That e-mail usually contains a link of a fraudulent site which may have same feel and look of the original site. This arrangement may trick the user believing that he is interacting with the valid entity and allures him to provide sensitive information like user-id and password or credit card details. Once provided, these information are abused to achieve financial gain or for some other purposes. Such method has been so successful from the attacker's point of view that a numerous research is going on to remedy the situation and well-known financial institutions take every measure to keep their client aware of the situation [20].

- **Link manipulation:** This technique is actually used for the above mentioned social engineering method [20]. As stated earlier, e-mails sent by the attacker contains link of fraudulent website which looks similar to the original one. The attackers use the link manipulation technique to trick the user. It has many forms. In one form, the text of the link will be a valid name while its hyperlink is the fraudulent one. For example, text of the link may contain *www.paypal.com* while its hyperlink can redirect to *www.attackersite.com* where the later site will be carbon-copy of the first one. In another form of link manipulation, user's poor knowledge on Domain Name System (DNS) is exploited. For example, user may believe that a link with *www.paypal.attackersite.com* will take him to the PayPal site. Another type of link manipulation method could be the use of '@' in the domain name. It also exploits the poor knowledge of the user on DNS. For example, many users may believe that *www.paypal.com@attackersite.com* will take him to the PayPal website.
- **Website forgery:** This type of attack methods can be very sophisticated and sometimes extremely difficult to track down. Attackers utilize many clever techniques to trick the user. Once a user clicks a link of a Phishing e-mail, he is redirected to a fraudulent website in where not only the look and feel are same, the address bar of the browser are also altered using scripting language so that the user can not detect that he has been tricked. Sometimes attackers exploit the flaws that are found in the trusted organization's website. Once such type of flaw was used in 2006 against PayPal [22]. A *Universal Man-in-the-Middle Phishing Kit* was discovered by the RSA Security in 2007 that allows attackers to create fake URL easily and the fake URL can communicate with the valid site in real time to capture sensitive information [23]. Another method deployed by the attacker is to redirect the user in a legitimate site and at the same time opening another site and then placing a pop-up window over the legitimate site in a way that the user may be tricked to believe that the legitimate site is asking for user-id and password [20].
- **Phone Phishing:** Another method is on the rise is the Phishing over the phone, also known as Vishing (Voice Phishing) [20]. Users are contacted over phone in such a way that they are convinced of its legitimacy and may reveal sensitive information to the attackers.

2.6.2 Defense Mechanisms

Several defense mechanisms have been put into practice to combat the growing number of Phishing attacks. We briefly describe some of them below [20]:

- **Social Awareness:** One effective method against Phishing attack is to educate people to identify Phishing attempts. Combining this with a cautious change of their browsing habit can reduce the Phishing attack significantly. And also if people can be educated on how to validate the security of a website with certificate, it could also help reduce the number Phishing attacks.

- **Anti-Phishing software:** Almost all modern browsers have anti-Phishing software built-into them [20]. These anti-Phishing softwares generally consult a third-party provided black-list of domain names that were known to be involved in Phishing. If a user is redirected to any of the websites that is found in the black-list, an active warning message is displayed. Any user also can report a suspicious website to those third parties for verification.
- **Improved spam filter:** One of major sources of the Phishing is unsolicited e-mails or spam. An improved spam filter can significantly reduce the number of spams that reach the inbox of users which in turn can also reduce the number of Phishing attacks.
- **Legal responses:** Several countries have introduced strong legal action against Phishing attacks that include a huge amount of monitory penalty and/or a long term detention in jail [20]. Many phishers were already arrested and convicted in several countries around the world.
- **Other techniques:** Researchers around the globe are trying to come up with different ideas to combat Phishing. One such idea is to use a "*Security Skin*"; that is, a user selected text or image will be overlaid into the login form of the website [24]. As only the browser and the user will know of such selection, a fraudulent website will fail to show such selection and thus helping the user to decide on which websites are legal. Implementation of such technique already put in practice by the log-in page of the Yahoo! Service [25]. In this page a user can create a personalized sign-in seal using a text or an image that will be displayed whenever a user visits that site again. We believe that this idea is good but it will require changing the behavior of a webpage to embed this technique which may not be scaled globally. As we will show that the Petname Model offers a better and intuitive approach to combat Phishing.

Chapter 3

PETNAME SYSTEMS

3.1 Background & Rationales of Petname Systems

IdM roughly consists of three phases [26]:

1. **Registration Phase:** An identifier is created to identify an entity uniquely. A corresponding credential may also be supplied along with the identifier. The identifier and the credential are kept as long as there is a relationship between the entity and the service provider.
2. **Operations Phase:** The entity provides the identity and the corresponding credential to the IdM in the server side for authentication and access control.
3. **Deregistration phase:** When the relationship between the user and the service provider ceases, the identity is normally deregistered so that it can no longer be used for accessing the service.

In the first phase, the Identity Management System (IdMS) has to generate and issue an identifier for each entity. The IdMS uses a namespace from which an identifier is selected or chosen. Simply, a namespace is a logical and abstract set of names that can be used to uniquely select an identity for an entity. The main requirement for an identity is uniqueness such that each identifier maps to a unique entity. It is obvious that the same identifier can be used to represent different entities in different namespaces. The larger the namespace, the more unique identifiers it contains. However, a global namespace will normally suffer from the shortcoming that interpretation and memorization by humans becomes problematic. IP address is an example of such a global namespace. While it is possible to remember a few IP addresses, the mental load of remembering and accessing a large number of web sites by their IP addresses would be intolerable for normal users.

Three desirable properties of an identifier were defined by Zooko Wilcox-O'Hearn in his influential web article published in 2001 [27, 28]. According to Wilcox-O'Hearn, an identifier should ideally be *Global*[1], *Unique*[2] and *Memorable*[3]. To be memorable, an identifier has to pass the so-called "*moving bus test*" [29]. That is, if one can correctly remember a name written on a moving bus for a fixed duration, that name can be considered memorable.

[1] Called *Decentralized* in [27]
[2] Called *Secure* in [27]
[3] Called *Human-Meaningful* in [27]

An identifier will be unique if it is collision-free within the domain [2] and has the property that it cannot be forged or duplicated or mimicked. Wilcox-O'Hearn also claimed with supporting evidence that no identifier could have all the three desirable properties simultaneously, and suggested to choose any two of them according to different scenarios. Clay Shirky, in his 2002 web article, also came up with the same conclusion [30]. Any attempt to achieve all the three properties by any identifier could lead into the following problems:

1. Dependency on a third party which could monopolize the system and create a single point of failure [27].
2. Political and legal conflict may arise when an identifier becomes a trademark for different companies locally in several region and those companies compete for the same identifier when it reaches the global scale [30].
3. Unintentional confusion among almost similar identifiers, for example any confusion between two email addresses, e.g. rahim@bd.com and rahim@bd.net, can be very dangerous in life critical situation. Intentional confusion caused by e.g. Phishing attacks can also be disastrous [2].

A triangle where the three properties are placed in the three corners is commonly known as Zooko's triangle, and represents the basic foundation for the Petname Model. Zooko's triangle is illustrated in Figure 3.1.

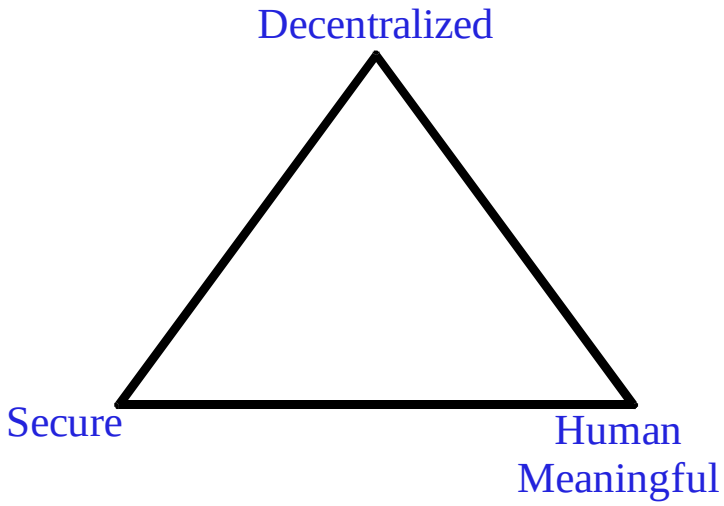

Figure 3.1: Zooko's triangle

The idea of placing the three properties at the three corners of a triangle can be explained as follows. In a triangle the three corners are never connected by a single line, only pairs of corners are connected. Placing those three properties in the three corners of the triangle provides a visual analogy to the fact that an identifier can only achieve two of the desirable properties at any one time.

In 2000, Jonathan S. Shapiro, being inspired by the idea of Marc Miller et al. while at Electric Communities, described in a web article about his scheme of adopting a system which utilized three types of naming conventions: *Petname*, *True Name* and *Nickname* [31]. He adopted this idea for a configuration management system. A True Name is synonymous to a global unique identifier, the Nickname is a global memorable assigned name of an entity by its creator, and the Petname is a memorable and locally unique user-assigned name for that entity.

A few months later in 2001, Mark Miller published another article in which he, for the first time, documented the structure of the Petname Model with three components: *Petname*, *Key* and *Nickname* [29]. These three components are essentially equivalent to Shapiro's *Petname*, *True Name* and *Nickname* respectively. Miller suggested to use the term *Key* instead of *True Name*, and pointed out that the Petname Model satisfies all the three desirable properties of Zooko's triangle. This idea was actually elaborated by Marc Stiegler when he formalized the Petname Model. Tyler Close of Waterken Inc. suggested to adopt the term *Pointer* instead of *Key* [32] and we will also use the term Pointer in this paper. This topic will be described in more details in the subsequent sections.

Tyler Close also pointed out the possibility of using Petname Systems for better trust management [33]. He developed the Petname Toolbar for the Firefox web browser. The main motif was to show the potential application of Petname Systems to counter Phishing attacks. According to Tyler Close, humans are not capable enough to manage the transition of trust from one entity to another in digital communication and this leads to identity-theft as a result of Phishing attacks. The next paragraph explains his view on the rationale behind Petname Systems.

Whenever we move from one website to another by clicking a hyperlink at the first site, there are two types of transitions that take place. One is the website transition that takes us to the next website and the second one is the transition of trust which enables us to retain or discard the trust relationship with the next website. We have different types of trust relationships with different entities. We may trust one entity more than another and with different scopes. As an analogy, when a user wants to buy something from an e-commerce website, he may not trust to give his credit card credentials to that site but he may trust PayPal. In this case, after choosing the item, the website may take him to the PayPal webpage and he completes the transaction there. But the problem here is to make sure that the e-commerce site takes him to the right PayPal site, not to a fraudulent one. Currently users are supposed to follow a set of steps to validate the identity of a website: 1) check if the target URL in the address uses the encrypted HTTPS protocol instead of the unencrypted HTTP protocol, 2) check if the received server certificate is issued by some trusted authority, and 3) check if the domain of the accessed site matches the domain specified in the certificate. Not only do these steps pose a significant mental load on the user, they even fail to consider whether the website is the one that the users intend to access [21]. This creates precisely the vulnerability that makes phishing attacks potent and successful. It is also observed that security is a secondary consideration from the user's point of view [34]. The primary issue is to conclude the transaction and buy the desired item. This leads the user to ignore the required steps. The malicious e-commerce site may exploit the technique of typo squatting, a technique in which similar domain names that only vary in one or two letters are utilized, e.g. as represented by PayPa1 (the last character here is number 1) instead of PayPal. When the fake website looks identical to the genuine PayPal website, most users will be tricked into believing that the fake website is genuine. That is, the transition of trust may not take place as desired. So Tyler Close concluded that it was unwise to perform both transitions on the recommendation from a non-trustworthy entity, and therefore suggested to use Petname Systems to enable manual trust evaluation by the user while the transition takes place.

It is interesting to note at this point the relationship between identity management and trust management, where applying them improperly may lead to identity theft attacks. A realistic scenario can be used as an example. In the brick-and-mortar world, we come across different people where the different biological differences help us identify each person uniquely.

Interaction with them enables us to decide whom to trust. Sometimes recommendations play a crucial role. When our near and dear ones tell us not to trust somebody, we usually do not trust him or her, though this perspective may change over time. So we usually identify a person at first and place trust afterwards. Now in the digital world this scenario is somewhat different. To trust a digital entity, recommendation is the best and sometime the only option. We read website reviews, blogs, etc. and receive advice from relatives and friends on which digital entity to trust for online transaction. We may learn from them that there is a website *www.paypal.com* (there are also other trusted websites for online transaction) which we can trust for online transactions, even before having accessed and identified it. Once the trust is placed, the only thing remaining is to identify the website which is truly the recommended one. It can also be the other way around, as for example when browsing and identifying unknown websites that are potentially suitable for a specific transaction, and choose to transact with a specific one that subsequently will be trusted based on positive experience. The first way obviously is the most hassle-free, and the second one requires the user to accept a certain risk of transacting with an unknown entity. Whichever is the best option, trust management and identity management are closely tied to each other when we try to derive a solution for identity theft. As we will see, the Petname Model provides a solution for both scenarios.

In 2005, Marc Stiegler extended the Petname Model based on Mark Miller's suggestion and also explained the detailed interaction among the components of the Petname Model [2]. He also formalized the properties and requirements for the Petname Model and gave examples of some applications of Petname Systems.

The evolutionary timeline in this section illustrates how the different topics of namespace, identity management and trust management are interrelated, and how they were combined to formulate the Petname Model.

3.2 The Petname Model

3.2.1 Rationale

As mentioned in the previous section, Zooko's triangle visualizes the hypothesis that no identifier can at the same time be Global, Memorable and Unique, but can only have two of the properties. Three unique pairs can be created using these three properties: 1) Global-Memorable, 2) Memorable-Unique and 3) Global-Unique. Even if no identifier can have all the three properties, a naming system can be designed to achieve all the three properties of the Zooko's triangle. The Petname Model represents one such naming system.

3.2.2 Components

The Petname Model uses three different types of names that in our terminology are called: Pointer, Nickname and Petname. These three name types actually represent the three sides of the Zooko's triangle and hence are synonymous to the three pairs discussed above. Detailed explanations for each of them are given next.

- **Pointer:** The Pointer was defined as *True Name* in Shapiro's interpretation and as *Key* in Miller's interpretation. A Pointer implies a globally unique and securely collision free identifier which can uniquely identify an entity. It inter-connects the *Global* and *Unique* corners of the Zooko's triangle. The security of the Petname Model mainly depends on two factors: 1) Difficulty to forge a Pointer and 2) Difficulty to mimic a Petname. A public/private key pair and a fully qualified pathname of a file in an Internet file server are good examples of Pointers. They are globally unique and difficult to forge. However, a Pointer (e.g. a public key, IP address, etc.) may not be memorable to human.

- **Nickname:** The Nickname inter-connects the *Global* and *Memorable* corners of the Zooko's triangle. It is an optional non-unique name created by the owner of the Pointer. The purpose of the Nickname is to aid in identifying the entity easily. The title of a web page that is displayed in the title bar of the browser is an example of a Nickname. Users may remember that webpage by the title, but another website may have the same title and can create a collision on the user's mind. Thus a Nickname is not necessarily unique.

- **Petname:** The Petname is a name created by the user to refer to a specific Pointer of an entity. Within the domain of a single user a bi-directional one-to-one mapping exists between Petnames and Pointers. A Petname connects the *Memorable* and *Unique* corners of the triangle. Petnames only have a local scope and may only be relevant for local jurisdiction. The same Petname can be used by different users to refer to either the same Pointer or to different Pointers. The security of a Petname System also depends on the privacy of Petnames and the difficulty to mimic a Petname. Here it is interesting to note that a Petname does not necessarily mean a text-based name. In addition to text, it can also be image and sound or any combination of all of the items in different ways.

The concept of *Referral* is also related to the Petname Model [2]. A Referral from a third party can consist of a Pointer and a so-called *Alleged Name* which is the introductory/referred name for an entity, like the Nickname. The distinction between a Nickname and an Alleged Name is that the Nickname is created by the owner of the entity and the Pointer, whereas the Alleged Name is provided by a third party. In the trivial case, the Nickname and the Alleged Name can be identical. If your friend sends you a message with the text “Best e-auction site” with the link *www.ebay.com*, then it can be thought as Referral where the text “Best e-auction site” can be interpreted as the Alleged Name.

3.2.3 Relationship among the Components

There is a bi-directional one-to-one mapping between Pointers and Petnames within the domain of each user. A Nickname has a one-to-many relationship to the set of Pointers. A Pointer is assumed to map to a single Nickname, but can map to several Alleged Names in the global domain. The relationship between Petnames and Nicknames can be confusing sometimes. In some situations, a Nickname can be used as a Petname or in other situations a Petname can be derived from the Nickname. A single Nickname can always be uniquely resolved from the Petname, but the Nickname is not necessarily unique for the Petname. For that reason, a Petname can not be uniquely resolved from a Nickname. Figure 3.2 illustrates this relationship. As seen from the figure, the Petname Model is actually a naming convention built on top of the Zooko's triangle.

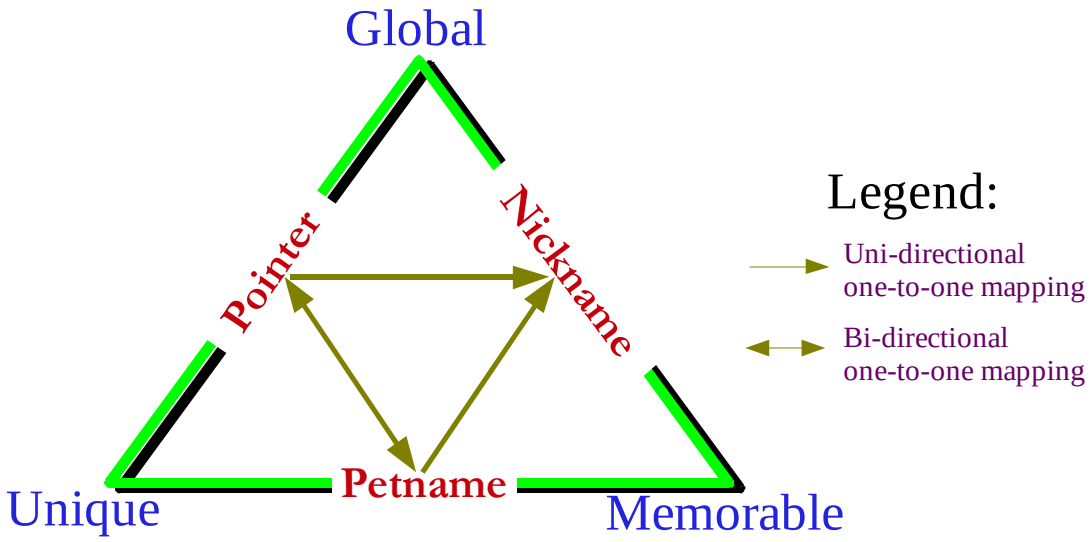

Figure 3.2: The Petname Model

It is fascinating to note here that other than providing a trivial bi-directional mapping, the relationship between the Pointer and the Petname offers a subtle indication of the trust transition that was mentioned previously. Thus a Petname can also be thought of as a trust indicator for the Pointer. In Section 3.4 it will be explained how Petnames can act as a trust indicator for Pointers.

3.3 Properties of Petname Systems

The properties of a Petname System can be divided into two broad categories: Functional properties and Security Usability properties.

3.3.1 Functional Properties

Functional properties are those basic properties that are mandatory for a Petname System. The functional properties are [2]:

F1. A Petname System must consist of at least a Pointer and a Petname.
F2. Nickname is optional.
F3. Pointers must be strongly resistant against forgery so that the Pointer can not be used to identify a false entity.
F4. For every user there must be a bi-directional one-to-one mapping between the Pointer and the Petname of each entity.

3.3.2 Security Usability Properties

Security usability will ensure the reliability of using the system and enables the user to draw conclusion on the actual security of the system. These properties will ensure that the Petname

System is not affected by usability vulnerabilities. Usability properties can again be categorized in two types [6]:

Security Action Properties: Security action enables a user to interact securely with an entity. Whenever a user needs to perform an action that is required to trigger or maintain the security mechanism of the system, can be defined as the security action. For example, submission of a password can be thought of as a security action. Properties related to the security action in the Petname System are [2]:

SA1. It is the user who must assign the Petname for the each Pointer.
SA2. Users must assign the Petname for the Pointer with explicit action.
SA3. As the relationship between the user and other entities evolve, the user should be able to edit the previously applied Petname for a Pointer to a new Petname.
SA4. Suggestion on the Petname based on the Nickname can be provided as an aid for the user to select a Petname for a Pointer. If the Nickname is missing, other criteria could be chosen for the suggestion.
SA5. If a suggestion is provided and the user wants to accept it as the Petname, then he must do so with explicit action.
SA6. Petname Systems must make sure that the user-selected, created or suggested Petname is sufficiently distinct from the Nickname so that the user does not confuse them with each other[4].
SA7. Petname Systems must make sure that the user-selected, created or suggested Petname must be sufficiently different from existing Petnames so that the user does not confuse them. This is needed to reduce the risk of mimicry of the Petname upon which the security of the Petname System largely depends
SA8. If the user chooses a Petname that may resemble a Nickname or other Petnames, he should be warned explicitly.
SA9. The User should be alerted to apply a Petname for the entity that involves in highly sensitive data transmission.

Among these properties, SA4 and SA5 are optional and the rest of the properties are mandatory to maintain the security of the system.

Security Conclusion Properties: Security conclusions enable the user to conclude on the security state of the system by observing security relevant evidence and assessing this together with assumptions. For example, drawing a conclusion that the communication over Internet is protected using web security technologies such as SSL/TLS by observing a closed padlock in the browser is a security conclusion. Properties related to the security conclusion are [2]:

SC1. The Pointer and the corresponding Petname must be displayed at all times through the user interface of the Petname System. This will make the user confident about his interaction and help to draw the security conclusion easily.
SC2. The Petname for a Pointer should be displayed with enough clarity at the user interface so that it can attract the user's attention easily.

[4] It might be acceptable that a Petname is equal to the Nickname in case a specific Nickname is unique within the user's local domain, but it would cause confusion and security usability vulnerabilities in case two or more Pointers correspond to the same Nickname in the user's domain. An alternative formulation of the SA6 property can therefore be that the Petname System must enforce that a Petname is different from the Nickname in case the Nickname is non-unique.

SC3. The absence of a Petname for a Pointer should be clearly and visually indicated at the user interface so that the user is surely informed about its absence.

SC4. The visual indication for Petnames and Nicknames should be unambiguous enough so that the user does not confuse them with each other.

SC5. The warning message that will be provided when there is a direct violation of any of the above properties should be clear enough so that the user can understand the problem and take the necessary security action.

All these properties are mandatory to maintain the security of the Petname System.

3.4 Application Domains

The presence of the Petname Model is so ubiquitous that people may sometimes be unaware of its existence. Here we will highlight the possible domains in which the Petname Model is used, intentionally or unintentionally, or could be used. For each of the applications we will try to determine the suitability of applying the Petname Model [2].

3.4.1 Real World

The principle of the Petname Model is so naturally integrated in the real world that we do not notice its existence. Let us first analyze how people actually recognize each other. This process is very simple and natural to us: through several physical attributes like face, voice, physique or maybe combinations of them. These combinations can be thought of as the Pointer to uniquely identify a single person. That single person introduces himself to us by stating his name XYZ which is actually a Nickname in the Petname Model terminology. From then on, we may perceive that man's identity as Mr. XYZ, which actually represents a Petname. Now if another person also introduces himself as XYZ, then our mind does not only assign that name as his Petname because it was already assigned to another person. Here things may evolve in different directions. One possible direction can be that our mind distinguishes between those two persons and changes the Petname for the first person as Mr. XYZ of London and Mr. XYZ of Paris for the second person or whatever seems practical.

3.4.2 Phone/E-mail Contact List

A phone/email contact list is another classic example of a Petname System. The phone number with international format (preceding the number with + or 00 and country code) may represent the Pointer and it is unforgeable and globally unique. We save the number in our contact book by placing a name for it which is nothing but a Petname for that number. Nicknames are absent here. The same analogy applies for email contact lists. Email addresses represent Pointers. A "*From-field*" in an email header may contain only the email address: *xyz@yahoo.com* or a given name by the sender with his email address: Mr. XYZ < *xyz@yahoo.com*>. Here the given name (Mr. XYZ) represents the Nickname. After receiving a mail from a new sender one can save the sender's email address in the email contact list. At that time a Petname is created by inserting a name suitable to identify that person, or by simply keeping the Nickname.

3.4.3 IM Buddy List

In the domain of a particular Instant Messaging Service each entity has a unique Id (email Id for yahoo, hotmail or passport service) which represents the Pointer for that entity. But sometimes those Ids can have quite close resemblance (logicman and 1ogicman, the second one actually is a 1 not a small L) to each other and thus can be quite confusing for the user to differentiate. A better option is used in the interface of the Instant Messenger where one can put a name for each of the IDs. Such name is actually a Petname. In the user interface all the interactions with the Id is usually done with the Petname and thus making the IM Buddy list a good example of a Petname System. Nicknames are absent here.

3.4.4 DNS

As mentioned earlier, that two domain names can be quite close to each other (typo squatting), intentionally or unintentionally, which can lead to Phishing or pharming attacks, Petname Systems can be a useful tool to thwart this type of attack. The domain name itself represents the Pointer. The title in the title bar of the browser for that domain name is the given Nickname. In the user interface (the browser), the user can provide a Petname for each domain name. All the interactions with that domain will be indicated by the Petname in the user interface. Providing a Petname for each domain name will impose a trust relationship to that domain name. Absence of Petname will indicate the absence of a trust relationship.

On the background of the above scenarios, the e-commerce transaction scenario of Section 3.1 can be revisited. A user frequently shops online and places his trust in PayPal to process his online transaction. Now to safely process his transaction he can define a Petname for PayPal in his browser. Assume that the user visits an e-commerce site that offers an item he wants to buy, but the users does not trust the site to know his credit card details. Luckily the site allows him to pay through PayPal, so he is redirected to *www.paypal.com* when the transaction enters the payment phase. Assuming that he has already defined a Petname for PayPal, his browser should indicate the Petname for it and he feels confident that it really is PayPal, and authorizes the transaction. Assuming that the e-commerce site is fraudulent, and redirects him to *www.paypa1.com* (note that it is “1”, not a small “L”) to phish him, his browser will not find a corresponding Petname because the domain name does not match. The missing Petname will alert him that the PayPal site is fraudulent, and that he should abort the transaction.

3.4.5 Anti-Phishing Tool

The Petname Model has been utilized effectively in the anti-phishing tool such as the Petname Tool [35], developed by Tyler Close; TrustBar [36], developed by the TrustBar team at the Dept. of Computer Science in the Bar Ilan University, Israel and Passpet [37], developed at the CYLAB of the Carnegie Mellon University. All of them are Firefox extensions and work only with the Firefox. The Petname Tool uses the hash of the public key of a website as the Pointer for that site. A user can assign a Petname for a website that is displayed in the browser when he visits it later. It does not work with non-HTTPS sites because it depends on certificate to retrieve the public key. Passpet extends the idea of the Petname tool also for non-HTTPS sites. It utilizes the combination of root_key, field_name and field_value to generate the Pointer. For the HTTPS sites, root_key is the hashed public key of the site,

field_name is "*O*" and field_value is the organization name if organization name is available in the certificate, otherwise field_name is "*CN*" and field_value is the certificate's common name. For the non- HTTPS sites, root_key is empty, field_name is "*D*" and field_value is the last n+1 level for the n-level TLD (Top level domain). Users can assign a Petname for each site by clicking an icon in the browser. The domain name represents the Pointer in case of TrustBar. The TrustBar displays the Petname for the domain name as well as the name of the certificate authority. It allows the user to enter a Petname if the visited site sends the server certificate for the first time and that Petname will be displayed when the site is revisited later. TrustBar also offers various options to manage Petnames.

3.4.6 IP Address

Not all IP addresses have domain names. If one would like to communicate only utilizing IP address, a Petname Model can be applied locally as a substitute for domain names. IP addresses are hard to remember, and Petnames will make it easy to refer to them. IP addresses will represent the Pointer, and the corresponding Petname will be used at the user interface. All communication from the user's side will be based on Petnames.

3.4.7 CapDesk and Polaris

CapDesk is a desktop environment that applies the principle of least authority and utilizes the Petname Model to provide security to the user for applications [38]. Whenever a new application is installed, CapDesk will feature a Pet Text and Pet Graphic for that application. The user may accept it or modify it. Once provided, Pet Text and Graphics will be used in the window of the application while it runs. Like CapDesk, Polaris is also based on the principle of least authority and also uses Pet Text similar to CapDesk and attaches it to the window of the application while it runs [39].

3.4.8 OpenPGP

The OpenPGP key is the Pointer and it carries the Nickname given by the owner of the Pointer. Some implementations of OpenPGP allow the user to change the Nickname and implement a Petname [2].

3.4.9 Process Handling

Every modern OS runs a number of processes simultaneously. *ps -e* command in Linux or the process tab in the task manager for Windows shows a long list of processes. Some of the process names are so obscure that it is impossible for the user to understand their functionality. A Petname Model can be applied to improve the situation significantly. When a process will run for the first time it will present a short description of what it will do. Then the user can create an informative Petname for that process. This Petname will be displayed in the memory map, for example in the process tab in task manager or with *ps -e* command. In this case the Pointer does not have to be global. It is simply the unique process name or unique command used to run the process.

Chapter 4

SECURITY USABILITY OF PETNAME SYSTEMS

The usability of security is one of the crucial elements to count for when assessing the overall security of the system, but is still a relatively poorly understood element of IT security and has drawn little attention until now. Many reported breaches of security have proved the fact that the breach occurred not because the security in technical sense was weak, but because of the gullible or unknowledgeable proper user behavior [40] and hence justly stated in [41]: "... *security is only as good as its weakest link, and people are the weakest link in the chain*". Therefore it is important to evaluate the Security Usability of Petname Systems as it is directly related to the security of client-side Identity Management.

4.1 Security Usability

Security Usability is a new multi-disciplinary approach that merges the expertise of several disciplines such as Computer Security, Human Computer Interaction, Behavioral Psychology and Social Science to design security techniques measures that are mathematically secure as well as practically usable. Being a new domain, its definition is still evolving. However a working definition can be found in [42]. The security of the software can be called usable if the users using it:

- are informed of the security tasks that they need to execute,
- can comprehend how to execute those tasks successfully,
- don't make fatal errors, and
- are reasonably comfortable interacting with the User Interface while using the software.

The major aim is to find a right balance between the usability and the security of an application. However, achieving the right balance has been proven to be a challenging task for the researcher and the developer. Tightening the security can degrade the usability of the system which in turn affects the overall security; on the other hand, a degraded usability can drastically affect the security of the system directly. Active research is going on to come across a proper trade-off.

4.2 Security Usability Principles

A set of general Security Usability principles related to Identity Management were proposed in [6]. We will use these principles as a basis to evaluate the Security Usability of the Petname System by analyzing if the Security Usability properties of the Petname System satisfy these principles. The Security Usability principles are described below.

4.2.1 Security Action Usability Principles

A1. Users must understand which security actions are required of them.
A2. Users must have sufficient knowledge and the ability to take the correct security action.
A3. The mental and physical load of a security action must be tolerable.
A4. The mental and physical load of making repeated security actions for any practical number of instances must be tolerable.

4.2.2 Security Conclusion Usability Principles

C1. Users must understand the security conclusion that is required for making an informed decision.
C2. The system must provide the user with sufficient information for deriving the security conclusion.
C3. The mental load of deriving the security conclusion must be tolerable.
C4. The mental load of deriving security conclusions for any practical number of instances must be tolerable.

4.3 Evaluation of Security Usability of Petname Systems

The Security Usability properties of Petname Systems that were discussed in subsection 3.3.2 can now be analyzed according to these security principles. When a Petname System satisfies SA1-SA3 and SA6-SA9 of the Security Action properties, it implicitly implies that principles A1 and A2 are also satisfied, because the former properties enable a user to select a unique and unambiguous Petname for a Pointer. This selection of a unique and unambiguous Petname for a Pointer can be thought of as the correct security action as it enables the user to securely identify an entity. Security Action properties SA4-SA8 will act as the aid for the user to select a Petname for a Pointer. We believe that selecting an unambiguous Petname will pose the most significant mental load for the user in the Petname System when repeated for several entities. Such mental load will be reduced significantly if these five properties are satisfied in a Petname System because users do not have to think about the ambiguity of the new Petname with other existing Petnames. Automated suggestion could also be a great aid in such selection. Therefore satisfying these five properties will implicitly lead to the principles A3 and A4 also being satisfied.

To analyze the Security Conclusion properties of the Petname System, we have to first define Security Conclusion in the Identity Management perspective. Security Conclusion in the Identity Management perspective is to correctly identify a specific entity. Displaying the Petname for a Pointer that points to the desired entity at the user interface will enable the user to draw conclusion that this Pointer and in turn the entity the user is interacting with is the intended one. The presence and absence of the Petname will provide the user with enough information to draw the security conclusion easily. So whenever a Petname System satisfies SC1-SC3, it will explicitly satisfy C1 and C2. Different visual techniques should be applied to help the user reduce their mental load in deriving security conclusion. Using different eye-catching colors to indicate the presence or absence of a Petname for a specific Pointer can be an example of one such visual technique. The security conclusion properties SC2-SC5 should be applied to enable a user to draw conclusion with ease and thus if followed will satisfy principles C3 and C4.

From the above analysis we can conclude that a complete implementation of all the properties of a Petname System will satisfy all the security usability principles.

4.4 Evaluation of Security Usability of Applications Based on Petname Model

Having formalized the properties of Petname Systems, and having analyzed security usability issues on a general level, the security usability for two of the existing Petname System applications are analyzed with the Cognitive Walkthrough method. The applications to be analyzed are: 1) Petname Tool and 2) TrustBar. Both toolbars are designed only to work with the Firefox browser, and are aimed at simplifying client-side management of SP identities and at providing a better defense mechanism against Phishing attacks. Though the application domains for the Petname System is much broader, as described in Section 3.4, we have decided to confine our evaluation only to these two in order to focus on managing SP identities at the client side. These two particular applications exactly meet this criterion.

The Cognitive Walkthrough method is a usability evaluation method in which an evaluator or a group of evaluators participate to identify the usability issues of an application by visually inspecting the user interface [43]. It focuses on evaluating the understandability and the ease of use for a user at the user interface to accomplish a task using that application. Among several usability evaluation methods we have chosen the Cognitive Walkthrough as our preferred method because of its main focus on the understandability of the user at the user interface [43]. Because Petname Systems affect the user interface, Cognitive Walkthrough is a suitable method for evaluating their usability. While performing the Cognitive Walkthrough for each application, we will try to note if the application satisfies the usability properties discussed in 3.3.2. The degree of compliance with the specified security usability properties will give an indication of the level of security usability of each application. For the evaluation we will be using Firefox version 3.0.7 with Nightly Tester Tool, a Firefox add-on, installed.

4.4.1 The Petname Tool

Setup: The Petname Tool is available as a Firefox add-on in [44]. The current version of the Petname Tool is compatible with the latest Firefox version, and can be easily installed by just

clicking the "*Add to Firefox*" button in the respective Firefox Add-on website. Once installed the toolbar will look like Figure 4.1.

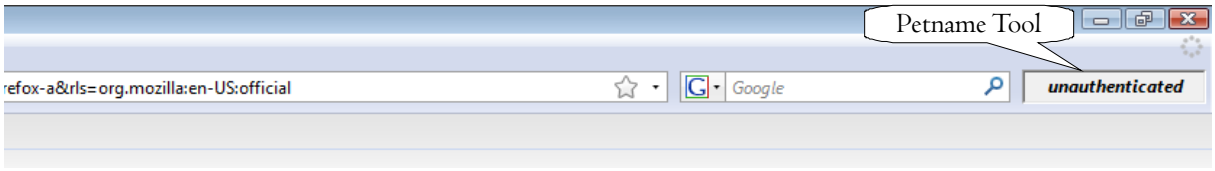

Figure 4.1: The Petname Tool in Firefox

Functionality: The first thing to note about the Petname Tool is its simplicity. It consists of only a text field in the navigation toolbar of the browser. Its main purpose is to allow a user to assign a Petname for a website that he wants to correctly recognize and to display that Petname in the text field when he visits the site later. The Petname will be absent if the visited site is not the intended one. A user can judge if a webpage comes from a previously identified website by observing the presence or absence of the Petname. The Petname Tool utilizes different font properties and graphical user interface elements to achieve its goal: 1) The text in the text field, 2) The typeface of the text, 3) Color of the text field, 4) Tooltip and 5) Dialog box. Different texts with different typefaces are displayed in the text field in different situations, color of the text field changes, different tooltips are provided accordingly when mouse pointer is placed over the text field and warnings are displayed using dialog boxes.

Some examples can illustrate how the Petname Tool operates. It is worth noticing here that the Petname Tool does not work for non-HTTPS sites, as it uses the hash of the public key of a website as the Pointer for that site. While visiting a non-HTTPS site, e.g. *www.wikipedia.org*, the text in the text field will be *unauthenticated* with italic typeface and it will be disabled with grey color so that nobody can assign a Petname for the site (Figure 4.2). The corresponding tooltip is: *Don't give this page sensitive information; it was not received*

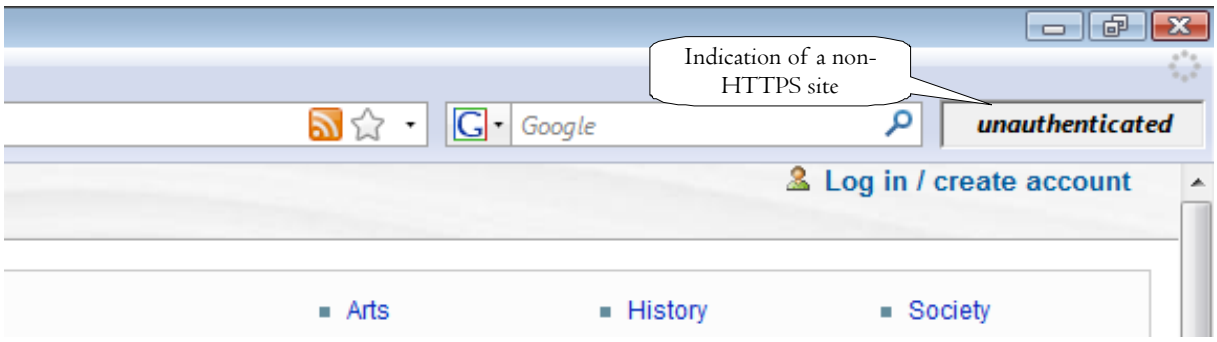

Figure 4.2: Disabled text field for a non-HTTPS site

securely (Figure 4.3). During the visit to an HTTPS site for the first time, e.g. *www.paypal.com*, the text in the text field becomes *unknown site* with italic typeface and the text field color changes to white **(**Figure 4.4**)** with the corresponding tooltip: *Assign a Petname to this site before exchanging sensitive information* **(**Figure 4.5). At this point, user can assign a Petname by just writing it in the text field and hitting the Enter key. The color of the text box changes from white to light green and type face becomes normal (Figure 4.6). When the user visits that site later, the Petname with normal typeface is displayed in the green

text field. Different dialog boxes are prompted to warn users whenever something goes wrong.

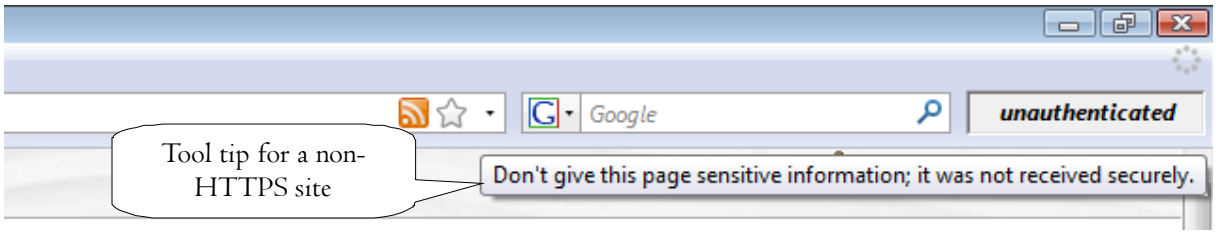

Figure 4.3: Tooltip for a non-HTTPS site

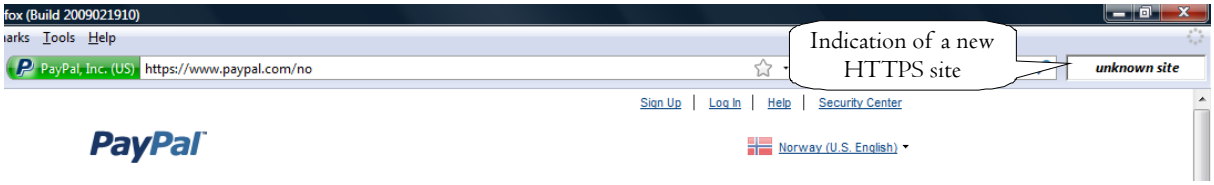

Figure 4.4: Indication of an HTTPS site

Figure 4.5: Tooltip for an HTTPS site

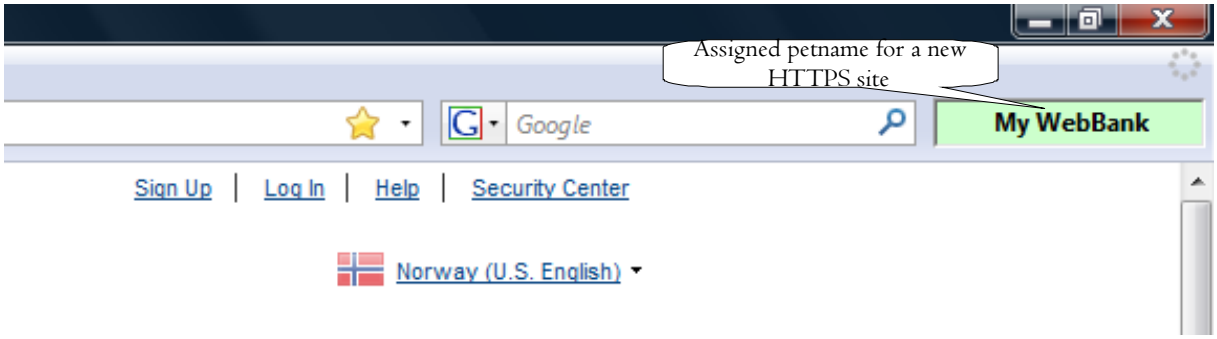

Figure 4.6: Assigned Petname for a new HTTPS site

Evaluation: As mentioned earlier, the Petname Tool is very simple; however, one may almost feel that it is too simple. It does not come with any text label; only a text field to enter Petnames. Absence of a text label can confuse unfamiliar users because they might not understand its purpose. The Petname Tool does not work for non-HTTPS sites, therefore it will not be possible for a user to assign Petnames for non-HTTPS sites. Many do not use HTTPS in the initial log-in stages, though the log-in name and the related password may be encrypted before transmission. An example is the famous social networking website

www.facebook.com. A potential vulnerability is caused by Facebook because email addresses represent user names. People often use the same passwords for different accounts, so a password used on Facebook will often allow access to the user's web email account. Therefore we think that the lack of support for non-HTTPS sites in the Petname Tool is a major drawback. Another thing is worth to note that the Petname Tool uses the hash of the public key of a website as a Pointer. Therefore if the site receives a new certificate and thus a new public key, the Petname Tool will fail to map between the already assigned Petname and the Pointer. A possible solution could be to let URL or domain name be the Pointer that will also remove the restriction of applying Petnames for HTTPS sites only.

In the following, the Petname Tool will be analyzed for compliance with the Petname System properties. The Petname Tool, obviously, deploys Petnames. The hash of the public key, derived from the certificate, represents the Pointer and is strongly resistant against forgery. Therefore we conclude that the Petname Tool satisfies F1 and F3. But a serious restriction of the Petname Tool is that it allows users to assign exactly the same Petname for different entities as demonstrated in the next paragraph, thus violates the principle of bi-directional one-to-one mapping for each entity and therefore also violates F4. It does not deploy Nicknames and therefore does not satisfy the optional property F2.

The Petname Tool enables users to explicitly assign a Petname for each entity, e.g. to select the text field, write down a Petname and hit the Enter key. This satisfies SA1 and SA2. Users can change any Petname any time, thereby satisfying SA3. No suggestion is provided for aiding the user to select a Petname, which is not compliant with SA4 and SA5. Also Nicknames are not used in the Petname Tool, resulting in non-compliance with SA6. Whenever a user selects a Petname that closely resembles existing Petnames, the user is alerted with an informative dialog box **(**Figure 4.7). The dialog box displays the existing Petnames to which the current Petname has close resemblance. The user can ignore the alert by clicking the *"Assign petname*" button or he can cancel this current Petname by clicking the *"Don't assign petname"* button. If a user assigns a Petname that is similar to an existing Petname, the Petname Tool displays another dialog box **(**Figure 4.8). The second dialog box contains the name of the existing similar Petname with its creation date. The user has the option to discard the current Petname by clicking the *"Don't assign petname"* button. If the user clicks the *"Assign petname"* button, the Petname will be assigned for the current entity. In this case, the same Petname will be displayed for both websites when he visits them later. Therefore, the Petname Tool is compliant with SA8 (showing the two dialog boxes with the warning), but directly violates SA7. Another thing to note is that two dialog boxes have the same title, though their purposes are quite different. The Petname Tool allows a user to assign a Petname at his will whenever he feels and does not show any alert when there is highly sensitive data transmission and therefore indicates the absence of SA9.

The Petname, if already supplied by the user, is displayed on the Petname Tool toolbar, thereby satisfying SC1. Different typefaces, tooltips and colors have been used in the Petname Tool to catch the user attention to indicate the presence or absence of a Petname. In our opinion, white and light green as used by the Petname Tool is less visible than Red, Yellow or Green, as suggested in [45]. In addition, blinking text or different text colors could be used to draw more user attention. Nevertheless, we can conclude that the Petname Tool is compliant with SC2 and SC3. As there is no suggested Petname or Nickname in the Petname Tool, it does not satisfy SC4. The Petname Tool provides warning through dialog boxes when there are conflicts with other Petnames or if there is ambiguity between Petnames and thus satisfies

SC5. However, it does not provide a warning message when there is a violation for other properties.

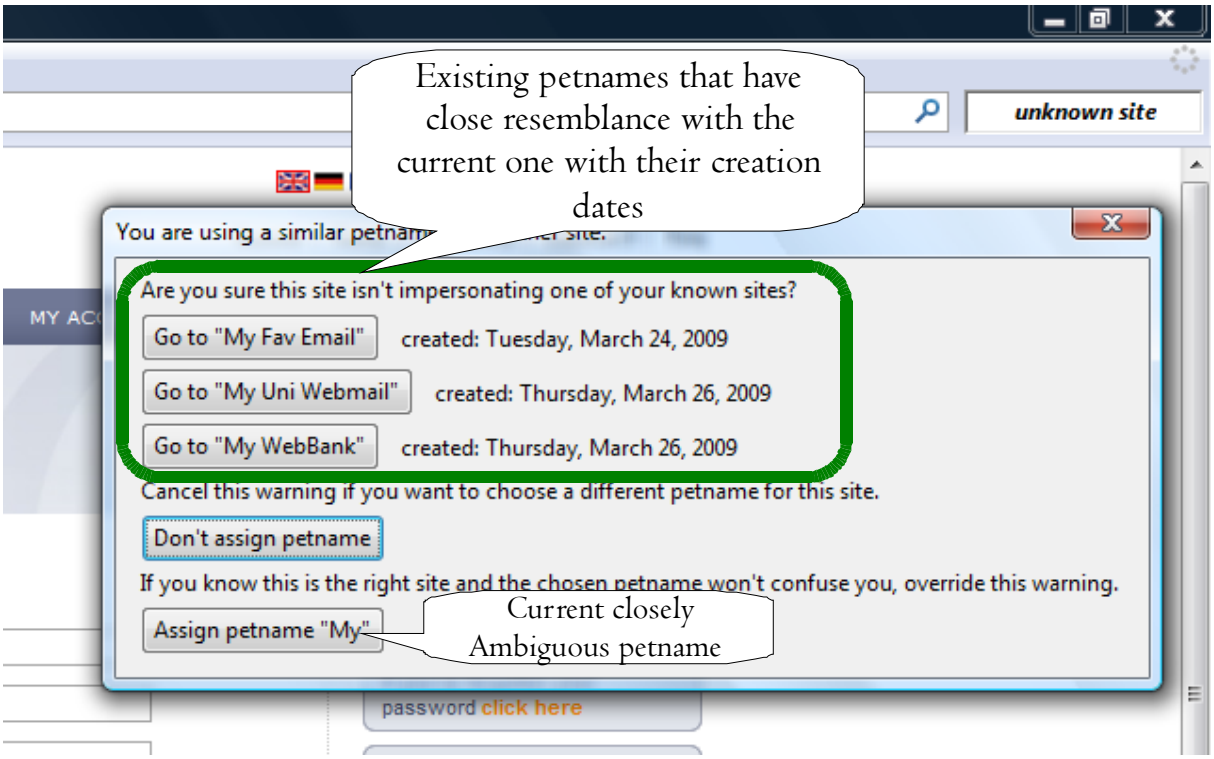

Figure 4.7: Dialog box warning about the close ambiguity among different Petnames

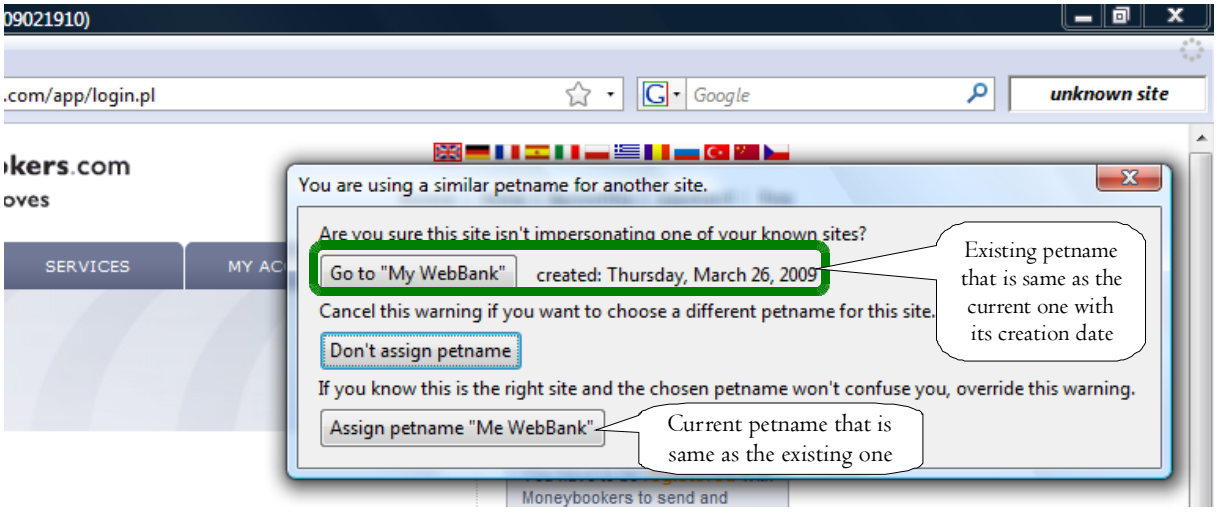

Figure 4.8: Dialog box warning about the similarity between two Petnames

Apart from security usability issues, there are some other weaknesses in the Petname Tool. For example, there is no help button that could explain what the user has to do to utilize it properly. It does not provide the standard *About* or *Help* menu item that could explain the purpose of the Petname Tool.

4.4.2 TrustBar

Setup: The TrustBar Tool is available as a Firefox add-on in [36]. The current version of TrustBar is not compatible with the latest Firefox version. Therefore the Nightly Tester Tool, another Firefox add-on, was used to resolve the compatibility issues. Once installed the toolbar looks like Figure 4.9.

Figure 4.9: TrustBar installed in Firefox

Functionality: TrustBar consists of a text field, a menu and a Security Status field. The text field allows users to enter a Petname for an entity, the menu provides the user with options, and the Security Status field provides visual indication of various security statuses based on the certificate. Unlike the Petname Tool, it also allows user to assign a logo that represents as a Petname for an entity. When a logo is used, an image replaces the text field and such logos can be called Petlogos. A user can assign a Petname text or Petlogo for a website that he wants to correctly recognize and to display that Petname or Petlogo when he visits the site later. The Petname or Petlogo will be absent if the visited site is not the intended one. A user can judge if a webpage comes from a previously identified website by observing the presence or absence of the Petname or Petlogo. TrustBar utilizes different graphical user interface elements to achieve its goal: 1) The Petname field for text or logo, 2) Color of the Petname field, 3) Drop down menu, 4) Tooltip and 5) Security Status field. The Petname field keeps changing as a user visits different sites. At the same time the color of the Petname field changes and different tooltips over the Security Status field are provided. The Security Status field provides a visual indication of the status of the server certificate, and changes according to different circumstances. Options in the menu allows users to set the Petname or Petlogo, to edit the Petname or Petlogo, remove the defined Petname(s), report fraudulent websites and display help regarding TrustBar (Figure 4.10). The menu also contains an *About* menu item that, if clicked, displays some relevant information regarding TrustBar, e.g. what is TrustBar, why is it used for, etc.

Some examples can illustrate the TrustBar functionality. Unlike the Petname Tool, TrustBar works both with HTTPS and non-HTTPS sites. When users visit a non-HTTPS site, e.g. *www.wikipedia.org*, the Petname field contains the domain name for that site (Figure 4.11). A user can assign a Petname by writing directly in the text box and hitting the enter key. The color of the text field will turn from white to light green. The Security Status filed displays a *No lock* icon indicating that the site does not have a server certificate and that TLS is not used, and also provides the tooltip: "*This site is not protected. Click here for more information*" (Figure 4.12). Clicking the tooltip will redirect the user to the TrustBar website that explains the necessary concept on TrustBar. A user can edit the Petname later just by writing the new one and hitting the enter key. The drop-down menu also provides methods to assign, edit or delete Petnames. Assigning and editing a Petlogo happens in a similar way, except that the user has to select an image from his computer. When the user visits an HTTPS site, e.g. *mail.yahoo.com*, the text field contains the organization name from the certificate. The color

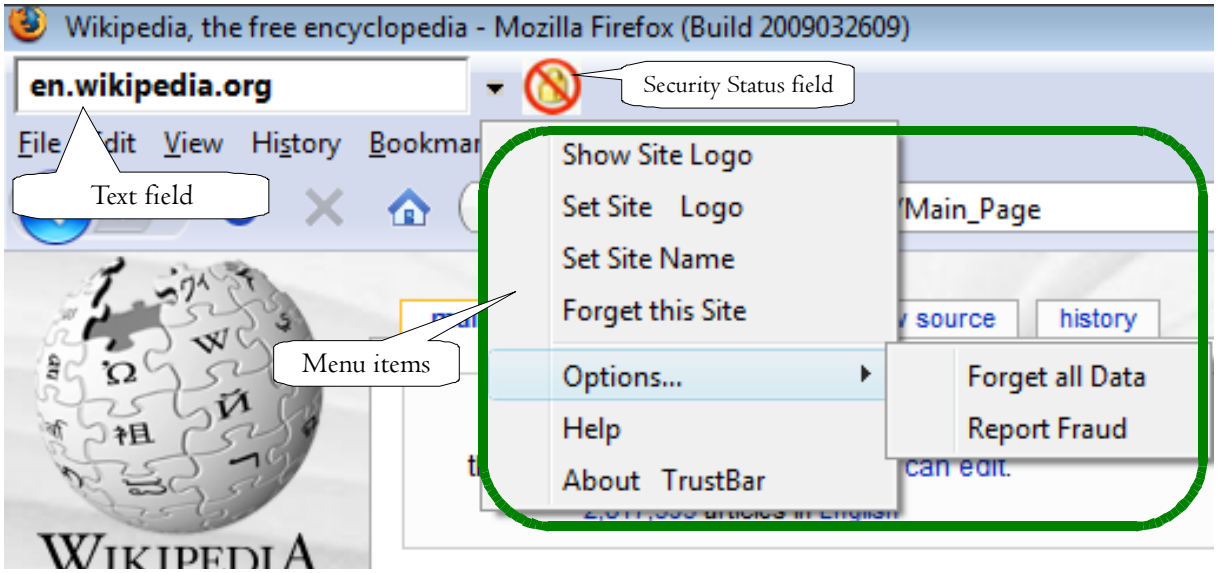

Figure 4.10: Components of TrustBar

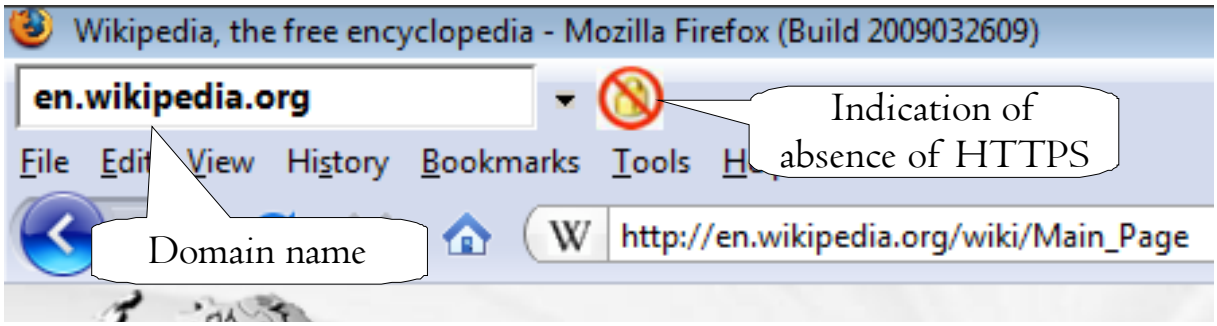

Figure 4.11: Indication of a non-HTTPS site in TrustBar

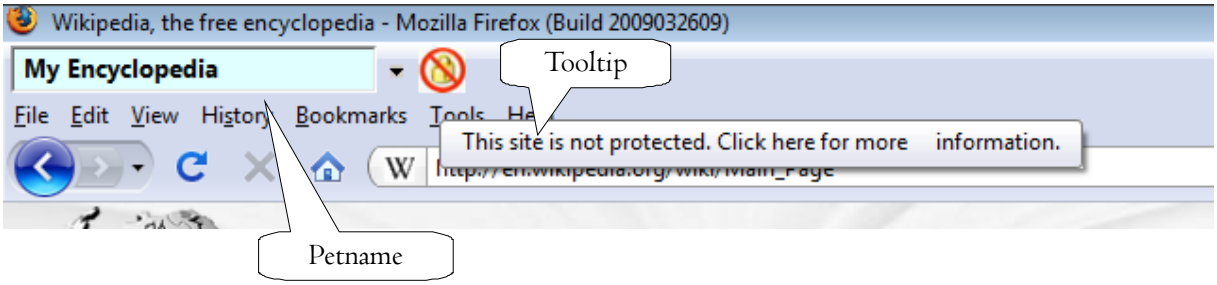

Figure 4.12: Assigning a Petname for a non-HTTPS site in TrustBar

of the text field turns to pale yellow. The Security Status field is modified with a lock icon and the text *Identified By*. The name of the CA and another drop-down menu are displayed adjacent to the Security Status field. This second menu allows the user to set, edit or delete a logo for CA, to ignore the CA, and some other options **(**Figure 4.13**)**. The user can assign a Petname or Petlogo to override the organization name like before. Once a Petname is assigned, the Petname field turns to light green **(**Figure 4.14).

Evaluation: TrustBar overcomes some of the shortcomings of the Petname Tool. For example, it works for non-HTTPS sites, provides an excellent *Help* feature and also comes with the standard *About* menu item that provides a short description on what it does.

The following simple analysis of TrustBar gives an indication of how it satisfies the properties of the Petname Model. TrustBar utilizes Petnames, and thereby complies with F1. The domain name or URL represents the Pointer and is strongly resistant against forgery. Therefore we conclude that TrustBar satisfies F1 and F3. TrustBar also displays a Nickname in the form of the organization name, if a certificate is available or in the form of the domain name for non-HTTPS sites and thus satisfies F2. However, a serious restriction of TrustBar is that it allows users to assign exactly the same Petname for different entities as demonstrated in the next paragraph, thus violates the principle of bi-directional one-to-one mapping for each entity and therefore violates F4.

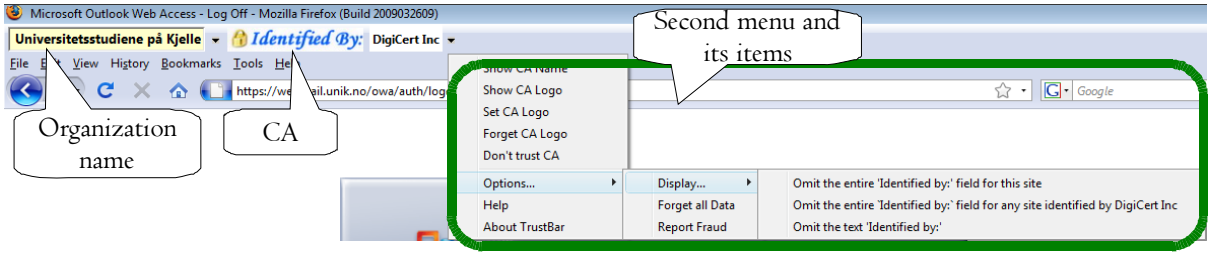

Figure 4.13: TrustBar interaction with an HTTPS site in TrustBar

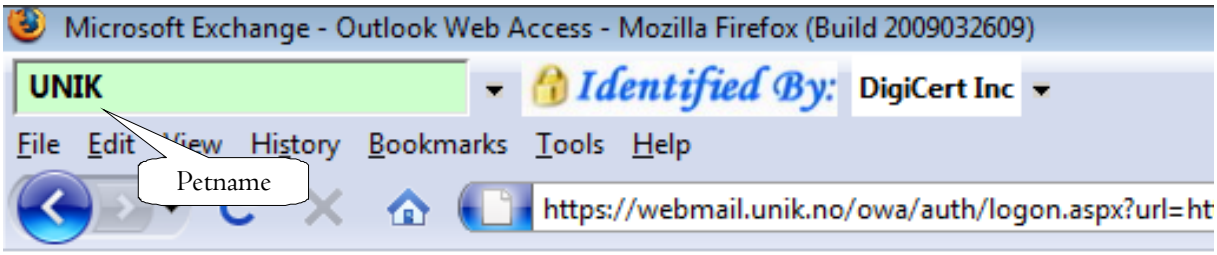

Figure 4.14: Assigning a Petname for an HTTPS site in TrustBar

TrustBar enables a user to assign a Petname for each entity so he has to act explicitly, e.g. select the text field, write down a Petname and hit the Enter key, to enable the Petname and this satisfies SA1 and SA2. Users can change any Petname any time and thus TrustBar meets the requirement of SA3. A suggestion is provided in the form of a Nickname for aiding the user to select a Petname if a server certificate is available and this satisfies SA4 partially, and the user has to act explicitly, e.g. by hitting the Enter key so the text field turns to light green (an indication for accepting the Petname) to accept the Nickname as the Petname. This satisfies SA5 too. However, it is important to note here that if the Nickname is accepted as the Petname without any modification then it represents a temporary Petname in TrustBar, because when the user visits it again, the Petname turns into the Nickname, also indicated by the pale yellow color of the text field. This means that TrustBar tries to ensure S6. But we feel that this approach is rather contradictory and think that a better approach could be taken that would not allow users to accept the Nickname as the Petname without any modification. As mentioned earlier, a serious restriction of TrustBar is that it allows users to assign ambiguous Petnames or even equal Petnames for different entities. It does not provide any sort of

warning to users about the ambiguity or similarity of the Petnames and thus directly violates SA7 and SA8. TrustBar allows a user to assign a Petname at his will whenever he feels and does not show any alert when there is highly sensitive data transmission and therefore violates SA9.

The Pointer and the related Petname in the TrustBar, if already supplied by the user, are displayed all the time in the browser toolbar, thereby satisfying SC1. Different icons, tooltips and colors have been used in the TrustBar to catch the user attention to indicate the presence or absence of Petnames. In our opinion it would have been better to use more flashy colors like Red, Yellow or Green instead of pale yellow and light green. Blinking text or different text colors could be used to draw more user attention to potential security problems in websites. Nevertheless, we can conclude that TrustBar satisfies SC2 and SC3. White, pale yellow or light green color has been used to differentiate among non-HTTPS Nicknames, HTTPS Nicknames and Petnames respectively, thereby satisfying SC4. TrustBar does not provide any sort of warning to the user and this indicates the complete absence of SC5.

4.4.3 Summary

Table 4.1 clarifies the distinction between the Petname Tool and TrustBar in terms of the properties of Petname Systems.

Tool Name	F				SA									SC				
	1	2	3	4	1	2	3	4	5	6	7	8	9	1	2	3	4	5
Petname Tool	Y	N	Y	N	Y	Y	Y	N	N	N	N	Y	N	Y	Y	Y	N	Y
TrustBar	Y	Y	Y	N	Y	Y	Y	Y	Y	Y	N	N	N	Y	Y	Y	Y	N

Table 4.1: Comparison between the Petname Tool and TrustBar

It can be noted that TrustBar satisfies more properties of the Petname model than the Petname Tool, though TrustBar has one major shortcoming: absence of any type of warning message. Both tools suffer from the absence of the crucial property F4. As none of them satisfies all the main properties of Petname Systems, we can conclude that none of them fully satisfies the Security Usability principles.

Chapter 5

DEVELOPMENT

Our analysis in the previous chapter revealed that the currently available applications that utilize the Petname Model for managing SP Identities on the client side do not satisfy all the properties of the Petname Model and thus are not compliant with the Security Usability principles, therefore, we opt to develop an application based on the Petname Model that serves dual purposes: 1) it acts as an aid for client side management of SP identities and thus provides a defense mechanism against Phishing attacks like the two Firefox extensions discussed in the previous chapter and 2) it satisfies all the properties of the Petname System and thus is compliant with the Security Usability principles.

5.1 UniPet

Our developed application will be called *UniPet* that originates from the analogy that it was developed at the University Graduate Center in Kjeller (UNIK), Norway and utilizes the Petname Model to serve its purposes. The application was named so by taking the first three characters '*UNI*' from UNIK and the three characters '*Pet*' from the word '*Petname*'. It has been developed as an extension for the Firefox browser. It supports Petnames as well as Petlogos and has several other features that will be discussed in subsequent sections. Its purpose will be similar to the other two extensions (The Petname Tool and TrustBar) discussed in the previous chapter, however it will provide better functionalities and improved experience than those two.

5.2 UniPet Architecture

In this section we will present the higher level architecture of the UniPet to explain its internal components and how UniPet interacts with different components of the browser to accomplish its task.

5.2.1 Architecture

The UniPet has been integrated within the Firefox browser and interacts with different components and services of the browser. Components are the address bar and the menu

bar whereas the services include: progress listener service, bookmark service, preference service and local file service. The UniPet obtains the domain name of the webpage from the address bar and places itself at the menu bar and updates its outlook and behavior at different situations. It utilizes the progress listener service of the browser to obtain the update on webpage loading and to obtain the security status of the webpage. The UniPet uses the bookmark service to save Petname, preference service to save user preference and local file service to create and delete files in local system for various purposes e.g. saving the Petlogos. Figure 5.1 illustrates the higher level interaction of the UniPet with the browser and with its different components and services.

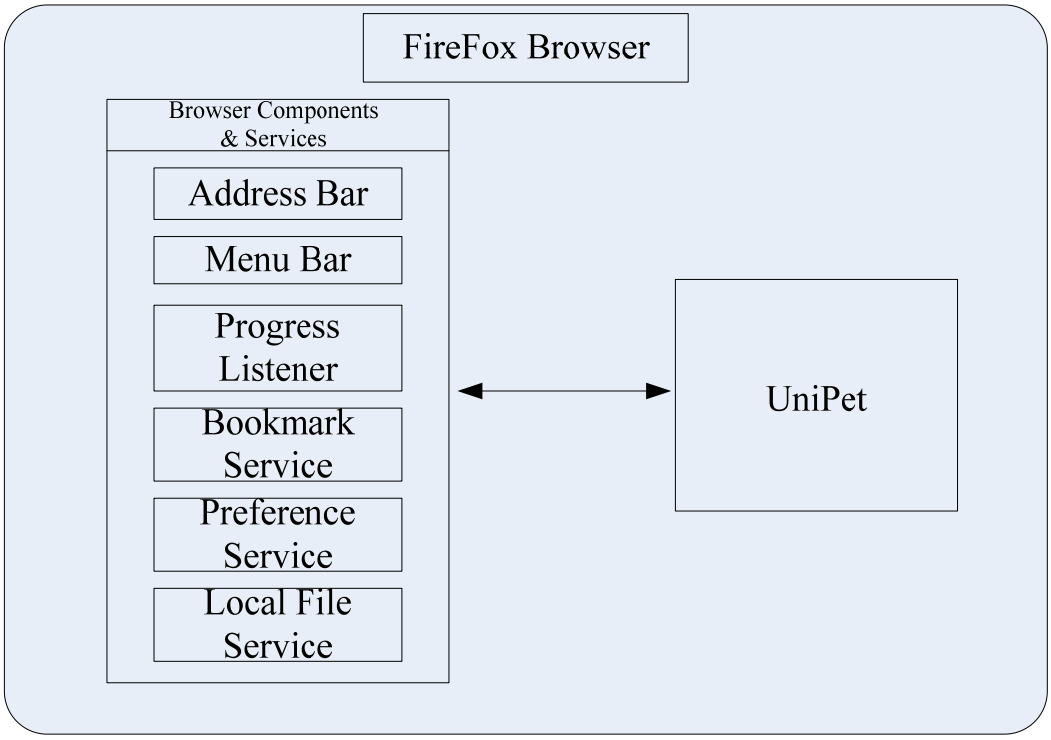

Figure 5.1: Higher level interaction between the UniPet and the Firefox browser

The UniPet itself has several components that interact within themselves to achieve the general goal. Figure 5.2 illustrates the GUI components of the UniPet that include: text labels, buttons, text box, menu and menu items. Within the menu there are several menu items. We will present the screen shots of the UniPet at the next chapter and only the architecture will be presented here. Detailed behavior of the components will be described also in the next chapter.

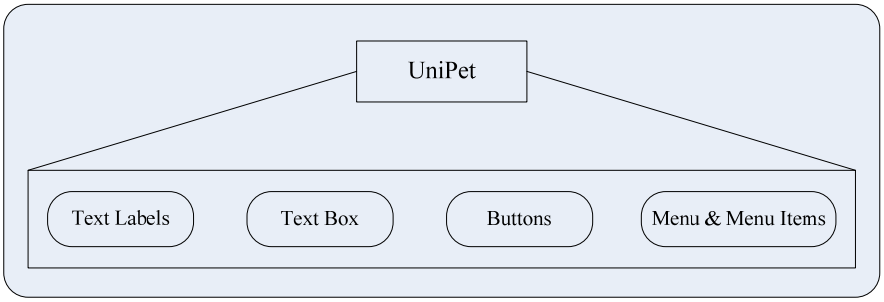

Figure 5.2: UniPet GUI Components

5.2.2 Interaction

When designing a browser extension that enables the user to assign a Petname or a Petlogo for a website and changes its behavior and user-interface as the user visits different sites, it is necessary to integrate the extension with the components of the browser and utilize the internal services that the browser provides to its components. Web browser is a complex reactive application. It has several components that work seamlessly to provide its services to the user. Therefore when we try to model the interaction between the UniPet and the different components and services of the browser, it becomes extremely complex. Message Sequence Chart (MSC) is well suited to present such complex interaction diagram in a convenient way [46]. We will use MSC to present the overall interaction of the UniPet with the browser components and services. Then we will use the Specification and Description Language (SDL) [47, 48] to define the reactive and real-time behavior of the UniPet itself. It is important to note here that both for MSC and SDL, we'll omit many formal definitions method (e.g. there will be no formal definition for signals, gates or other entities, etc.) and only concentrate on the core diagram that helps to better understand the interaction of the UniPet. Another thing to note here is that there are many more components and services available in the browser other than those used by the UniPet and presented here.

Figure 5.3 is the first one in a series of MSCs to explain the interaction of the UniPet.

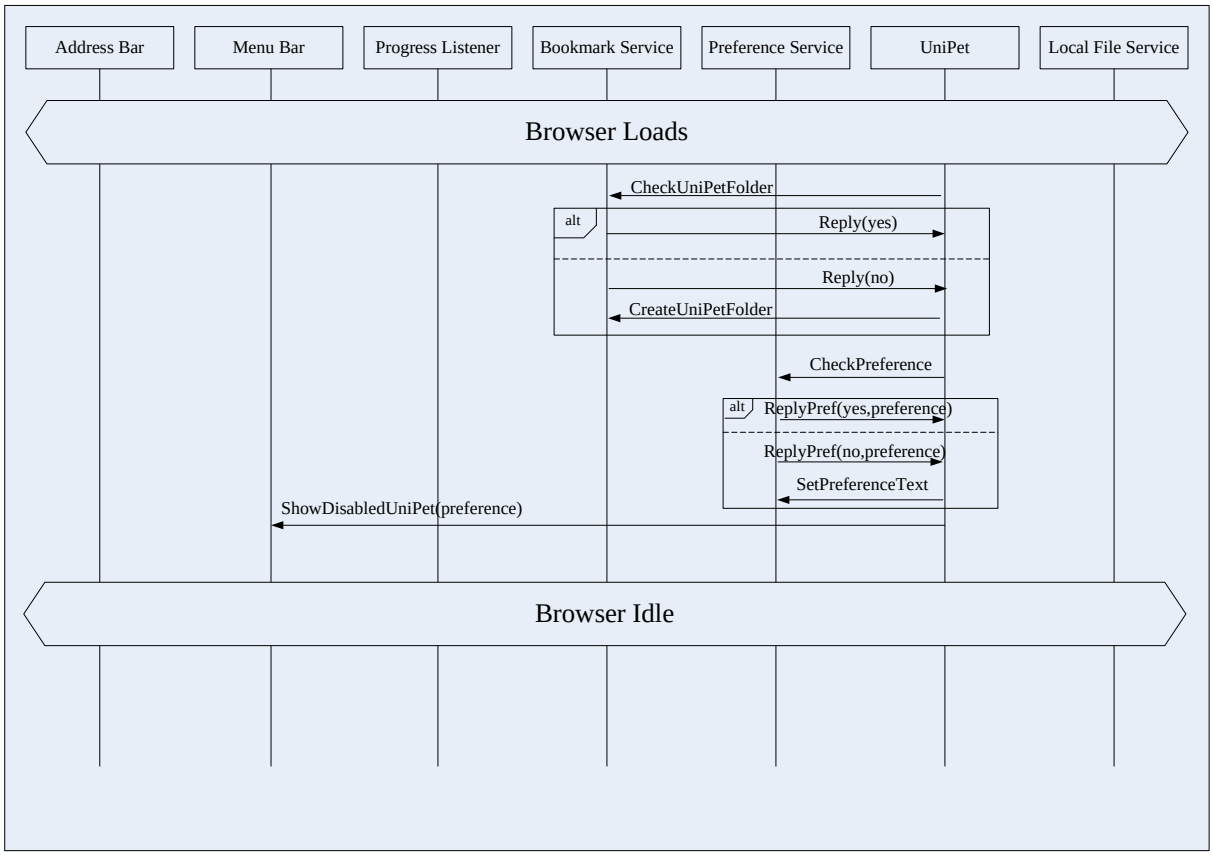

Figure 5.3: MSC-1: Interaction of the UniPet with the browser components and services

When the browser loads, the UniPet will attach itself to the menu bar of the browser and checks if there is a special folder in the bookmark of the browser that UniPet uses to save the

Petname. The UniPet uses the bookmark service provided by the Firefox for accessing the bookmark. It also checks if there is a specific file in the local file system that will be used to store the location of the image for using it as a Petlogo. If the folder or the file not found, the folder will be created in the bookmark and the file will be created in the local file system. Then the UniPet checks if there is a specific preference variable in the preference repository of the browser. As said earlier, a user can use Petname or Petlogo service of the UniPet. However, a user can select any one service at any time and this variable is used to store the user preference on Petname or Petlogo service and is used to dynamically change the user-interface of the UniPet. Then, as the browser loads without any webpage to display, some parts of the UniPet are disabled. The browser remains idle at this point and waits for any user command.

The user has two options at this point: either he can create a new tab or he can write a web address in the address bar. Let's illustrate the interaction when a user creates a new tab (Figure 5.4). When a new tab is created while the browser is already loaded, UniPet checks

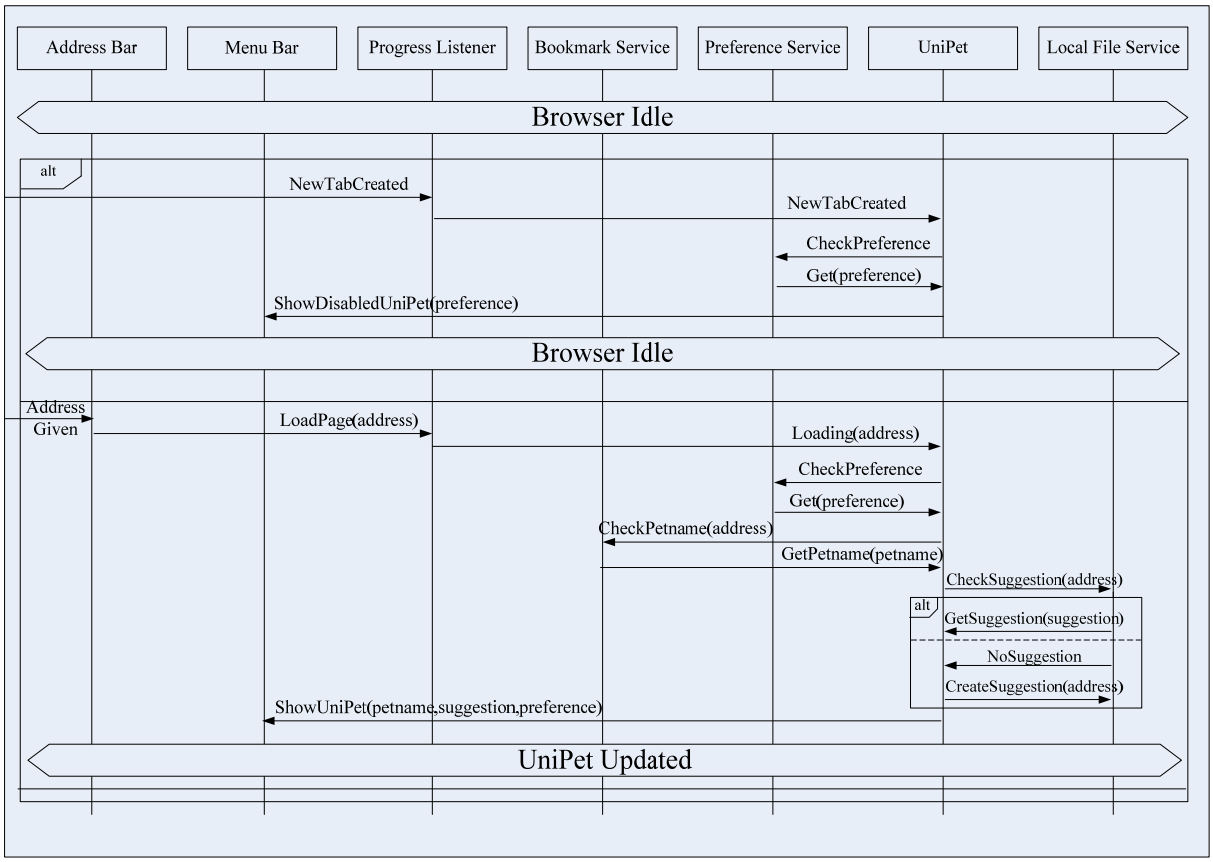

Figure 5.4: MSC-2: Interaction of the UniPet when a new tab is created or a web address is given

the preference of the user, e.g. either the UniPet is in the Petname mode or in the Petlogo mode and based on this preference, the UniPet is updated in disabled mode and the browser remains in the idle mode waiting for the user command.

When the user writes a web address and hits the enter button, the UniPet is informed of the progress of the web page loading by the progress listener service (Figure 5.4). It starts

working by looking at the preference of the user and changes the user-interface accordingly. Then it checks if there is any Petname or Petlogo available for the respective domain name. The UniPet also provides suggestion to the user using the title of the webpage. The UniPet checks a specific file to ensure if there is any suggestion for the address. If found, it is retrieved; otherwise a new suggestion is created and saved to that file. Once the Petname or Petlogo and the suggestion are retrieved, they are then displayed in the UniPet. At this point, the UniPet and the all other elements are in the "*UniPet Updated*" mode.

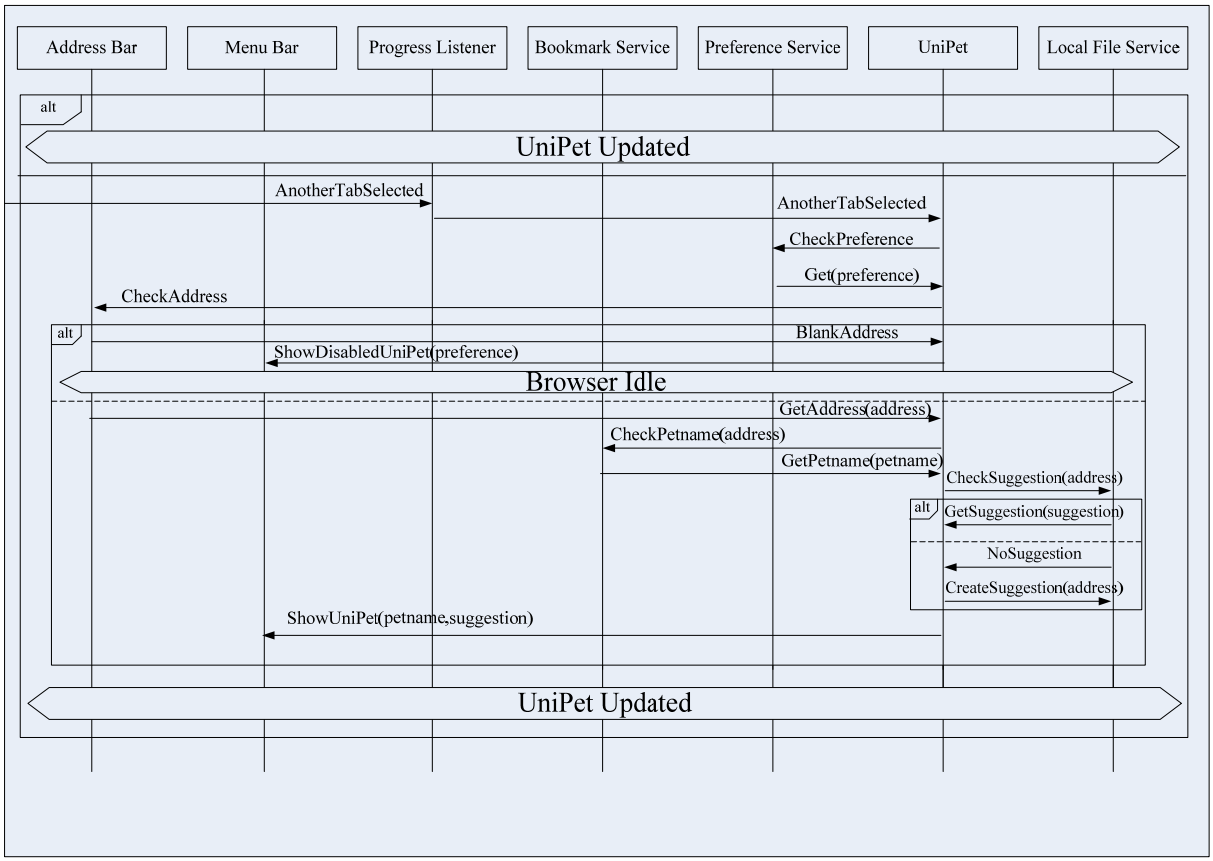

Figure 5.5: MSC-3: Interaction of the UniPet when an existing tab is selected

Once the user starts browsing and creates a few tabs to enable tabbed browsing, then he may select one of these existing tabs. Figure 5.5 shows the interaction at this situation. The UniPet is informed that a tab has been selected and it starts working by looking at the preference of the user and changes the user-interface accordingly. Then it checks the web address from the address bar. It may happen that the address bar is blank. In this case, the UniPet is updated in disabled mode and the browser remains in the idle mode waiting for the user command. If there is an address in the address bar, then it checks if there is any Petname or Petlogo available for the respective domain name. The suggestion for the domain is retrieved or created in the same way discussed previously. Once the Petname or Petlogo and the suggestion are retrieved, they are then displayed in the UniPet. At this point, the UniPet and the all other elements are in the "*UniPet Updated*" mode.

While the browser is loaded and it is in any of the modes discussed previously, a user can save a Petname or Petlogo (according to the UniPet preference) at any time (Figure 5.6). If he does so, the UniPet will check the address of the address bar. If it is blank, a message will be

shown to the user. If there is any address, then the Petname will be saved against the domain name to the UniPet folder of bookmark service or the location of the Petlogo will be saved to a specific local file in the system. Once it is saved the UniPet will remain in the same mode as it was before.

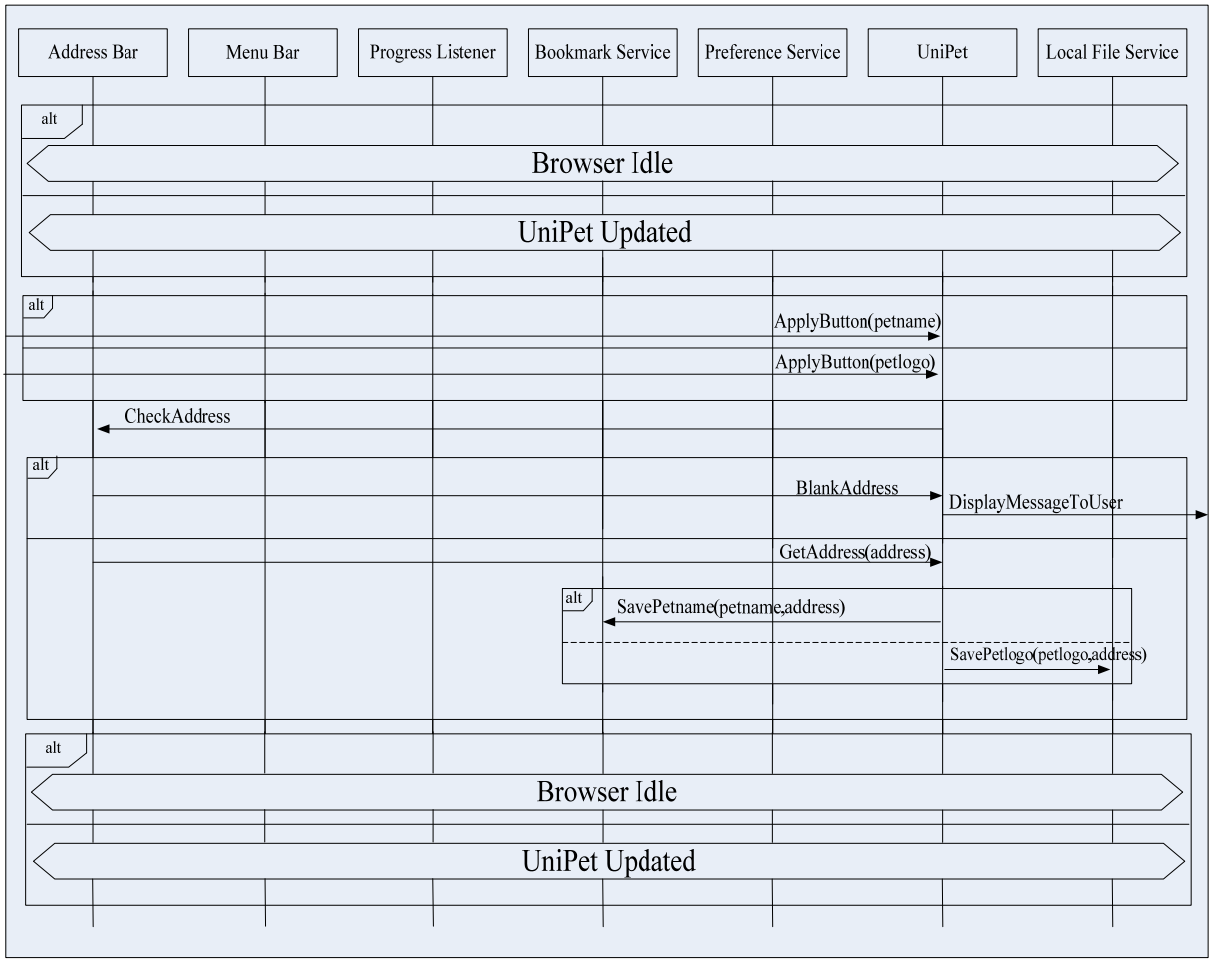

Figure 5.6: MSC-4: Interaction of the UniPet when a Petname or Petlogo is saved

Users can delete all the Petnames or Petlogos at any time. If such action is performed (Figure 5.7), all the Petnames from the UniPet folder of the bookmark service are deleted and all the file contents that have the location of the Petlogos from a specific file are deleted. There is an extensive help page that explains the functionality of the UniPet. Users can read that page by choosing the "*Display Help*" option. Other than this, users can choose "*Display About*" option and a dialog box will be displayed that will briefly state the purpose of the UniPet. Users can also switch back and forth seamlessly between the Petname or Petlogo mode and the UniPet will be updated according to the user preference.

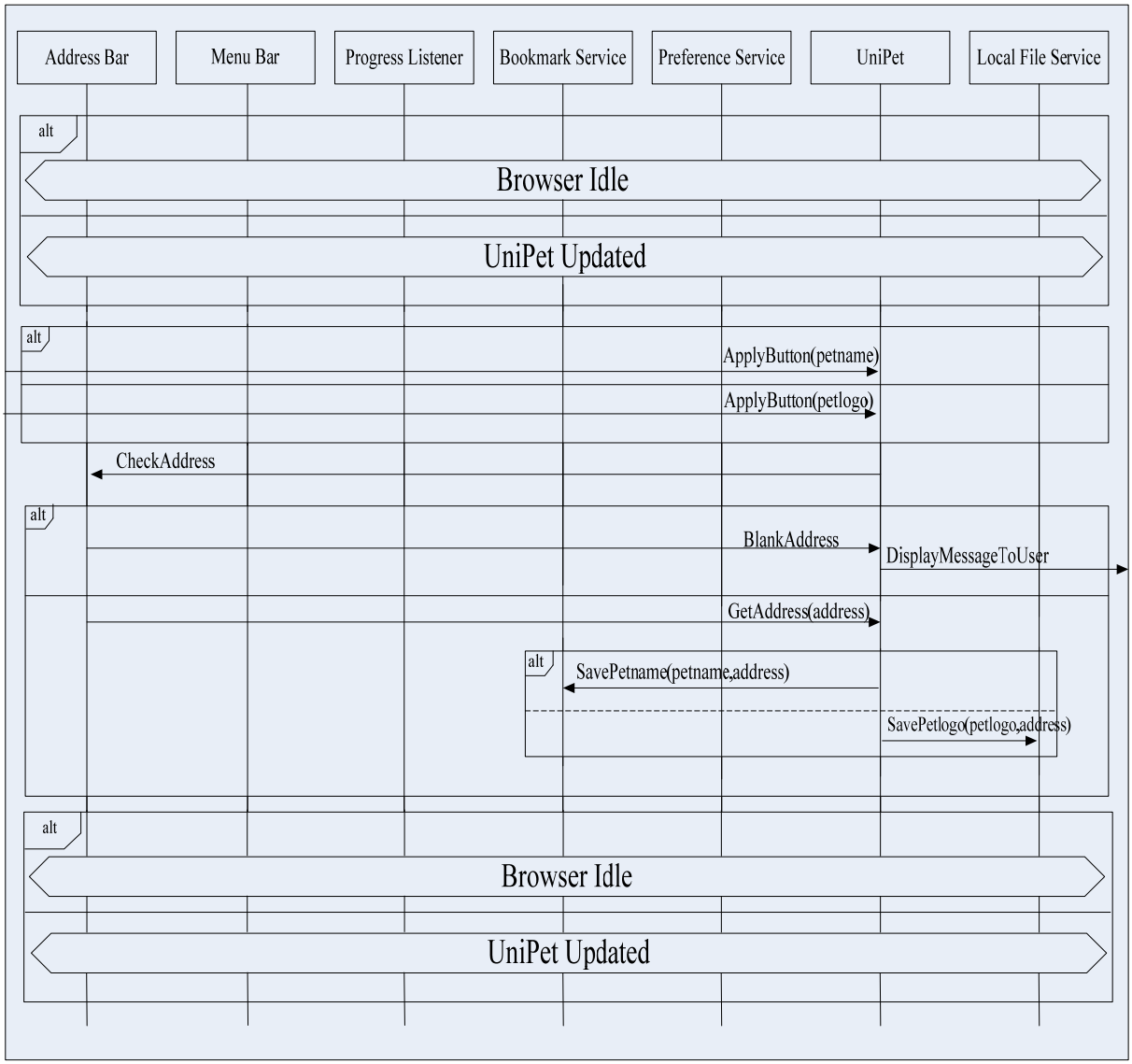

Figure 5.7: MSC-5: Interaction of the UniPet when other options of the UniPet are selected

In the next few pages we will present the SDL diagrams of the UniPet process. These diagrams define the behavior of the UniPet solely and explain its reactivity in different situations. There will be no explanation for these diagrams as we believe that these diagrams follow the same line of reasoning that were discussed in the previous paragraphs in details and there should be very less difficulties understanding them properly.

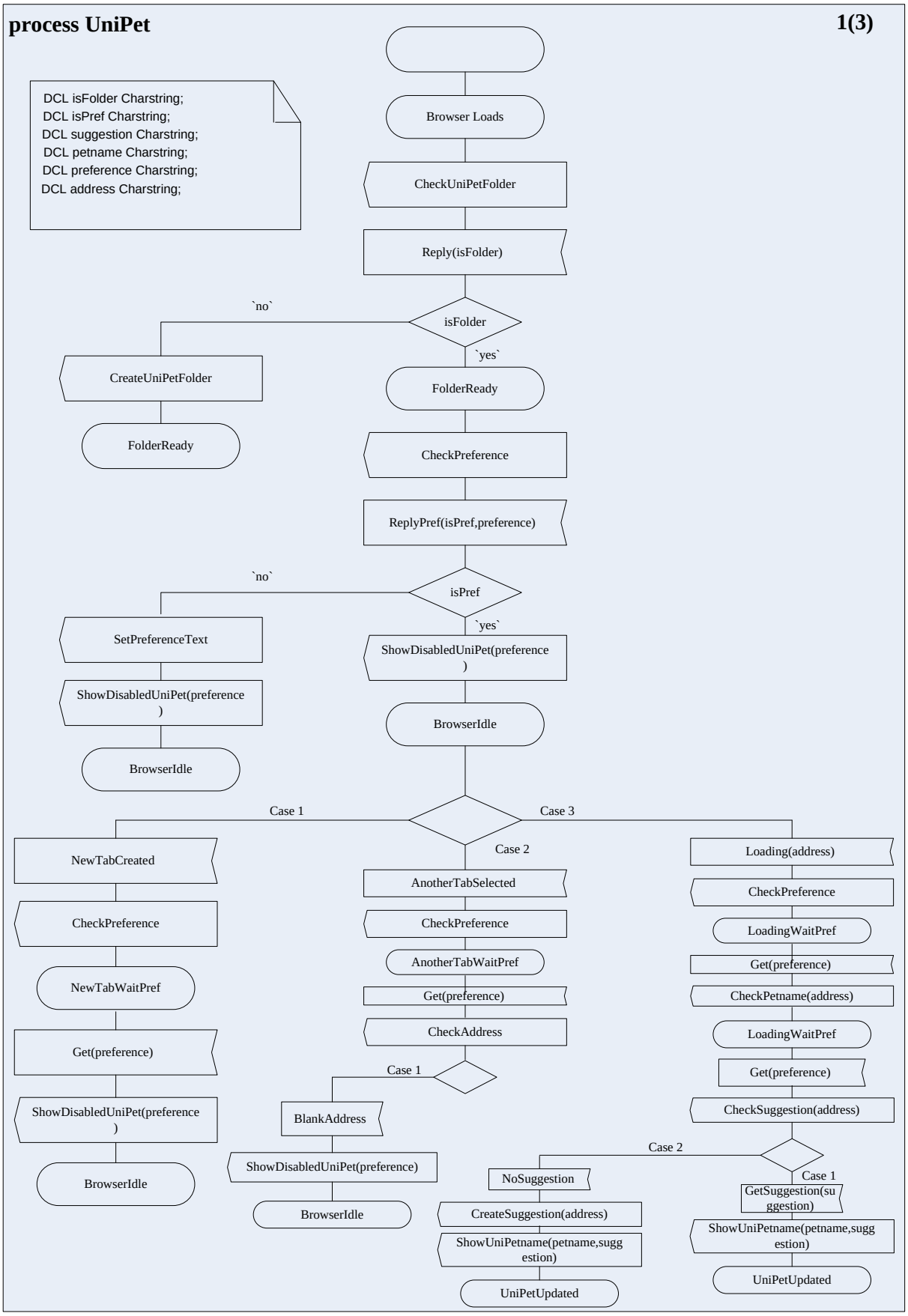

Figure 5.8: SDL 1 of the UniPet process

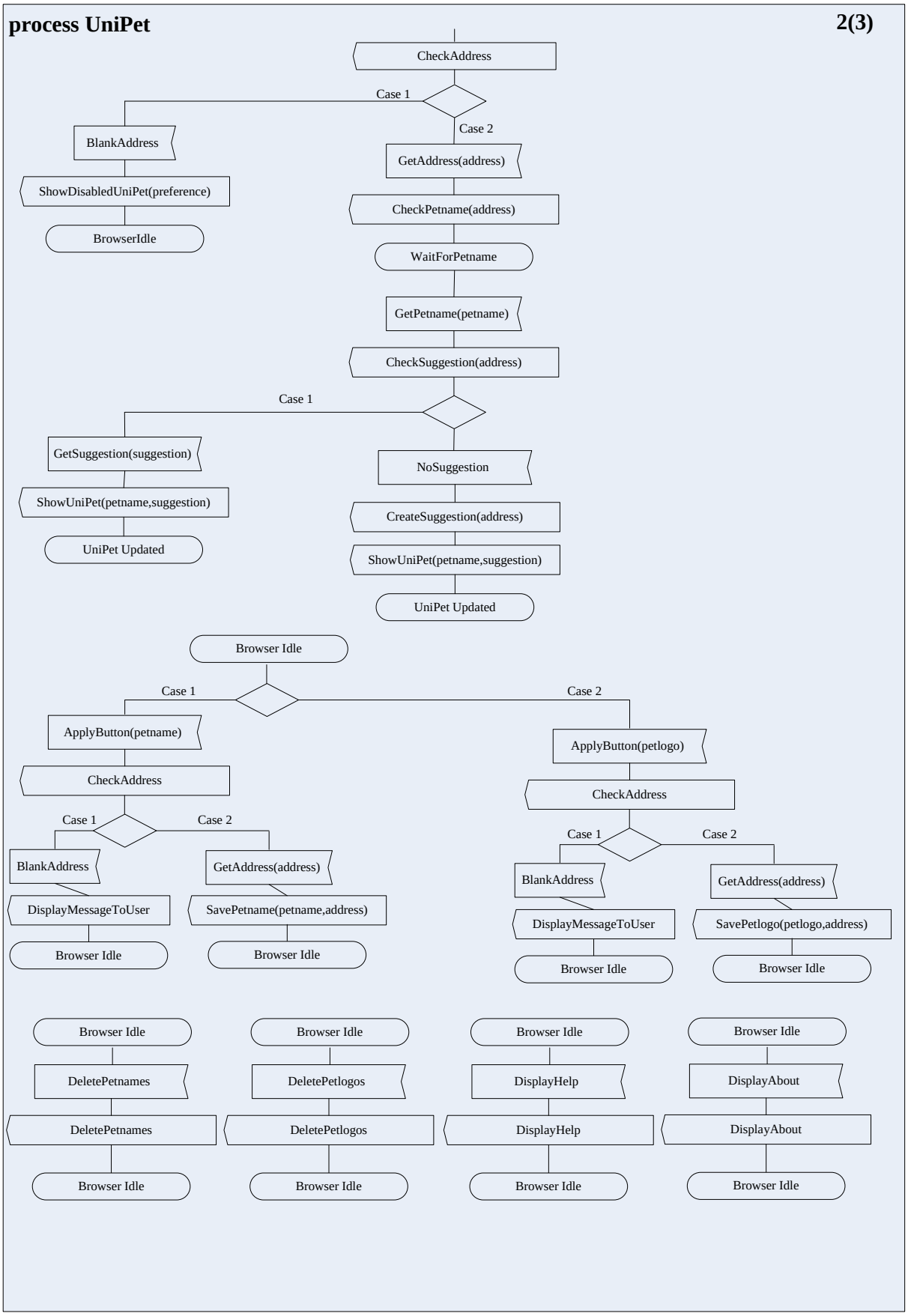

Figure 5.9: SDL 2 of the UniPet process

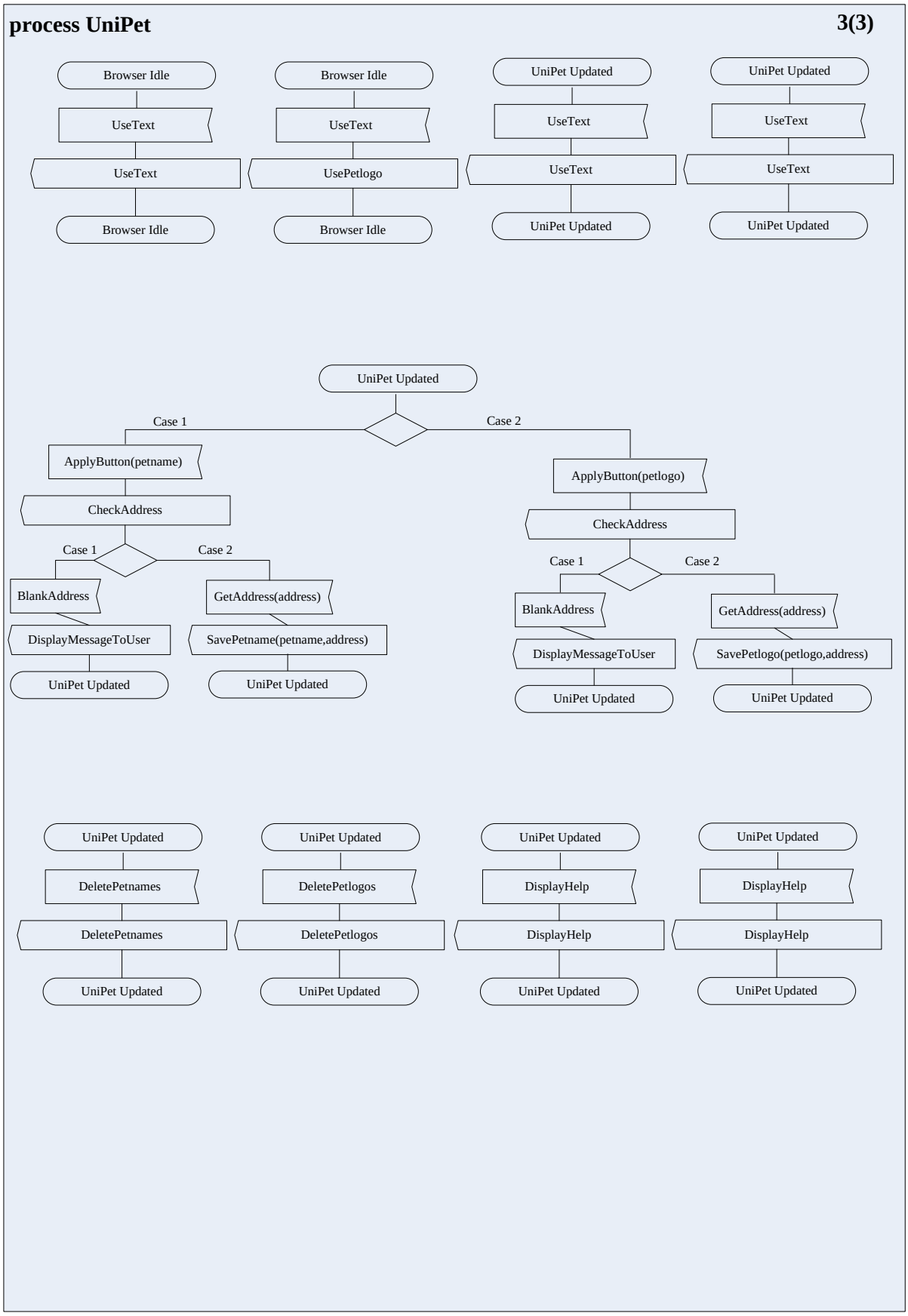

Figure 5.10: SDL 3 of the UniPet process

5.3 Used technologies

Firefox extension provides a way to extend the ability and feature found in the core browser [49]. Extension can be used to modify the behavior of the browser and is a popular method to add a completely new feature to the browser. Extension is very popular in Firefox and the Mozilla Foundation, producer of the Firefox browser, provides an extensive repository of resources for developing extensions [50]. In this section, we will briefly cover all the technologies that are commonly used to develop extension and also used for developing the UniPet.

Firefox extension uses a combination of technologies that include: CSS, DOM, JavaScript, RDF, XPCom, XPConnect, XPI and XUL [49]. The UniPet also has used DTD to accomplish some of its tasks. A brief discussion on each of them follows.

5.3.1 CSS

CSS stands of Cascading Style Sheet that is a style sheet language and is used to illustrate the appearance (look and feel) of a document written in a markup language [51]. It is a World Wide Web Consortium (W3C) recommendation for designing the layout of a document written in a markup language. The main motif was to provide a line of separation between the content of the document (usually written in markup languages) and appearance of the document, thereby providing more flexibility and control over the design and the presentation. The CSS itself is a document that follows several rules in a pre-defined format that defines the appearance of different elements found in a markup document. It usually accompanies HTML document, but also is widely used with other different types of XML documents such as XUL, SVG, etc. Firefox extension utilizes CSS heavily to define its user-interface. We have also used CSS to present graphically the dynamic behavior of the UniPet.

5.3.2 DOM

DOM stands for Document Object Model and is a cross-platform and language independent API used for accessing different elements found in a document written in markup languages such as HTML, XHTML and XML [52]. It provides a structural presentation of elements and enables dynamic modification of the content and visual layout of the markup document. It is designed to be used in conjunction with a scripting language, e.g. JavaScript, to access and modify markup document. It enables the manipulation of elements of the markup document as objects and this specific functionality provides flexibility and more control to suit the object-oriented nature of the JavaScript. Firefox extension uses DOM API to access and modify the internal components dynamically. The UniPet has also used it extensively in conjunction with JavaScript to change the behavior and layout of the application dynamically.

5.3.3 JavaScript

JavaScript is a widely used cross-platform, object-oriented scripting language that is primarily used in developing web-based applications and adds the dynamic flavor in the webpage [53]. It is not practical to be used as a stand-alone programming language like Java and C, but can

be embedded easily in other applications such as web browser. Together with HTML and CSS, JavaScript infuses the term DHTML (Dynamic HTML). JavaScript provides more dynamic control over the elements of a webpage utilizing DOM API and increases interactivity between a webpage and the user. JavaScript and Java may have the similarity in their name and also share many common rules and syntaxes; however they are fundamentally different languages with much dissimilarity. Firefox extension uses JavaScript to add dynamism and increase interactivity. The core behavior of the UniPet depends solely on the JavaScript and can be stated as the backbone of the whole system.

5.3.4 XPCom & XPConnect

XPCom stands for Cross Platform Component Object Model which is a framework for developing and maintaining cross platform components. It provides a set of core components and classes much similar to Microsoft COM to implement a set of advanced features not available otherwise [54]. JavaScript lacks in performing some tasks such as file operation, character-code conversion, etc. Other technology was needed to perform these advanced set of features, XPCom fills in that gap for the Mozilla product.

XPConnect (Cross Platform Connect) is a technology that provides interoperability between XPCom components and JavaScript so that JavaScript can easily utilize XPCom objects [55]. Many functionalities of the UniPet depend on the XPCom and XPConnect. Specifically, the UniPet has utilized the bookmark, security, location and state change service of the web progress listener, local file service, preference service and history service using the combined technology of XPCom and XPConnect.

5.3.5 XPI

XPI, pronounced as *Zippy*, stands for Cross Platform Installer Module that is used for installing Firefox extension, plug-in, themes, etc. using XPInstall technology [56]. A XPI usually contains a RDF file containing instruction for installation and a JAR in which all the necessary files for the extension are zipped together. To install the extension, users just have to download the XPI file and open it through Firefox.

5.3.6 RDF

Resource Description Framework, RDF in short, is a language to provide abstract description about resources in the World Wide Web (WWW) thus providing a metadata model for the resources found in WWW [57]. RDF uses various syntaxes to describe such metadata about resources. Firefox extension uses RDF to define an install manifest that describes various information regarding the extension. Firefox Add-ons Manager, a tool built in Firefox, is used to install extension in Firefox. This Add-ons Manager actually consults the install manifest before installing the extension to check for different issues such as compatibility, etc. The UniPet has also used RDF to define its install manifest.

5.3.7 XUL

XML User Interface Language (XUL, in short and pronounced as '*Zool*') is the basic building block for the Firefox extension. XUL is based on XML and used as the GUI markup language for the Firefox browser [58]. It provides several elements, known as XUL elements and similar to the HTML elements, for building the GUI of extensions. An XUL file is created containing all the elements needed to build the GUI of an extension. A CSS file can be linked with an XUL file to define the style sheet for the XUL elements. JavaScript with DOM API is used to dynamically remove or add new XUL elements or to change the behavior the existing elements. An array of elements are available that can be divided into several categories [59, 60]. The categories are:

- Top level elements that include window, page, dialog, wizard, etc.
- Widgets that include label, text box, button, toolbar, menu, menu item, image, check box, radio button, etc.
- Box model that includes hbox (horizontal box), vbox (vertical box), grid, stack, etc.
- Events and scripts that include: script, command, key, observer, etc.
- Others that include iframe, browser, overlay etc.

There are also a good number of attributes, some are common to all elements and some are reserved to be used with specific elements, to provide additional information regarding the XUL elements. It is important to note here that an XUL document is actually an XML document, therefore the common rule and syntax that need to be followed in XML, also need to be followed in XUL. A DTD document, described next, can be linked with an XUL document to define its structure, legal elements and their attributes. The UniPet has used a number of XUL elements to define its GUI that was mentioned in 5.2.1.

5.3.8 DTD

Document Type Definition (DTD, in short) is used to define the document structure of an XML element with a list of legal elements and their attributes that may reside in the respective XML document [61]. As an XUL document is essentially a variant of the XML, a DTD document can be linked with an XUL documentation to define its structure and to define the legal elements and attributes. The UniPet has used several DTD documents to define the structure of several XUL documents.

5.4 Implementation

This section will describe how the technologies discussed in the previous sections have been put together to achieve the level of interaction mentioned in the subsection 5.2.2. The UniPet consists of several files: one manifest file, one RDF file, several XUL files, one unipet.properties file, several JavaScript files, one CSS file, a few DTD files and one html file. A brief description on each file will be provided in the subsequent subsections. Figure 5.11 presents a conceptual illustration of the UniPet and its several components.

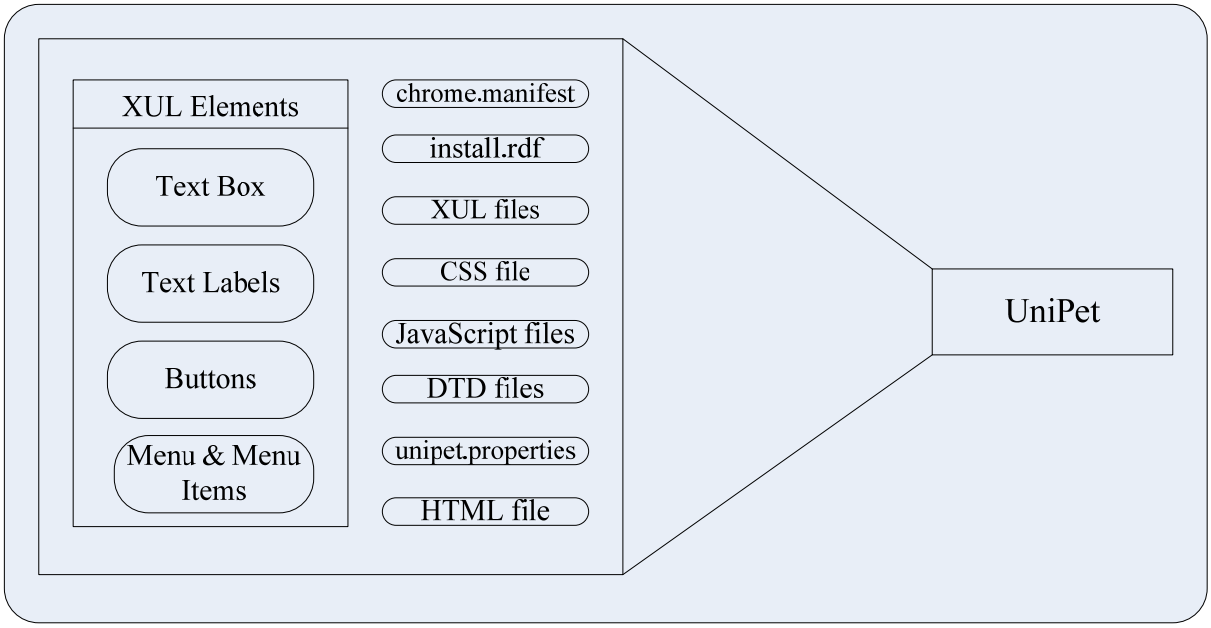

Figure 5.11: The UniPet components

Figure 5.11 explains that the UniPet contains several XUL elements that build the GUI and several files that define its behavior.

5.4.1 Chrome

Before describing each of the files, some terminologies regarding Firefox extension development need to be clarified. We will start with the '*chrome*'. Chrome is used to describe the structural elements of an XUL application [62]. Firefox itself is an XUL application and all the elements other than the webpage displayed are part of the chrome. Extension, being an XUL application too, utilizes chrome to define its structure. Chrome consists of three packages: *content*, *skin* and *locale*. The content package contains all the XUL and script files that are part of the extension. The skin package contains the style sheet files and image files whereas the locale package contains language data that can be used for language translation. To be compatible with different GUI languages, several locale packages could be used, each for one language.

Chrome URL is a special type of URI scheme used essentially with Firefox extension [62]. It is extensively used in extension development to point to the location of a file in the local file system. The structure of a chrome URL is:

```
chrome://%package name%/%package type%/%relative path to source file%
```

Here, "*package name*" defines the name of the extension and its location in the local file system, "*package type*" defines the type of packages, e.g. content or skin, etc and the last portion of the URL states the name of the file. The following chrome URL is used to represent the browser itself:

```
chrome://browser/content/browser.xul
```

5.4.2 Overlay

Overlay is one of the distinctive features of XUL that allows several XUL files to be treated as s single XUL file during processing and thus supporting modularization [63]. This technique is used in developing extension that requires new elements to be inserted into the chrome of the browser. There are two types of overlay: normal overlay in which it is necessary to use the "*xul-overlay*" instruction to the target XUL document so that another XUL document can be overlaid in the first one and cross package overlay in which there is no need to use the "*xul-overlay*" instruction, but an external manifest file is necessary to define the source and the target overlay files. The second approach is used for the extension development to allow extension GUI to be overlaid in the browser. The overlay element has to be the root element for the XUL file of the extension that will be merged with the XUL document of the browser.

5.4.3 Manifest file

The file "*chrome.manifest*" is used to register the packages of the extension with the browser and to invoke the cross package overlay as explained in subsection 5.4.2. The file contains the location of the packages relative to the chrome URL, the name and location of the source and target overlay files and the name and location of the all the CSS files. Code snippet for this file is given in Listing 5.1.

```
content   unipet                    jar:chrome/unipet.jar!/content/
skin      unipet    classic/1.0     jar:chrome/unipet.jar!/skin/
locale    unipet    en              jar:chrome/unipet.jar!/locale/en/

overlay chrome://browser/content/browser.xul chrome://unipet/content/overlay.xul
style chrome://browser/content/browser.xul chrome://unipet/skin/unipet.css
style chrome://global/content/customizeToolbar.xul chrome://unipet/skin/unipet.css
```

Listing 5.1: Code snippet for the chrome.manifest file

First line in Listing 5.1 defines the name of the package, name of the extension and the relative location of the package. The line that starts with *overlay* keyword defines the source and target overlay XUL files respectively and the last line defines the style sheet which starts with the *style* keyword, then there is the location and name of the XUL file in which the style sheet will be applied and at last the name and location of the CSS file.

5.4.4 RDF file

The "*install.rdf*" file is the installation manifest file for each extension. When the extension is installed for the first time, the Add-ons manager of the Firefox checks this file for different information such as unique id for the extension, type of add-on, version, name of the creator, email of the creator, homepage of the extension, a small description of the extension, maximum and minimum version of the browser in which the extension is supposed to work, etc. Listing 5.2 shows the code snippet for this file. All parts of the codes are self-explanatory except the type. There are many different types of Firefox add-ons such as theme, extension, plug-ins, etc. Type 2 defines that the application is an extension.

```
<?xml version="1.0"?>

<RDF xmlns="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:em="http://www.mozilla.org/2004/em-rdf#">
  <Description about="urn:mozilla:install-manifest">
    <em:id>unipet@example.net</em:id>
    <em:version>0.99</em:version>
    <em:type>2</em:type>
.....................................................................
<em:name>UniPet</em:name>
 <em:description>Manage your website identities more securely with the
UniPet</em:description>
 <em:creator>Md. Sadek Ferdous</em:creator>
 <em:homepageURL>http://www.example.com/</em:homepageURL>
  </Description>
```

Listing 5.2: Code snippet for the install.rdf file

5.4.5 XUL files

The "*overlay.xul*" file is the main XUL file that is overlaid in the browser and defines the XUL elements that formulate the GUI of the UniPet. Because of the overlay technology, when the browser loads itself, UniPet becomes the part of the browser. It also defines the

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/css" href="chrome://unipet/skin/"?>
<?xml-stylesheet type="text/css" href="chrome://unipet/skin/unipet.css"?>
<overlay id="unipetOverlay" xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">
.....................................................
<hbox id="pet_hbox" align="center" style="border: 1px solid ;">
<label id="name" class="pet_label" tooltiptext="Enter a name/Select an image and click
apply">Petname:</label>
<textbox id="unipet-text" class="unipet-disabled" readonly="true" size="18" onfocus="tBFocus()"/>
……….……
</hbox>
</toolbar>
<script type="application/x-javascript" src="unipet.js"/>
</overlay>
```

Listing 5.3: Code snippet for the overlay.xul file

place in which the UniPet will be placed in the browser as well as different attributes and command actions of the XUL elements are also described. Listing 5.3 shows the code snippet for this file. The unipet.css file has been linked in this file to define the style sheet property of the XUL elements. The unipet.js that defines the behavior of the XUL elements is also linked in this file.

The "*collision.xul*" file is used to define the XUL elements that will be used to construct an informative warning message for the scenario when users try to assign similar looking Petnames for different entities. The "*same.xul*" file is similar in purpose like collision.xul but used for scenario when users try to assign same Petname for different entities. The "*aboutDialog.xul*" file is used to construct an informative dialog box using XUL elements. This dialog box is displayed when the user clicks the about menu item. The "*collision.xul*",

"*same.xul*" and "*aboutDialog.xul*" use "*unipet-collision.dtd*", "*unipet-similar.dtd*" and "*about.dtd*" files to define the structure of the XUL documents.

5.4.6 JavaScript files

As previously told, the "*unipet.js*" file is the backbone of the UniPet. It defines the dynamic behavior of the XUL elements that are needed to establish and maintain the complex level of interaction of the UniPet. It also uses the combined technology of XPCom and XPConnect to access different services that were mentioned previously. It is a large file with around 1300 lines of codes. We are not providing any code snippet here, as presenting a few lines of codes will be meaningless, in our point of view, to justify its purpose. The file is itself well documented and there should be little trouble understanding the codes, should anyone is interested.

The "*collision.js*" and "*aboutDialog.js*" files are used to define the behavior of the XUL elements used in "collision.xul" and "aboutDialog.xul" respectively.

5.4.7 Unipet.properties

This is a non-standard file that is used to define some common attribute values of the XUL elements and is linked with overlay.xul file so that these states become part of overlaying XUL file and are accessible in the script file. Different XUL elements will share these values at different situations. Listing 5.4 shows the code snippet for this file.

```
bookmarks.title=unipet
init.title=
init.tooltip=No Site to apply Petname.
null.title=unknown site
.....
enabled.tooltip=Assign a petname to this site before exchanging sensitive information.
edit.tooltip=Edit the previously applied Petname.
```

Listing 5.4: Code snippet for the unipet.properties file

5.4.8 HTML File

An HTML named "*Help on toolbar.html*" has been used to write an extensive help page regarding the UniPet. This help page describes the different functionalities of the UniPet, provides a short description on its purposes, on Phishing and how the UniPet can help in defense against Phishing. Users can watch this help by choosing the Help menu item from the UniPet.

5.4.9 CSS File

The file "*unipet.css*" has been used to define style sheet for all the XUL elements and contains the styles that need to be applied to the XUL elements at different situations.

5.5 Packaging

Before an extension can be distributed for downloading and installation, it is necessary to package them in XPI format. To do so, all the files of the content, locale and skin packages are grouped together in a jar file. That jar file is then placed under a directory named "*chrome*". Then this directory, the manifest and the RDF file are zipped together to create a zip file. The zip file has to be renamed as .xpi file. This .xpi file is the desired extension installer. Users can install the extension by opening the .xpi file from the browser. We also created an XPI file named "*unipet.xpi*" that acts as the installer for the UniPet.

Chapter 6

SECURITY USABILITY OF THE UNIPET

In this chapter we will analyze the security usability of the UniPet using the Cognitive Walkthrough method. While performing the Cognitive Walkthrough for the UniPet, we will try to note if the application satisfies the usability properties discussed in subsection 3.3.2. The degree of compliance with the specified security usability properties will give an indication of the level of security usability of the UniPet.

6.1 Setup

The UniPet can be installed by opening the "*unipet.xpi*" file using the File menu of the Firefox. At this point, Firefox Add-ons manager will start installing it in the browser by looking at the chrome.manifest and install.rdf files and then utilizing other files of the UniPet. Once installed in the browser, the UniPet will look like Figure 6.1.

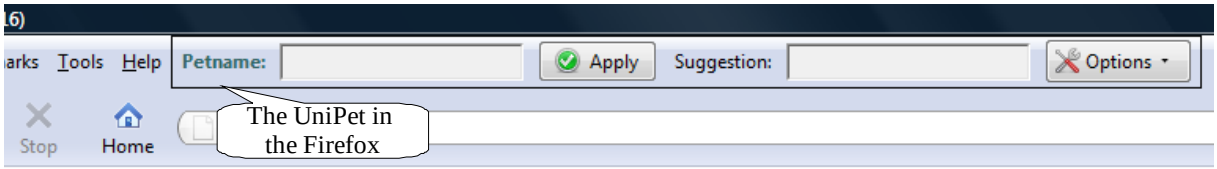

Figure 6.1: Installed UniPet in the Firefox

First thing to note about the UniPet is its location. It has been placed just beside the menu bar and at the top of the address bar. The Petname tool was located beside the search bar that squeezes the address bar a bit (Figure 4.1). The TrustBar was located at the top of the menu bar that created an extra toolbar on top of the menu bar (Figure 4.9). We think that the location for such kind of application is very important and users may feel uncomfortable with a squeezed address bar or an extra toolbar that resides at the top of the menu bar. We believe that placing the UniPet in such location serves dual purpose: 1) it does not alter the structure of any element of the browser, as the UniPet utilizes only the unused space of the menu bar and 2) dynamic changes of different elements of the UniPet may catch more user attention as it resides just top of the address bar and as address bar itself resides in between the main focus area (assuming the displayed webpage and the address bar is in the primary focus area of the user while browsing). Placing it beside the search bar or at the top of the menu bar can make it more out of focus.

6.2 Functionality

The main purpose of the UniPet is to allow a user to assign a Petname/Petlogo for a website that he wants to correctly recognize and to display that Petname/Petlogo when he visits the site later. The Petname/Petlogo will be absent if the visited site is not the intended one. A user can judge if a webpage comes from a previously identified website by observing the presence or absence of the Petname/Petlogo. It also allows the user to edit and delete Petnames and Petlogos. The Petname Tool utilizes different XUL elements to build its GUI. Some XUL elements are dynamic in nature and they change accordingly to indicate different situations. The dynamic elements are: 1) The text in the text field, 2) The typeface of the text, 3) Color of the text field, 4) Label, 5) Button, 4) Tooltip, 5) Dialog box and 6) Alert box. Different texts with different typefaces are displayed in the text field in different situations, color of the text field changes, different labels are displayed, different tooltips are provided accordingly when mouse pointer is placed over the text field and warnings are displayed using dialog and alert boxes. The UniPet allows user to use either Petname or Petlogo for any site and provides options to switch back and forth between Petname and Petlogo seamlessly. It also provides an About menu item that displays relevant information regarding the UniPet and a Help menu item to explain the purpose and the functionalities of the UniPet in detailed fashion. Figure 6.2 shows the GUI components of the UniPet.

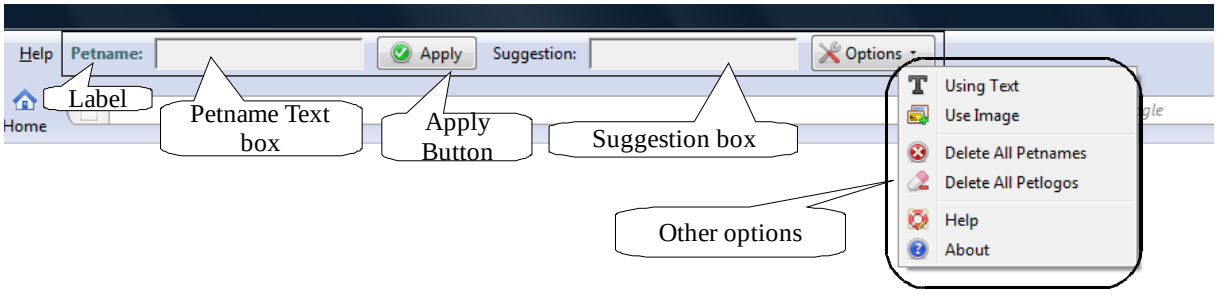

Figure 6.2: GUI components of the UniPet in Firefox

Some examples can illustrate how the UniPet operates. When the browser loads with blank page, there are two ways the UniPet can display itself depending on the last user preference. If the last user preference was to use Petname, the UniPet will look like the one shown in Figure 6.1. Both the Petname text box (the text box just beside the label Petname) and the suggestion text box (the other text box) will be disabled with grey color. If a user tries to select the text, an alert will pop up stating "*No site to apply Petname*" (Figure 6.3). While the Petname text box is disabled and the user clicks the "*Apply*" button, the same alert box will be displayed. This is to ensure that blank address should not have any Petname. The tooltip for the Petname text box is "*No site to apply Petname*" and the suggestion box is "*Suggestion regarding Petname will be displayed here*". The general tooltip that does not change over different situations for the Options menu is "*Click here for more options*".

If the last user preference was to use Petlogo, the UniPet will look like the one shown in Figure 6.4. As seen in the figure, when the UniPet is in Petlogo mode, the Petname text box is removed and a label (Petlogo label) and another button are added in the UniPet. The label contains the text "*Image Disabled*" to indicate the situation that currently there is no webpage and thereby no Petlogo can be assigned or displayed. At this point, clicking any of the two

buttons will show the same alert box of Figure 6.3 with message "*No site to apply Petlogo*". Also the label with '*Petname*' text will be substituted by the text '*Petlogo*'.

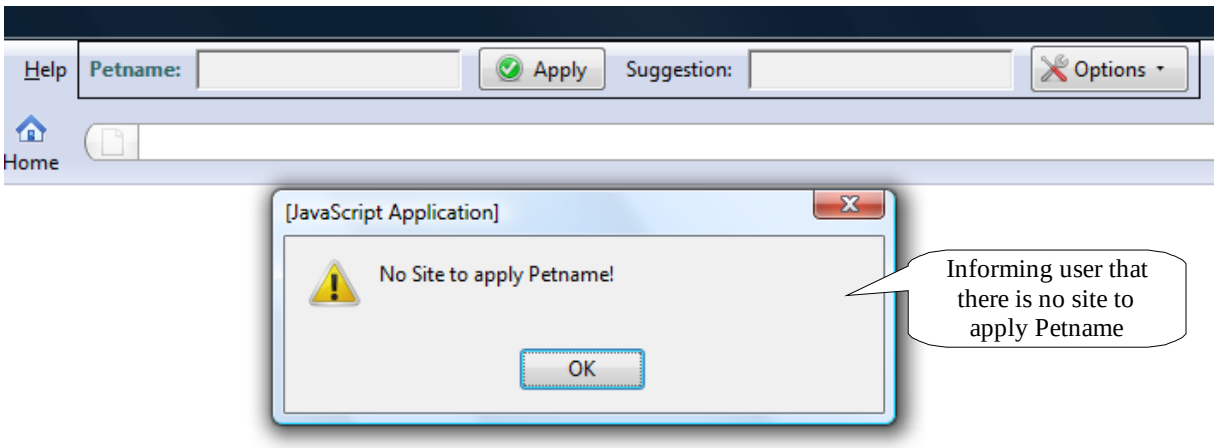

Figure 6.3: The UniPet with disabled Petname text box

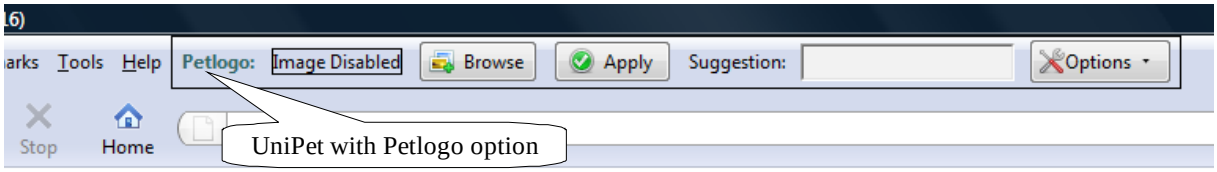

Figure 6.4: The UniPet with Petlogo option

If the UniPet is in the Petname mode and the user visits an HTTP site, e.g. *www.cricinfo.com*, and there is no Petname for that site, the Petname text box will be enabled with red color and with italic text "*unknown site*" (Figure 6.5). The color of the suggestion box turns to deep sky

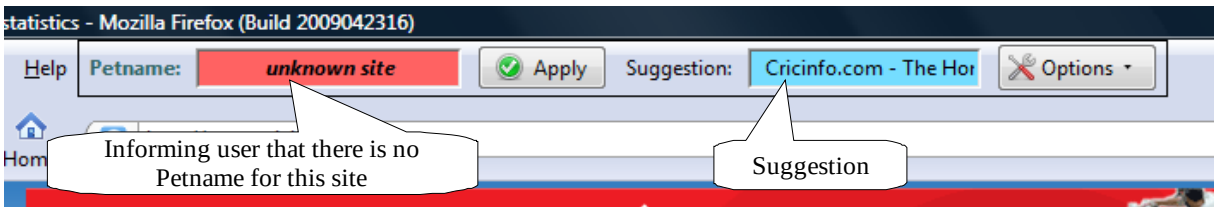

Figure 6.5: Informing user about the absence of a Petname for an HTTP site

blue color and contains the title page of the web site as a suggestion. The tooltip for the Petname box becomes "*Apply a Petname for this site*". At this point, a user can select the Petname text box and write down a Petname and click the Apply button. If the Petname is not same or similar to other Petnames, it will be assigned against the last two parts of a domain name of the web page and the color of the text box turns to light green and the assigned Petname is displayed in the text box with normal typeface (Figure 6.6).

Whenever a user wants to edit an existing Petname, he has to select the Petname text box. The text box will turn to red and the type face of the text will be italic to indicate that the user is

editing the existing Petname (Figure 6.7). Once he writes down the new Petname and clicks the Apply button, the UniPet will check if it is similar or same to the existing Petnames. If not, the new Petname will be assigned for the site.

Figure 6.6: Displaying Petname for a site

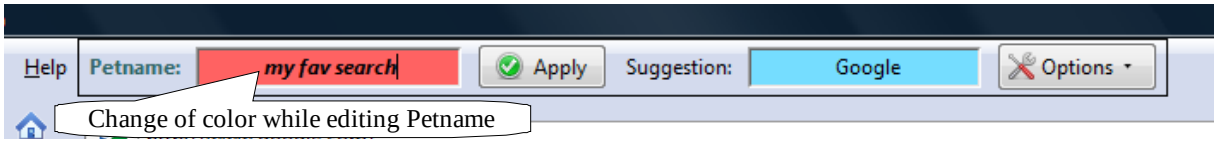

Figure 6.7: Petname editing

When the user will visit an HTTPS site and there is no assigned Petname for that site, he will be explicitly warned to apply a Petname for the site (Figure 6.8). We choose this option to warn the user for such scenario as HTTPS site often carries very sensitive information and a theft of identity could be catastrophic. Therefore it is very important to make sure that the visited site is the intended one. As found by the analysis in the previous chapters, applying a Petname for sensitive site can make a user better informed.

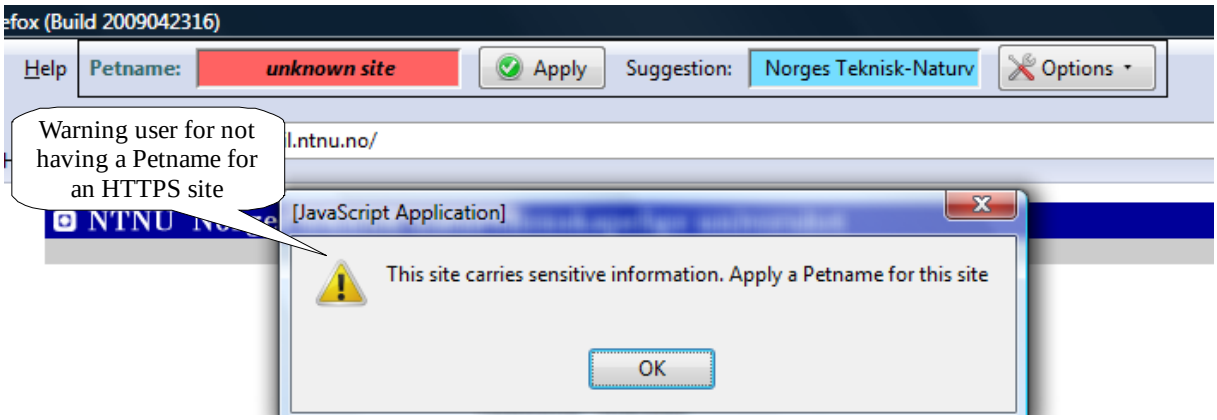

Figure 6.8: User warning to indicate the absence of a Petname for an HTTPS site

Whenever a user writes down a Petname and clicks the apply button, the UniPet checks if they are same or similar to the existing Petnames. If found that there are same or similar Petnames, the user is alerted explicitly using dialog boxes (Figure 6.9 and Figure 6.10). It is important to note here that same means two Petnames are exactly same and similar means two Petnames are similar in some way, e.g. my web mail and my web. As both these types can confuse the user, users will not be allowed to assign same or similar Petnames. The dialog box contains the information if the current Petname is same or similar to the existing

Petnames; the existing Petname that is same or similar to the current is also shown in a button in the dialog box. The user can check the web site for which the existing Petname has been assigned by clicking the "*Go to*" button. The dialog box also contains the date on which the exiting Petname has been created.

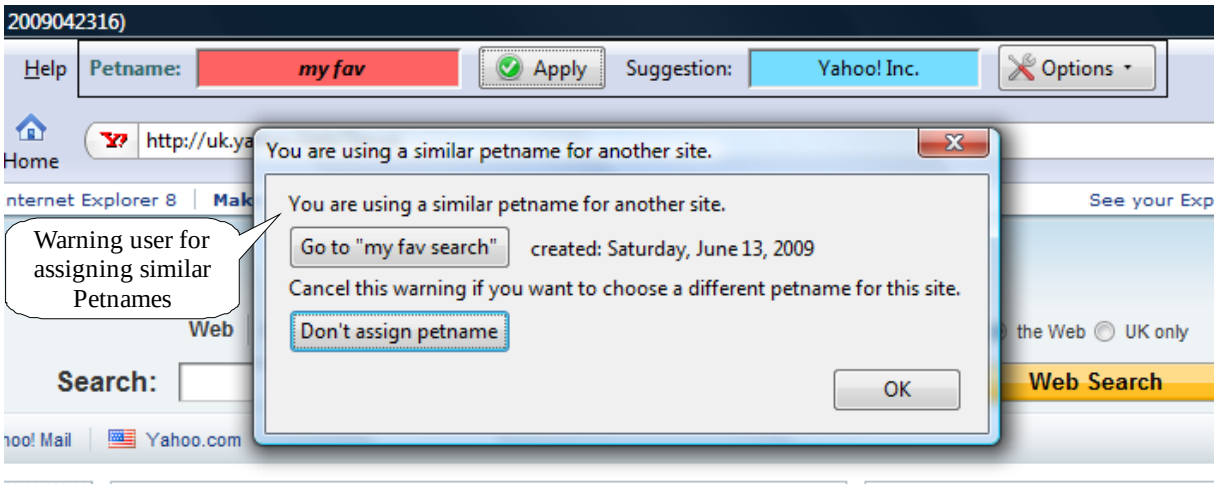

Figure 6.9: Warning on similar Petnames

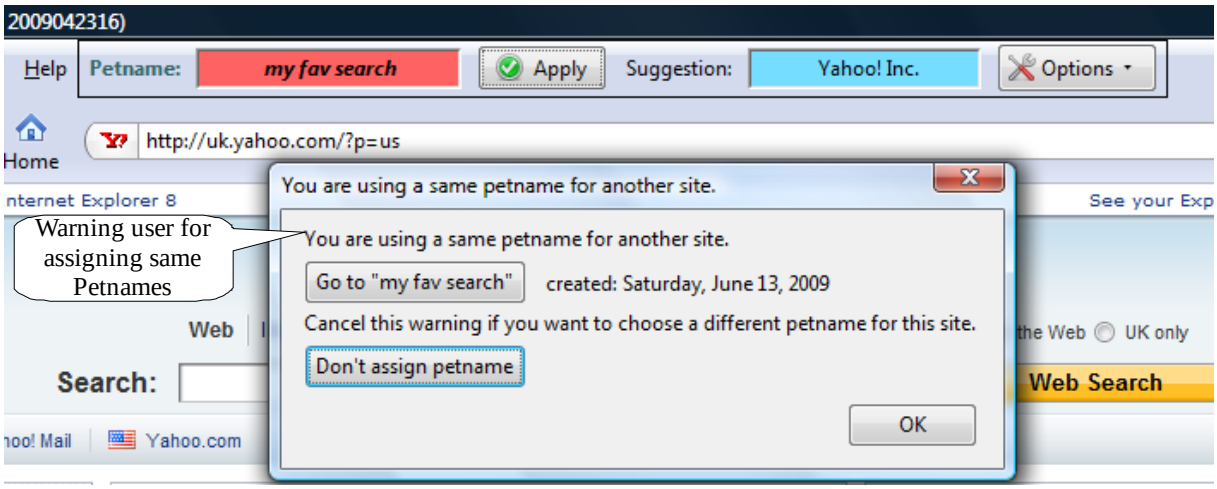

Figure 6.10: Warning on same Petnames

A user can delete all the existing Petnames altogether by clicking the "*Delete All Petnames*" menu item from the Options menu. The user will be asked if he really wants to delete all the Petnames using a confirmation box (Figure 6.11). If user clicks OK, all the Petnames will be deleted from the UniPet bookmark folder; otherwise no action will be taken.

If the UniPet is in the Petlogo mode, the user visits an HTTP site (*www.cricinfo.com*), and there is no Petname for that site, the Petlogo label (the label at the left of the Browse button) will have the text "*Add Image*" in red color (Figure 6.12). The suggestion box turns to deep sky blue color and contains the title page of the web site as a suggestion. The tooltip for the Petlogo label becomes "*Click the browse button to add an image*". At this point, a user can click the Browse button and a File Open dialog box will open. The user can select an image from his computer and click the Apply button. The Petlogo will be assigned against the last

two parts of the domain name of the web page, a confirmation message will be shown to the user and the assigned Petlogo will be displayed in the UniPet (Figure 6.13).

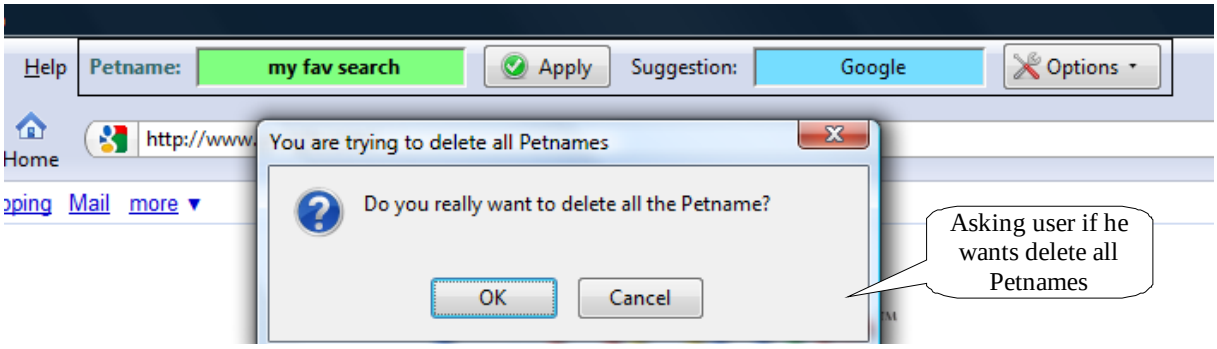

Figure 6.11: Asking user for his consent to delete all Petnames

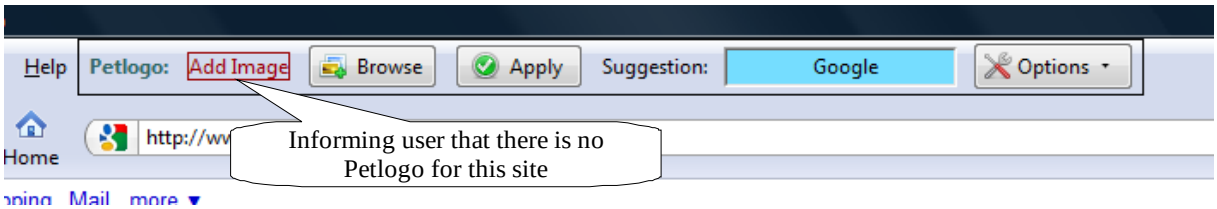

Figure 6.12: Informing user about the absence of a Petlogo for an HTTP site

Figure 6.13: Displaying Petlogo for an HTTP site

Whenever a user wants to edit an existing Petlogo, he has to click the Browse button and select an image. Once an image is selected, the UniPet will check if the same image is being used by any other website. If not, the new Petlogo will be assigned for the site when the user will click the apply button. A confirmation message will be shown to the user.

When the user will visit an HTTPS site and there is no assigned Petlogo for that site, he will be explicitly warned to apply a Petlogo for the site (Figure 6.14). We choose this option for the same reason as explained previously.

Whenever a user selects an image as a Petlogo, the UniPet checks if the same image is being used by any other website. If found that another website is using the same image, the user is alerted explicitly and the user is presented with the File open dialog box again to choose another image (Figure 6.15).

A user can delete all the existing Petlogos altogether by clicking the "*Delete All Petlogos*" menu item from the Options menu. The user will be asked if he really wants to delete all the

Petlogos using a confirmation box (Figure 6.16). If user clicks OK, all the Petlogos will be deleted from the UniPet Petlogo file; otherwise no action will be taken.

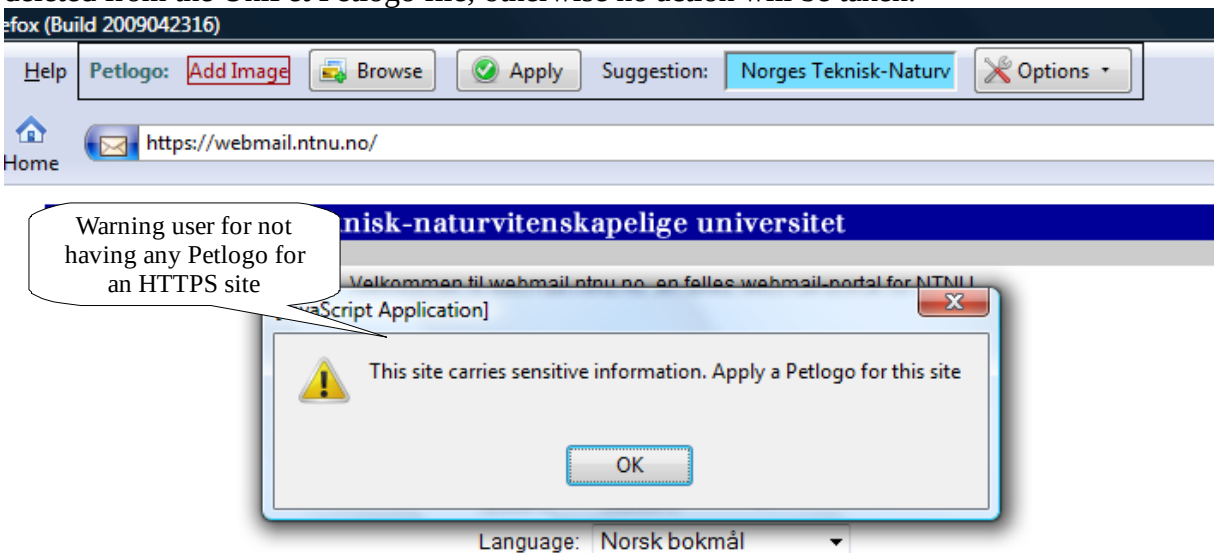

Figure 6.14: User warning to indicate the absence of a Petlogo for an HTTPS site

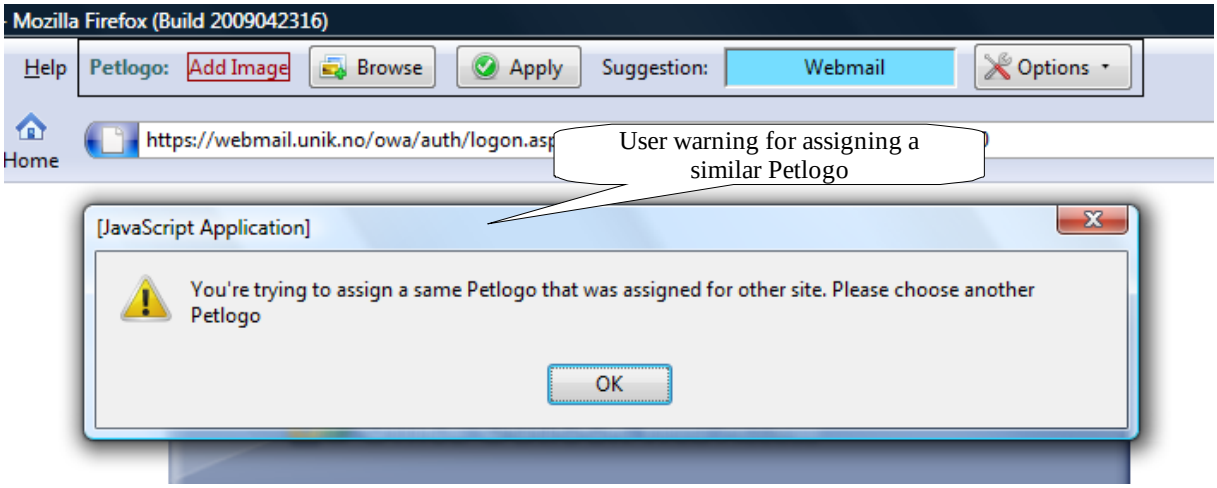

Figure 6.15: Warning on similar Petlogos

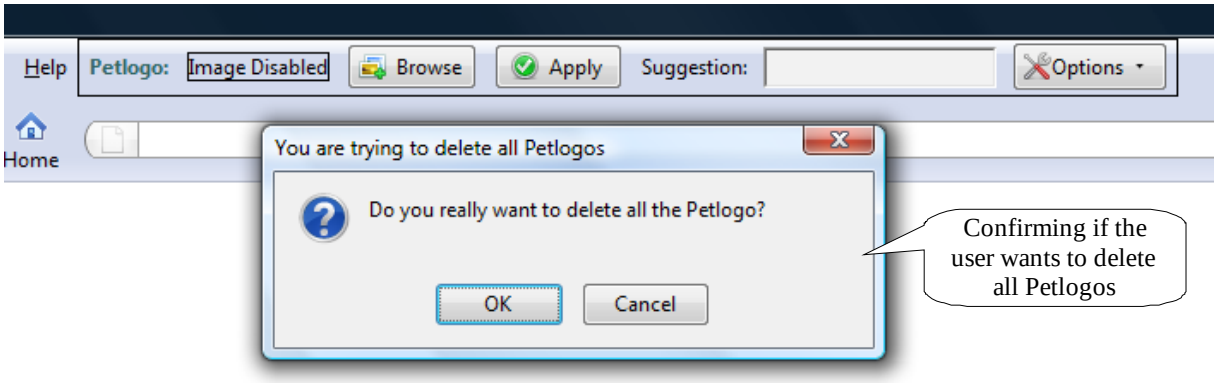

Figure 6.16: Asking user for his consent to delete all Petlogos

If a user selects the Help menu item from the Options menu, an HTML file will be opened in a new tab (Figure 6.17). The help page contains detailed explanations on Phishing, counter measures of Phishing, the purpose of the UniPet, the UniPet functionalities, how the UniPet can be used as an aid against Phishing, etc.

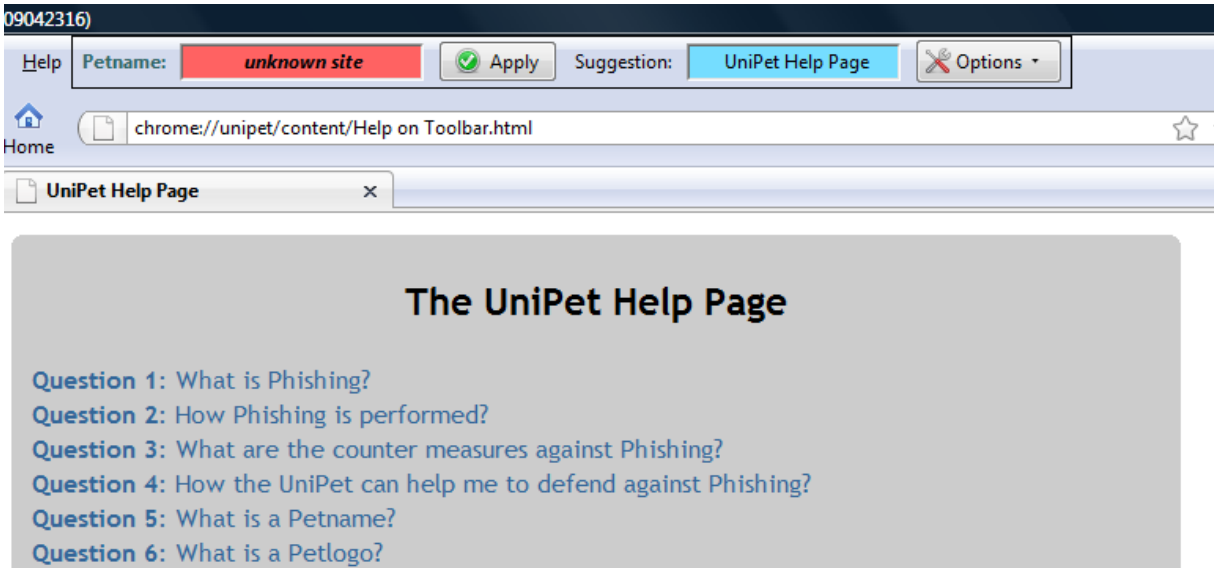

Figure 6.17: The UniPet help page

If a user selects the About menu item from the Options menu, a dialog box will be shown to the user. The dialog box contains a short explanation regarding the purpose of the UniPet, name and email of the developer, etc (Figure 6.18).

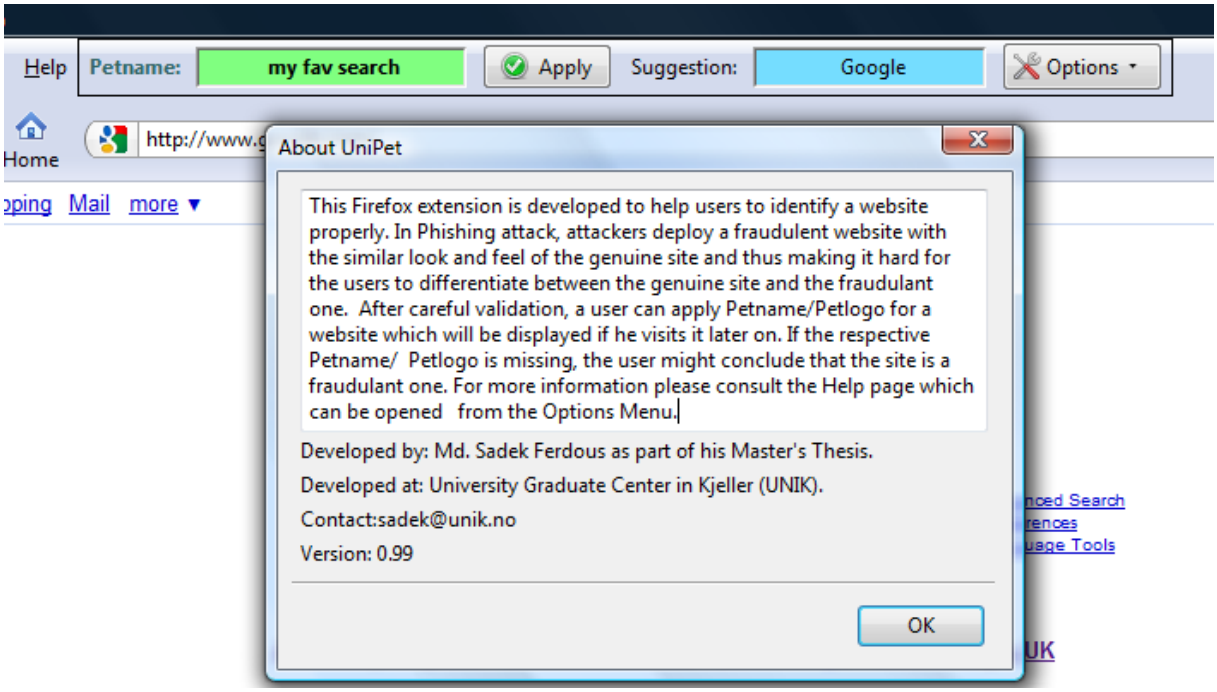

Figure 6.18: The UniPet about dialog box

6.3 Evaluation

In the following, the UniPet will be analyzed for compliance with the Petname System properties of Section 3.3. The UniPet deploys Petnames as well as Petlogos. The last two parts of a domain name of a web site is used as the Pointer and is strongly resistant against forgery. Therefore we conclude that the UniPet satisfies F1 and F3. The UniPet does not allow assigning same Petname or same Petlogo for different websites. It even does not allow assigning similar looking Petnames for different websites. Therefore it preserves the principle of bi-directional one-to-one mapping for each entity and thus F4. It also utilizes the title of a webpage (Nickname) for HTTP sites or the organization name of the subject when there is a certificate available (for HTTPS sites) to act as suggestion. Therefore we can conclude that it deploys Nickname and thus satisfies the optional property F2.

The UniPet enables users to explicitly assign a Petname for each entity, e.g. to select the text field, write down a Petname and click the Apply button. It also involves users actively to select a Petlogo. Users have to browse and select an image and then click the Apply button to assign it as the Petlogo. These active involvements satisfy SA1 and SA2. Users can change a Petname/Petlogo at any time, thereby satisfying SA3. The title of a webpage for HTTP sites or the organization name of the subject from the certificate of the HTTPS site is selected as a suggestion for aiding users to select a Petname. It satisfies SA4. If a user wants to choose this suggestion to act as a Petname, he has to copy and paste it into the Petname text box, change it a bit so that it is not same as the Nickname (suggestion) and click the Apply button; meaning to accept suggestion as a Petname users have take explicit actions. This satisfies SA5. The UniPet tool also makes sure that the Petname is not an exact copy of the Nickname (suggestion) and thus complies with SA6. Whenever a user selects a Petname that closely resembles existing Petnames or selects the same Petlogo which is already being used as a Petlogo, the user is alerted with an informative dialog box (Figure 6.9, Figure 6.10 and Figure 6.15). Therefore we can conclude that the UniPet satisfies SA7 and SA8 too. Whenever a user is involved with highly sensitive data transmission (indicated by the HTTPS protocol), the UniPet shows alert if no Petname or Petlogo is found for the site and therefore satisfies SA9.

The Petname/Petlogo, if already supplied by the user, is displayed on the UniPet all the time, thereby satisfying SC1. Different typefaces, tooltips, colors and alert boxes have been used in the UniPet to catch the user attention to indicate the presence or absence of a Petname/Petlogo. For the Petname, we choose the red color to indicate the absence of a Petname and green color to indicate the presence of the Petname. These two colors are proved to be the most eye catchy colors [45]. Thereby we can conclude that the UniPet is compliant with SC2 and SC3. The UniPet deploys two different colors (red or blue for the Petname and deep sky blue for suggestion) to indicate the Petname and the Nickname. There are also two labels stating as "*Petname*" and "*Suggestion*" by the left of the Petname and Suggestion text boxes to distinguish them more visually. We believe that these two approaches are sufficient enough to satisfy SC4. The UniPet provides warning through dialog and alert boxes when there are conflicts with other Petnames or if there is an ambiguity between Petnames/Petlogos and thus satisfies SC5.

6.4 Comparison

Our analysis in Section 6.3 proves that the UniPet has satisfied all the properties of the Petname System, both optional and mandatory. We can summarize them in Table 6.1. To provide a better comparison we have also included the other two applications in the table.

Tool Name	F				SA									SC				
	1	2	3	4	1	2	3	4	5	6	7	8	9	1	2	3	4	5
UniPet	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Petname Tool	Y	N	Y	N	Y	Y	Y	N	N	N	N	Y	N	Y	Y	Y	N	Y
TrustBar	Y	Y	Y	N	Y	Y	Y	Y	Y	Y	N	N	N	Y	Y	Y	Y	N

Table 6.1: Comparison among the Unipet, the Petname Tool and TrustBar

As the UniPet satisfies all the properties of the Petname System, we can conclude that it fully satisfies all the Security Usability principles. We believe that it will provide users with an improved and secure way of managing SP identities on their side.

Chapter 7

CONCLUSION

7.1 Summary

In the first part of the thesis, we have focused on providing a brief overview of Petname Systems starting from the introductory concept of Entity, Identity and Identity Management with a brief description on different IdM architectures. We have found that the Petname Model is well suited to be integrated in the Personal SP Identity model. We also provided a brief description on Identity Theft and the Phishing attack with different attack techniques and defense mechanisms. Then we summarized the history and evolution of the Petname Model in one place. Previously it was scattered among several web articles. We have formally defined the properties of Petname Systems and explained how this set of properties can satisfy the essential security usability principles. It is our belief that if these properties are followed in developing applications based on the Petname Model, it will improve the user experience and improve overall security by removing security vulnerabilities related to poor usability. The thesis has also analyzed two available Petname-based applications for server identification management and shown that they represent an improvement in usability, but unfortunately do not satisfy all the specified Security Usability principles.

In the second part, we have developed the UniPet, a Petname Model based application with similar functionalities of the Petname Tool and the TrustBar, that utilizes the concept of aiding user in identifying SP identities securely on their side. We have deployed several technologies to meet the complex level of interaction the UniPet asks for. We have provided a brief discussion on each of the technologies to better understand the UniPet architecture. We have also shown that the UniPet has been a major improvement on GUI and on the security usability issues over those two applications. The UniPet satisfies all the properties of a Petname System and thus is fully compliant with the Security Usability principles. We believe that the UniPet will provide the users with an improved and secure browsing experience.

7.2 Future work

In the UniPet, the title of a web page for HTTP sites and the organization name of the subject from the certificate for HTTPS sites have been chosen to act as suggestions. But we think there are more scopes for research to find a good suggestion that really helps in finding a Petname.

The UniPet has used the last two parts of a domain name to act as the Pointer. For example, if the domain name is *www.google.com*, the Pointer of the UniPet is *google.com*. However, if the domain name consists of different Top Level Domain (TLD), then other parts of the domain name also need to be part of the pointer. For example, if the last two parts of the domain name are *ac.uk*, obviously it is not a unique pointer as there are many more websites that may contain these two parts. A proper solution could be to ask users to assign the number of level for each domain name that he wants to be part of the pointer and then the UniPet could build the pointer based on that level number. For example, if the domain name is *www.reading.ac.uk*, users can assign the number of level as 3 and then the pointer will be *reading.ac.uk* which is unique. The next version of the UniPet will add this feature.

We used Cognitive Walkthrough method to evaluate the security usability of the UniPet. This method largely depends on the expertise of the evaluator to produce a good and unbiased evaluation. Another approach that could be used is "*Laboratory User Test*" in which a group of users will be asked to evaluate the usability and the effectiveness of the UniPet to generate a more general and unbiased evaluation.

As the Petname Model is based on Zooko's triangle, any shortcut in the triangle may collapse the relationships among the components of the Petname Model or may create a new dimension of relationship. Bob Wyman in his web blog proposed to update the Zooko's triangle into a pyramid by inserting a new attribute called "*Persistent*" and connecting it to the other corners. The new attribute was proposed to signify the longevity of each name [64]. This proposal to change the shape of Zooko's triangle can be another potential topic for research which could give Petname Systems additional security properties.

Smart phones are becoming increasingly popular and the number of people that access the Internet from their smart phones is growing every day. Investigations on how the Petname Model can be implemented and adapted for the tiny screen of a mobile phone can be a challenging task and another scope for future research.

7.3 Conclusion

Current security measures found in the browser are often proved to be futile due to the lack of proper security usability which makes identity theft, in particular Phishing attacks successful. An alternative approach is necessary which helps users to identify the identities of the service providers in a better way.

The Petname Model is naturally embedded in human perception to identify different entities. Implementing it in computer networks and system is a natural extension of human cognitive capabilities and represents a great aid for humans in digital environments. This fact has been demonstrated through several applications, experiments and proposals. A large scale adaptation of the Petname Model is therefore timely.

References

[1] Gartner Survey on Phishing Attack. December, 2007
http://www.gartner.com/it/page.jsp?id=565125

[2] Marc Stiegler: An Introduction to Petname Systems. 2005
http://www.skyhunter.com/marcs/petnames/IntroPetNames.html

[3] Wikipedia entry on entity. Accessed on May 25, 2009.
http://en.wikipedia.org/wiki/Entity

[4] Thanh, D.V., Jørstad, I.: The ambiguity of identity. Telektronikk issue on Identity Management 103, (No.3/4-2007, ISSN:0085-7130), 3--10 (2007)

[5] Jøsang, A., Pope, S.: User centric identity management. In: Asia Pacific Information Technology Security Conference, AusCERT2005, Austrailia, pp. 77--89 (2005)

[6] Jøsang, A., Al Zomai, M., Suriadi, S.: Usability and privacy in identity management architectures. In: L. Brankovic, C. Steketee (eds.) Fifth Australasian Information Security Workshop (Privacy Enhancing Technologies) (AISW 2007), {CRPIT}, vol. 68, pp. 143--152. ACS, Ballarat, Australia (2007)

[7] Wikipedia entry on service provider. Accessed on May 25, 2009
http://en.wikipedia.org/wiki/Service_provider

[8] OASIS. Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0. Oasis Standard, 15 March, 2005
http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf

[9] Liberty ID-FF Architecture Overview Version: 1.2-errata-v1.0
http://www.projectliberty.org/liberty/content/download/318/2366/file/draft-liberty-idff-arch-overview-1.2-errata-v1.0.pdf

[10] Shibboleth Project. Shibboleth Architecture Protocols and Profiles. Working Draft 05, 23 November, 2004. Internet2/MACE, 2004

[11] Microsoft Windows CardSpace
http://www.microsoft.com/windows/products/winfamily/cardspace/default.mspx

[12] Higgins- Open Source Identity Framework
http://www.eclipse.org/higgins/index.php

[13] OpenID
http://openid.net/

[14] SourceID-Open Source Federated Identity Management
http://www.sourceid.org/

[15] DotGNU Virtual Identities
http://www.gnu.org/software/dotgnu/auth.html

[16] Wikipedia entry on Identity Theft, accessed on 28th May 2009
http://en.wikipedia.org/wiki/Identity_theft

[17] Sasha Romanosky: Do Data Breach Disclosure Laws Reduce Identity Theft? May, 2008
http://www.heinz.cmu.edu/research/241full.pdf

[18] Phishing: Language Log
http://itre.cis.upenn.edu/~myl/languagelog/archives/001477.html

[19] Tom Spring: Spam Slayer: Do You Speak Spam? November, 2003
http://www.pcworld.com/article/113431/spam_slayer_do_you_speak_spam.html

[20] Wikipedia entry on Phishing, accessed on 28th May, 2009
http://en.wikipedia.org/wiki/Phishing

[21] Jøsang, A., AlFayyadh, B., Grandison, T., AlZomai, M., McNamara, J.: Security usability principles for vulnerability analysis and risk assessment. In: Proceedings of the Annual Computer Security Applications Conference (ACSAC´07), Miami Beach, December 2007

[22] Mutton, Paul: PayPal Security Flaw allows Identity Theft. Netcraft. June, 2006. http://news.netcraft.com/archives/2006/06/16/paypal_security_flaw_allows_identity_theft.html

[23] Hoffman, Patrick: RSA Catches Financial Phishing Kit. eWeek. January, 2007. http://www.eweek.com/c/a/Security/RSA-Catches-Financial-Phishing-Kit/

[24] Rachna Dhamija, J.D. Tygar: The Battle Against Phishing: Dynamic Security Skins. Symposium On Usable Privacy and Security (SOUPS) 2005.

[25] Yahoo! Service log-in page https://login.yahoo.com/

[26] Wikipedia entry on identity management, accessed on April 15, 2009 http://en.wikipedia.org/wiki/Identity_management

[27] Wilcox-O'Hearn, Z.: Names: Decentralized, secure, human-meaningful: Choose two. 2005 http://www.zooko.com/distnames.html

[28] Internet archive wayback machine: snapshot on Zooko's writing. http://web.archive.org/web/*/http://zooko.com/distnames.html

[29] Miller, M.: Lambda for humans. 2001 http://www.erights.org/elib/capability/pnml.html

[30] Shirky, C.: Domain names: Memorable, global, non-political? 2002 http://shirky.com/writings/domain_names.html

[31] Shapiro, J.S.: Pet names, true names, and nicknames. 2000 http://www.eros-os.org/~majordomo/dcms-dev/0036.html

[32] Close, T.: Naming vs. pointing. 2003 http://www.waterken.com/dev/YURL/Analogy/

[33] Close, T.: Waterken yurl: Trust management for humans. 2003 http://www.waterken.com/dev/YURL/Name/

[34] Dhamija, R., Tygar, J.D.: Why phishing works. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 581--590. ACM Press (2006)

[35] Close, T.: Petname tool: Enabling web site recognition using the existing ssl infrastructure. http://www.w3.org/2005/Security/usability-ws/papers/02-hp-petname/

[36] Trustbar Firefox addon. http://u.cs.biu.ac.il/~herzbea/TrustBar/

[37] Yee, K.P., Sitaker, K.: Passpet: convenient password management and phishing protection. In: SOUPS, pp. 32--43 (2006)

[38] Capdesk. http://www.skyhunter.com/marcs/CapDeskSpec.html

[39] Stiegler, M., Karp, A.H., Yee, K.P., Miller, M.: Polaris: Virus safe computing for windows XP. 2004 http://www.hpl.hp.com/techreports/2004/HPL-2004-221.pdf

[40] M A Sasse, S Brostoff and D Weirich: Transforming the 'weakest link' — a human/computer interaction approach to usable and effective security. BT Technology Journal, 2001;19(3):122—31

[41] Schneier B: Secrets and Lies, John Wiley and Sons (2000)

[42] Alma Whitten and J. D. Tyger: Usability of Security: A Case Study. December 2008 http://reports-archive.adm.cs.cmu.edu/anon/anon/1998/CMU-CS-98-155.pdf

[43] Whitten, A., Tygar, J.D.: Why Johnny can't encrypt: A usability evaluation of PGP 5.0. In: 8th USENIX Security Symposium (1999)

[44] Close, T.: Petname tool 1.6. https://addons.mozilla.org/en-US/firefox/addon/957

[45] Drelie Gelasca, E., Tomasic, D., Ebrahimi, T.: Which Colors Best Catch Your Eyes: a Subjective Study of Color Saliency. In: First International Workshop on Video Processing

[46] ITU. Message Sequence Charts (MSC). Ø Haugen (ed.). Geneva, 1999. (Recommendation Z.120, p. 126.)

[47] Bræk, R. and Haugen, Ø.: Engineering Real Time Systems. An object-oriented methodology using SDL. Hemel Hempstead: Prentice Hall. ISBN 0-13-034448-6, 1993

[48] Z.100 (1994), CCITT Specification and description language (SDL), ITU-T

[49] Wikipedia entry on Add-on (Mozilla), accessed on 3rd June, 2009 http://en.wikipedia.org/wiki/Add-on_(Mozilla)

[50] Mozilla Developer Center https://developer.mozilla.org/En

[51] Wikipedia entry on CSS, accessed on 3rd June, 2009 http://en.wikipedia.org/wiki/Cascading_Style_Sheets

[52] Wikipedia entry on DOM, accessed on 3rd June, 2009 http://en.wikipedia.org/wiki/Document_Object_Model

[53] Wikipedia entry on JavaScript, accessed on 3rd June, 2009 http://en.wikipedia.org/wiki/Javascript

[54] XPCom documentation page in Mozilla Development Center https://developer.mozilla.org/en/XPCOM

[55] Wikipedia entry on XPConnect, accessed on 3rd June, 2009 http://en.wikipedia.org/wiki/XPConnect

[56] XPI page in Mozilla Development Center https://developer.mozilla.org/en/XPI

[57] RDF Primer. W3C Recommendation, 10 February, 2004 http://www.w3.org/TR/rdf-primer/

[58] XUL page in Mozilla Development Center https://developer.mozilla.org/en/XUL

[59] Chapter 3: Introduction to XUL—How to build a more intuitive UI https://developer.mozilla.org/En/Firefox_addons_developer_guide/Introduction_to_XUL—How_to_build_a_more_intuitive_UI

[60] Wikipedia entry on XUL, accessed on 3rd June, 2009 http://en.wikipedia.org/wiki/XUL

[61] Wikipedia entry on Document Type Definition, accessed on 3rd June, 2009 http://en.wikipedia.org/wiki/Document_Type_Definition

[62] Tutorial on Firefox extension development on Mozilla Development Center https://developer.mozilla.org/En/Firefox_addons_developer_guide/Let's_build_a_Firefox_extension

[63] XUL Overlays on Mozilla Development Center https://developer.mozilla.org/en/XUL_Overlays

[64] Wyman, B.: The persistence of identity. 2006 http://www.wyman.us/main/2006/12/the_persistence.html

APPENDICES

APPENDIX A: chrome.manifest

```
content   unipet                          jar:chrome/unipet.jar!/content/
skin      unipet        classic/1.0       jar:chrome/unipet.jar!/skin/
locale    unipet        en                jar:chrome/unipet.jar!/locale/en/

overlay chrome://browser/content/browser.xul chrome://unipet/content/overlay.xul
style chrome://browser/content/browser.xul chrome://unipet/skin/unipet.css
style chrome://global/content/customizeToolbar.xul chrome://unipet/skin/unipet.css
```

APPENDIX B: install.rdf

```
<?xml version="1.0"?>

<RDF xmlns="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:em="http://www.mozilla.org/2004/em-rdf#">

  <Description about="urn:mozilla:install-manifest">
    <em:id>unipet@example.net</em:id>
    <em:version>0.99</em:version>
    <em:type>2</em:type>

    <!-- Target Application this extension can install into,
         with minimum and maximum supported versions. -->
    <em:targetApplication>
      <Description>
        <em:id>{ec8030f7-c20a-464f-9b0e-13a3a9e97384}</em:id>
        <em:minVersion>3.0</em:minVersion>
        <em:maxVersion>3.0.*</em:maxVersion>
      </Description>
    </em:targetApplication>

    <!-- Front End MetaData -->
    <em:name>UniPet</em:name>
    <em:description>Manage your website identities more securely with the UniPet</em:description>
    <em:creator>Md. Sadek Ferdous</em:creator>
    <em:homepageURL>http://www.example.com/</em:homepageURL>
  </Description>
</RDF>
```

APPENDIX C: overlay.xul

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/css" href="chrome://unipet/skin/"?>
<?xml-stylesheet type="text/css" href="chrome://unipet/skin/unipet.css"?>
<overlay id="unipetOverlay" xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">
        <stringbundleset>
        <stringbundle id="unipet-properties" src="chrome://unipet/locale/unipet.properties"/>
        </stringbundleset>
        <toolbar id="toolbar-menubar" insertafter="main-menubar">
        <hbox id="pet_hbox" align="center" style="border: 1px solid ;">
        <label id="name" class="pet_label" tooltiptext="Enter a name/Select an image and click apply">Petname:</label>
        <textbox id="unipet-text" class="unipet-disabled" readonly="true" size="18" onfocus="tBFocus()"/>
        <button id="apply_button" onclick="bCommand()" image="chrome://unipet/skin/tick-circle-frame.png" label="  Apply"/>
        <label id="suggestion" class="suggest_label" image="chrome://unipet/skin/light-bulb--arrow.png" >Suggestion:</label>
        <textbox id="unipet-suggest" class="unipet-disabled" readonly="true" size="18" tooltiptext="Suggestion reagarding
Petname will be displayed here"/>
        <button id="options_bar" label="Options" image="chrome://unipet/skin/wrench-screwdriver.png" dir="normal"
orient="horizontal" type="menu" popup="button-popup" context="button-popup" tooltiptext="Click here for more options"/>
        <menupopup id="button-popup">
        <menuitem id="text" class="menuitem-iconic" image="chrome://unipet/skin/edit.png" label="Using Text"
onclick="useText(false)"/>
        <menuitem id="image" class="menuitem-iconic" image="chrome://unipet/skin/image-2.png" label="Use Image"
onclick="useImage(false)"/>
        <menuseparator />
        <menuitem class="menuitem-iconic" image="chrome://unipet/skin/cross-circle-frame.png" label="Delete All Petnames"
onclick="delPetname()"/>
        <menuitem class="menuitem-iconic" image="chrome://unipet/skin/eraser--minus.png" label="Delete All Petlogos"
onclick="delPetlogo()"/>
        <menuseparator />
        <menuitem class="menuitem-iconic" image="chrome://unipet/skin/lifebuoy--pencil.png" label="Help" onclick="help()"/>
        <menuitem class="menuitem-iconic" image="chrome://unipet/skin/question-frame.png" label="About" onclick="about()"/>
        </menupopup>
        </hbox>
        </toolbar>
        <script type="application/x-javascript" src="unipet.js"/>
</overlay>
```

APPENDIX D: collision.xul

```
<?xml version="1.0"?>
<?xml-stylesheet href="chrome://global/skin/" type="text/css"?>
<!DOCTYPE window SYSTEM "chrome://unipet/locale/unipet-collision.dtd">

<dialog xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul"
        title="&dialogTitle;"
        defaultButton="cancel">
 <vbox align="start">
  <description value="&similarDescription;"/>
  <vbox id="petname-collision-similar">
   <hbox pack="start" align="center">
    <button dlgtype="extra1" label="&navigate;" oncommand="return true;"/>
    <label class="created" value="&createdOn;"/>
   </hbox>
  </vbox>
  <description value="&cancelDescription;"/>
  <button dlgtype="cancel" id="petname-collision-cancel" label="&cancel;"/>
 </vbox>
 <script type="application/x-javascript" src="collision.js"/>
</dialog>
```

APPENDIX E: same.xul

```
<?xml version="1.0"?>
<?xml-stylesheet href="chrome://global/skin/" type="text/css"?>
<!DOCTYPE window SYSTEM "chrome://unipet/locale/unipet-similar.dtd">

<dialog xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul"
        title="&dialogTitle;"
        defaultButton="cancel">
 <vbox align="start">
  <description value="&similarDescription;"/>
  <vbox id="petname-collision-similar">
   <hbox pack="start" align="center">
    <button dlgtype="extra1" label="&navigate;" oncommand="return true;"/>
    <label class="created" value="&createdOn;"/>
   </hbox>
  </vbox>
  <description value="&cancelDescription;"/>
  <button dlgtype="cancel" id="petname-collision-cancel" label="&cancel;"/>
 </vbox>
 <script type="application/x-javascript" src="collision.js"/>
</dialog>
```

APPENDIX F: aboutDialog.xul

```
<?xml version="1.0"?> <!-- -*- Mode: HTML -*- -->
<?xml-stylesheet href="chrome://global/skin/" type="text/css"?>
<!DOCTYPE window SYSTEM "chrome://unipet/locale/about.dtd">
<dialog xmlns:html="http://www.w3.org/1999/xhtml"
      xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul"
      id="aboutDialog"
      title="&dialogTitle;"
      buttons="accept">

            <vbox id="detailsBox" align="center" flex="1">
            <spacer flex="1"/>
            <textbox id="userAgent" multiline="true" readonly="true" cols="60" rows="8" value="&Description;"/>
            </vbox>
            <vbox flex="1" id="creditsBox">
            <description id="creditsDesc">
                  Developed by: Md. Sadek Ferdous as part of his Master's Thesis.
            </description>
            <description id="place">
                  Developed at: University Graduate Center in Kjeller (UNIK).
            </description>
            <description id="email">
                  Contact:sadek@unik.no
            </description>
            <description id="version">
                  Version: 0.99
            </description>
            </vbox>
            <separator class="groove" id="groove"/>
</dialog>
```

APPENDIX G: unipet.properties

```
bookmarks.title=unipet
init.title=
init.tooltip=No Site to apply Petname.
null.title=unknown site
null.tooltip=Apply a Petname for this site.
disabled.title=
disabled.tooltip=No Site to apply Petname.
enabled.title=unknown site
enabled.tooltip=Assign a petname to this site before exchanging sensitive information.
edit.tooltip=Edit the previously applied Petname.
```

APPENDIX H: unipet.css

```
.unipet-disabled {
    -moz-appearance: none;
    text-align: center;
    font-style: italic;
    font-weight: bold;
}

.unipet-enabled {
		-moz-appearance: none;
		text-align: center;
		font-style: italic;
		font-weight: bold;
		background-color: #FF6262;
}

.unipet-populated {
		-moz-appearance: none;
		text-align: center;
		font-weight: bold;
		background-color: #80FF80;
}

.sug-box-enabled{
		-moz-appearance: none;
		text-align: center;
		background-color:#75DDFF;
}

.sug-box-disabled{
		-moz-appearance: none;
		text-align: center;
		background-color:#F2F2F2;
}
.label-disabled{
		-moz-appearance: none;
		background-color:#F2F2F2;
}

.label-enabled{
		-moz-appearance: none;
		color:#990000;
}
.pet_label{
		-moz-appearance: none;
		font-weight: bold;
		color: #336666;
}
```

APPENDIX I: unipet-collision.dtd

```
<!ENTITY dialogTitle
 "You are using a similar petname for another site.">
<!ENTITY similarDescription
 "You are using a similar petname for another site.">
<!ENTITY navigate "Go to &amp;quot;$petname&amp;quot;">
<!ENTITY createdOn "created: ">
<!ENTITY cancelDescription
 "Cancel this warning if you want to choose a different petname for this site.">
<!ENTITY cancel "Don't assign petname">
```

APPENDIX J: unipet-similar.dtd

```
<!ENTITY dialogTitle "You are using a same petname for another site.">
<!ENTITY similarDescription
 "You are using a same petname for another site.">
<!ENTITY navigate "Go to &amp;quot;$petname&amp;quot;">
<!ENTITY createdOn "created: ">
<!ENTITY cancelDescription "Cancel this warning if you want to choose a different petname for this site.">
<!ENTITY cancel "Don't assign petname">
```

APPENDIX K: about.dtd

```
<!ENTITY dialogTitle "About UniPet">
<!ENTITY Description
 "This Firefox extension is developed to help users to identify a website properly. In Phishing attack, attackers deploy a fraudulent website with the similar look and feel of the genuine site and thus making it hard for the users to differentiate between the genuine site and the fraudulent one. After careful validation, a user can apply Petname/Petlogo for a website which will be displayed if he visits it later on. If the respective Petname/Petlogo is missing, the user might conclude that the site is a fraudulent one. For more information please consult the Help page which can be opened from the Options Menu.">
```

APPENDIX L: collision.js

```
(function () {
   const arg = window.arguments[0];
   const cancel = document.getElementById('petname-collision-cancel');

   function go(uri) {
      return function () {
         cancel.click();
         arg.navigate(uri);
      }; }
```

```
    const similar = document.getElementById('petname-collision-similar');
    const petnameDescriptionPrototype = similar.removeChild(similar.firstChild);
    for (var i = 0; i !== arg.similar.length; ++i) {
      const petname = arg.similar[i];
      const description = similar.appendChild(petnameDescriptionPrototype.cloneNode(true));
      const link = description.getElementsByTagName('button')[0];
      link.label = link.label.replace('$petname', petname.title);
      link.addEventListener('click', go(petname.uri), false);
      description.getElementsByAttribute('class', 'created')[0].value +=
        new Date(petname.dateAdded / 1000).toLocaleDateString();
    }
}) ();
```

APPENDIX M: aboutDialog.js

```
var gSelectedPage = 0;

function init(aEvent)
{
        if (aEvent.target != document)
                return;
        var userAgentField = document.getElementById("userAgent");
        userAgentField.value = navigator.userAgent;

        var button = document.documentElement.getButton("extra2");
        button.setAttribute("label", document.documentElement.getAttribute("creditslabel"));
        button.addEventListener("command", switchPage, false);
        document.documentElement.getButton("accept").focus();
}

function uninit(aEvent)
{
        if (aEvent.target != document)
                return;
        var iframe = document.getElementById("creditsIframe");
        iframe.setAttribute("src", "");
}

function switchPage(aEvent)
{
        var button = aEvent.target;
        if (button.localName != "button")
                return;
        modes.setAttribute("selectedIndex", ((1 + gSelectedPage) % 2));
}
```

APPENDIX N: unipet.js

```
/*This is the main JavaScript file for the UniPet. It follows the sturcture of the Petname tool and has re-used the code of the some
features found in the Petname tool, developed by Tyler Close
and updated them with new features of the UniPet. For example: the UniPet resued the structure of model, viewer, navigate, controller
and listener found in the Petname tool. However, the level of complexity in terms of functionalities in each of them are much higher
than those of the Petname tool as the UniPet has added new complex, absent in the Petname tool, features onto them
*/
var prfefs,properties;
const err = false && Cc['@mozilla.org/consoleservice;1'].getService(Ci.nsIConsoleService);
   err && err.logStringMessage('unipet: loading...');

              var rootFold;
              var title;

              var wasVersion = 0;
              var wasURL;
              var wasSecure;
              var wasPetname;

              const model = function () { //model is used for various functions such as to find similar petnames, to save , delete or change
petnames in bookmark folder, mostly influenced from the Petname tool.
   const URI = Cc['@mozilla.org/network/io-service;1'].getService(Ci.nsIIOService);
   const TLD = Cc["@mozilla.org/network/effective-tld-service;1"].getService(Ci.nsIEffectiveTLDService);
   const bookmarks = Cc['@mozilla.org/browser/nav-bookmarks-service;1'].getService(Ci.nsINavBookmarksService);
const history = Cc['@mozilla.org/browser/nav-history-service;1'].getService(Ci.nsINavHistoryService);

              function forEachInContainer(op, container) {
      container.containerOpen = true;
      for (var i = 0; i < container.childCount; ++i) {
         const child = container.getChild(i);
         switch (child.type) {
            case Ci.nsINavHistoryResultNode.RESULT_TYPE_URI: {
               const r = op(child);
               if (undefined !== r) { return r; }
            } break;
            case Ci.nsINavHistoryResultNode.RESULT_TYPE_FOLDER: {
               const rr = forEachInContainer(op, child.QueryInterface(
                  Ci.nsINavHistoryContainerResultNode));
               if (undefined !== rr) { return rr; }
            } break;
         }
      }
         return undefined;
      }

   function forEachPetname(op) {
      const q = history.getNewQuery();
```

```
    q.setFolders([ rootFold ], 1);
    return forEachInContainer(op, history.executeQuery(q,history.getNewQueryOptions()).root);
}

function eyeball(text) {
    text = text.toLowerCase();          // case-insensitive
    text = text.replace(/\s/g, '');     // collapse whitespace
    text = text.replace(/0/g, 'o');     // digit zero as letter O
    text = text.replace(/1/g, 'l');     // digit one as letter L
    text = text.replace(/i/g, 'l');     // letter I as letter L
    text = text.replace(/n/g, 'm');     // letter N as letter M

    // construct a list of all unique characters in the text
    var charSet = {};
    for (var i = text.length; 0 !== i--;) {
        charSet[text.charAt(i)] = true;
    }
    var charList = [];
    for (var k in charSet) {
        if (charSet.hasOwnProperty(k)) {
            charList.push(k);
        }
    }
    charList.sort();
    return charList.join('');
}
function confusing(aEye, bEye) {
    var longer;
    var shorter;
    if (aEye.length > bEye.length) {
    longer = aEye;
    shorter = bEye;
    } else {
        longer = bEye;
        shorter = aEye;
      }
    for (var i = shorter.length; 0 !== i--;) {
        if (-1 === longer.indexOf(shorter.charAt(i))) { return false; }
    }
    return true;
}
    function eTLD(uri) {
    try {
                        if ('localhost' === uri.asciiHost) { return uri.hostPort; }
                        const r = TLD.getBaseDomain(uri);
                        if ('yurl.net' === r) { return TLD.getBaseDomain(uri, 1); }
            return r;
    } catch (e) {
        return uri.asciiHost;} }
```

```
return {
    findPetname: function (uri) {
    const host = eTLD(uri);
    return forEachPetname(function (bookmark) {
        if (host === eTLD(URI.newURI(bookmark.uri, null, null))) {
            return bookmark;
            }
        });                },

                        createPetname: function (title, url) {
                                const id = bookmarks.insertBookmark(rootFold,url,bookmarks.DEFAULT_INDEX, title);
                                return forEachPetname(function (bookmark) {
                                        if (bookmark.itemId === id) { return bookmark; }
                                        });
                        },

                        changePetnameTitle: function (petname, title) {
                                bookmarks.setItemTitle(petname.itemId, title);
                        },

                        deletePetname: function (petname) {
                                bookmarks.removeItem(petname.itemId);
                        },

                        findSimilarPetnames: function (title, except) {
                                const titleEye = eyeball(title);
                                const exceptId = except ? except.itemId : undefined;
                                const r = [];
                                forEachPetname(function (bookmark) {
                                        if (exceptId !== bookmark.itemId && confusing(titleEye, eyeball(bookmark.title))) {
                                                r.push(bookmark);
                                        }
                                });
                                return r;
                        }
};
} ();
const viewer = function () { //this updates the UniPet at different situations
    return {
            init: function () { //this initializes the UniPet when the browser loads
            err && err.logStringMessage('unipet: viewer init');
            const text = document.getElementById('unipet-text');
             if(prefs.getBoolPref('unipet.useText'))
                { if (text) {
                                text.onchange = undefined;
                                text.readOnly = true;
                                text.setAttribute('class', 'unipet-disabled');
                                text.value = properties.getString('init.title');
                                text.setAttribute('tooltiptext',properties.getString('init.tooltip'));
```

```
                text.onfocus = undefined;
        }
        }
    },

        null: function (isSec,stat,dom) {  //this is called when a page is loaded and no Petname or Petlogo found
        err && err.logStringMessage('unipet: viewer disable');
        const text = document.getElementById('unipet-text');
        const sug=document.getElementById('unipet-suggest');

        var file = Components.classes["@mozilla.org/file/directory_service;1"].
        getService(Components.interfaces.nsIProperties).
        get("ProfD", Components.interfaces.nsIFile);
        file.append("domain.txt");
        if( !file.exists())
        file.create(Components.interfaces.nsIFile.NORMAL_FILE_TYPE, 0666);


        var data,tmp_data;

        if(isSec)
        {
                tmp_data=dom+" "+stat.serverCert.organization+'\r\n';
        }
        else
        {
                tmp_data=dom+" "+gBrowser.contentDocument.title+'\r\n';
        }

        var foStream = Components.classes["@mozilla.org/network/file-output-stream;1"].
          createInstance(Components.interfaces.nsIFileOutputStream);
        var istream = Components.classes["@mozilla.org/network/file-input-stream;1"].
          createInstance(Components.interfaces.nsIFileInputStream);
        if(tmp_data.lastIndexOf(" ")!=(tmp_data.length-2)-1)
        {
                istream.init(file, 0x01, 0444, 0);
                istream.QueryInterface(Components.interfaces.nsILineInputStream);
                var flag_val=true,tmp_str,tmp_str2;

                var line = {}, lines = [], hasmore;
                do {
                        hasmore = istream.readLine(line);
                        lines.push(line.value);
                        tmp_str=lines.join();
                        if((tmp_str.substring(0,tmp_str.indexOf(" "))).match(dom))
                        {
                                flag_val=false;
                                sug.setAttribute('class','sug-box-enabled');
                                sug.value=tmp_str.substring(tmp_str.indexOf(" "),tmp_str.length);
```

```
                        break;
                }
                lines = [];
                } while(hasmore);

                istream.close();

                if(flag_val)
                {
                        foStream.init(file, 0x02 | 0x08 | 0x10, 0666, 0);

                        if(isSec)
                        {
                                sug.setAttribute('class','sug-box-enabled');
                                sug.value=stat.serverCert.organization;

                        }
                        else
                        {
                                sug.setAttribute('class','sug-box-enabled');
                                sug.value=gBrowser.contentDocument.title;
                        }

                data=dom+" "+sug.value+'\r\n';

                var converter = Components.classes["@mozilla.org/intl/converter-output-stream;1"].
                createInstance(Components.interfaces.nsIConverterOutputStream);
                converter.init(foStream, "UTF-8", 0, 0);
                converter.writeString(data);
                converter.close();
                }
                }

        if (text && prefs.getBoolPref('unipet.useText')) {//if the UniPet is in the Petname mode do this....
                text.onchange = undefined;
                text.readOnly =false;
                text.setAttribute('class', 'unipet-enabled');
                text.value = properties.getString('null.title');
                text.setAttribute('tooltiptext',properties.getString('null.tooltip'));
                text.onfocus = undefined;

              }
else if(!prefs.getBoolPref('unipet.useText')) //if the UniPet is in the Petlogo mode do this....
              {
                        const imageLabel = document.getElementById('imageLabel');
                        const petLabel = document.getElementById('petLogo');
                        const label = document.getElementById('name');
                        var element = document.getElementById('pet_hbox');
                        var parentDiv = label.parentNode;
```

```
        if(petLabel)
        {
                element.removeChild(petLabel);
                var sp1 = document.createElement("label");
                sp1.setAttribute('id','imageLabel');
                sp1.setAttribute('style','border: 1px solid;');
                sp1.setAttribute('value','Add Image');
                sp1.setAttribute('class','label-enabled');
                sp1.setAttribute('tooltiptext','Click the browse button to add an image');
                parentDiv.insertBefore(sp1,document.getElementById('browse_button'));
        }
        else if(imageLabel)
        {
                imageLabel.setAttribute('class','label-enabled');
                imageLabel.value="Add Image";
        }       }
},

     edit: function () {//this is called when users want to edit the Petname...
    err && err.logStringMessage('unipet: viewer disable');
    const text = document.getElementById('unipet-text');
    if (text) {
      text.onchange = undefined;
      text.readOnly =false;
      text.setAttribute('class', 'unipet-enabled');
      text.setAttribute('tooltiptext',properties.getString('edit.tooltip'));
    }
  },

  disable: function () { //this sets up the GUI of the UniPet when it is used in the Petname mode and no page is loaded...
    err && err.logStringMessage('unipet: viewer disable');
    const text = document.getElementById('unipet-text');
        const sug=document.getElementById('unipet-suggest');

    if (text && prefs.getBoolPref('unipet.useText')) {
      text.onchange = undefined;
      text.readOnly = true;
      text.setAttribute('class', 'unipet-disabled');
      text.value = properties.getString('disabled.title');
      text.setAttribute('tooltiptext',properties.getString('disabled.tooltip'));
      text.onfocus = undefined;
      sug.setAttribute('class','sug-box-disabled');
      sug.value="";
    }
     else if(!prefs.getBoolPref('unipet.useText'))
     {
          const imageLabel = document.getElementById('imageLabel');
          const petLabel = document.getElementById('petLogo');
          const label = document.getElementById('name');
```

```
            var element = document.getElementById('pet_hbox');
            var parentDiv = label.parentNode;
            if(petLabel && !imageLabel)
            {
                        element.removeChild(petLabel);
                        var sp1 = document.createElement("label");
                        sp1.setAttribute('id','imageLabel');
                        sp1.setAttribute('style','border: 1px solid;');
                        sp1.setAttribute('class','label-disabled');
                        sp1.setAttribute('value','Image Disabled');
                        sp1.setAttribute('tooltiptext','Click the browse button to add an image');
                        parentDiv.insertBefore(sp1,document.getElementById('browse_button'));
            }
            else if(!petLabel && imageLabel)
            {
                        imageLabel.value="Image Disabled";
            }
            sug.setAttribute('class','sug-box-disabled');
            sug.value="";
            }       },
    showPetname: function (petname,isSec,stat,dom) { //this is called to display the Petname or Petlogo
       err && err.logStringMessage('unipet: viewer populate');
       const text = document.getElementById('unipet-text');
            const sug=document.getElementById('unipet-suggest');
            var data,tmp_data;
            var file = Components.classes["@mozilla.org/file/directory_service;1"].
        getService(Components.interfaces.nsIProperties).
        get("ProfD", Components.interfaces.nsIFile);
            file.append("domain.txt");
            if( !file.exists())
            file.create(Components.interfaces.nsIFile.NORMAL_FILE_TYPE, 0666);

            if(isSec)
            {
                        tmp_data=dom+" "+stat.serverCert.organization+'\r\n';
            }
            else
            {
                        tmp_data=dom+" "+gBrowser.contentDocument.title+'\r\n';
            }

            var foStream = Components.classes["@mozilla.org/network/file-output-stream;1"].
        createInstance(Components.interfaces.nsIFileOutputStream);
            var istream = Components.classes["@mozilla.org/network/file-input-stream;1"].
        createInstance(Components.interfaces.nsIFileInputStream);
            if(tmp_data.lastIndexOf(" ")!=(tmp_data.length-2)-1)
            {
                        istream.init(file, 0x01, 0444, 0);
                        istream.QueryInterface(Components.interfaces.nsILineInputStream);
```

```
var flag_val=true,tmp_str,tmp_str2;

var line = {}, lines = [], hasmore;
do {
        hasmore = istream.readLine(line);
        lines.push(line.value);
        tmp_str=lines.join();
        if((tmp_str.substring(0,tmp_str.indexOf(" "))).match(dom))
        {
                flag_val=false;
                sug.setAttribute('class','sug-box-enabled');
                sug.value=tmp_str.substring(tmp_str.indexOf(" "),tmp_str.length);
                break;
        }
        lines = [];
}while(hasmore);
istream.close();
if(flag_val)
{
        foStream.init(file, 0x02 | 0x08 | 0x10, 0666, 0);
        if(isSec)
        {
                sug.setAttribute('class','sug-box-enabled');
                sug.value=stat.serverCert.organization;
        }
        else
        {
                sug.setAttribute('class','sug-box-enabled');
                sug.value=gBrowser.contentDocument.title;
        }
        data=dom+" "+sug.value+'\r\n';
        var converter = Components.classes["@mozilla.org/intl/converter-output-stream;1"].
        createInstance(Components.interfaces.nsIConverterOutputStream);
        converter.init(foStream, "UTF-8", 0, 0);
        converter.writeString(data);
        converter.close();
}
}
if (text && prefs.getBoolPref('unipet.useText')) {
text.onchange = undefined;
        text.readOnly = false;
        text.setAttribute('class', 'unipet-populated');
        text.value = petname.title;
        text.setAttribute('tooltiptext',properties.getString('populated.tooltip'));
text.onfocus = text.select;
}
 else if(!prefs.getBoolPref('unipet.useText'))
 {
        const petLabel = document.getElementById('petLogo');
```

```
            const label = document.getElementById('name');
            const imageLabel = document.getElementById('imageLabel');
            var element = document.getElementById('pet_hbox');
            var parentDiv = label.parentNode;
            if(imageLabel)element.removeChild(imageLabel);
            var URL;

            var file2 = Components.classes["@mozilla.org/file/directory_service;1"].
            getService(Components.interfaces.nsIProperties).
            get("ProfD", Components.interfaces.nsIFile);
            file2.append("petlogo.txt");
            if( !file2.exists())
            file2.create(Components.interfaces.nsIFile.NORMAL_FILE_TYPE, 0666);

            var istream = Components.classes["@mozilla.org/network/file-input-stream;1"].
            createInstance(Components.interfaces.nsIFileInputStream);
            istream.init(file2, 0x01, 0444, 0);
            istream.QueryInterface(Components.interfaces.nsILineInputStream);
            var line = {}, lines = [], hasmore;
            do {
                        hasmore = istream.readLine(line);
                        lines.push(line.value);
                        tmp_str=lines.join();
                        if((tmp_str.substring(0,tmp_str.indexOf(" "))).match(dom))
                        {
                                    URL=tmp_str.substring(tmp_str.indexOf(" "),tmp_str.length);
                                    break;
                        }
                        lines = [];
            } while(hasmore);

            istream.close();
            if(!petLabel)
            {
                        var sp1 = document.createElement("image");
                        sp1.setAttribute('id','petLogo');
                        sp1.setAttribute('height','26');
                        sp1.setAttribute('width','26');
                        sp1.setAttribute('src',URL);
                        parentDiv.insertBefore(sp1,document.getElementById('browse_button'));
            }

            if(petLabel)
            {
                        petLabel.setAttribute('src',URL);
            }           }           },
alertCollisions: function (proposed, similar) {             //this is called when there is similar Petname already assigned
            err && err.logStringMessage('unipet: viewer alert');
            window.openDialog('chrome://unipet/content/collision.xul','', 'chrome,centerscreen,modal', {
```

```
                proposed: proposed,
                similar: similar,
                navigate: function (uri) {
                    window.document.commandDispatcher.advanceFocus();
                    gBrowser.contentWindow.location = uri;
                }
            });
        },

alertSame: function (proposed, similar) {     //this is called when there is same Petname already assigned
            err && err.logStringMessage('unipet: viewer alert');
            window.openDialog('chrome://unipet/content/same.xul','', 'chrome,centerscreen,modal', {
            proposed: proposed,
            similar: similar,
            navigate: function (uri) {
                    window.document.commandDispatcher.advanceFocus();
                    gBrowser.contentWindow.location = uri;
                }
            });
        }
    };
} ();


window.addEventListener('load', function () {              //this is the entry point of the UniPet that runs when the browser loads
            const pref = Cc['@mozilla.org/preferences-service;1'].getService(Ci.nsIPrefService);
            const prop = document.getElementById('unipet-properties');
            prefs=pref;
            properties=prop;

            const URI = Cc['@mozilla.org/network/io-service;1'].getService(Ci.nsIIOService);
const TLD = Cc["@mozilla.org/network/effective-tld-service;1"].getService(Ci.nsIEffectiveTLDService);
            const bookmarks = Cc['@mozilla.org/browser/nav-bookmarks-service;1'].getService(Ci.nsINavBookmarksService);
    const history = Cc['@mozilla.org/browser/nav-history-service;1'].getService(Ci.nsINavHistoryService);
            var file1 = Components.classes["@mozilla.org/file/directory_service;1"].
        getService(Components.interfaces.nsIProperties).get("ProfD", Components.interfaces.nsIFile);
            var file2 = Components.classes["@mozilla.org/file/directory_service;1"].
        getService(Components.interfaces.nsIProperties).get("ProfD", Components.interfaces.nsIFile);

            file1.append("Petname_suggestion.txt");
            file2.append("Petlogo_suggestion.txt");
            if(file1.exists())
            {
                    file1.remove(true);
                    file1.create(Components.interfaces.nsIFile.NORMAL_FILE_TYPE, 0666);
                    }
            else file1.create(Components.interfaces.nsIFile.NORMAL_FILE_TYPE, 0666);
            if(file2.exists()){
                                file2.remove(true);
```

```
            file2.create(Components.interfaces.nsIFile.NORMAL_FILE_TYPE, 0666);
            }
            else file2.create(Components.interfaces.nsIFile.NORMAL_FILE_TYPE, 0666);

            const rootPetnameFolder = function () {       //checks if there is already a folder named UniPet, otherwise creates it
            try {
           const existing = bookmarks.getItemIdForGUID(prefs.getCharPref('unipet.bookmarks.root'));
           if (-1 !== existing) { return existing; }
        } catch (e) {}

            const old = bookmarks.getChildFolder(bookmarks.toolbarFolder,"unipet");
            const r = 0 !== old ? old : bookmarks.createFolder(
                        bookmarks.toolbarFolder,
                    properties.getString('bookmarks.title'), 0);
            prefs.setCharPref('unipet.bookmarks.root', bookmarks.getItemGUID(r));
                    return r;
            } ();
            rootFold=rootPetnameFolder;
            if (!prefs.prefHasUserValue('unipet.firstrun')) {prefs.setBoolPref('unipet.firstrun', true);}
            if (!prefs.prefHasUserValue('unipet.useText')) { //unipet.useText is used to switch back and forth between the Petname or the
Petlogo mode
                        prefs.setBoolPref('unipet.useText', true);
                        useText(true);
            }
            else
            {
                        if(prefs.getBoolPref('unipet.useText'))useText(true);
                        else useImage(true);
            }

   const controller = function () {// this functions controls the behavior of the UniPet

   return {
                        navigate: function (url) {//this is called whenever a web page loads or another tab gets selected
                    if (wasURL && (!url || wasURL.prePath !== url.prePath)) {
                                    wasSecure = undefined;
                                    wasPetname = undefined;
                                    viewer.disable();
            }
                        wasURL = url;
        },

                        authenticated: function (isSecure) {//
                                    wasSecure = isSecure;
                                    wasPetname = model.findPetname(wasURL);
                                    const sug=document.getElementById('unipet-suggest');
                                    const nsISSLStatusProvider = Components.interfaces.nsISSLStatusProvider;
                                    const nsISSLStatus = Components.interfaces.nsISSLStatus;
                                    var sui = gBrowser.securityUI;
```

```
                                var sp = sui.QueryInterface(nsISSLStatusProvider);
                                if (sp)var status = sp.SSLStatus;
                                if (status) status = status.QueryInterface(nsISSLStatus);

                                var host=wasURL.prePath;
                                var flag_val=false;
                                var TLD = host.substring(host.lastIndexOf('.')+1, host.length);
                                var hostNoTLD = host.substring(0, host.lastIndexOf('.'));
        var secLevDomain = hostNoTLD.substring(hostNoTLD.lastIndexOf('.')+1, hostNoTLD.length);
                                var firstSecLevDomain = secLevDomain+'.'+TLD;

                                if(!prefs.getBoolPref('unipet.useText'))
                                {
                                        var file = Components.classes["@mozilla.org/file/directory_service;1"].
                                        getService(Components.interfaces.nsIProperties).
                                        get("ProfD", Components.interfaces.nsIFile);
                                        file.append("petlogo.txt");
                                        if( !file.exists())
                                        file.create(Components.interfaces.nsIFile.NORMAL_FILE_TYPE, 0666);

                                var istream = Components.classes["@mozilla.org/network/file-input-stream;1"].
createInstance(Components.interfaces.nsIFileInputStream);
                                istream.init(file, 0x01, 0444, 0);

                                istream.QueryInterface(Components.interfaces.nsILineInputStream);

                                var tmp_str;
                                var line = {}, lines = [], hasmore;

                                do {
                                        hasmore = istream.readLine(line);
                                lines.push(line.value);
                                        tmp_str=lines.join();
                                if((tmp_str.substring(0,tmp_str.indexOf(" "))).match(firstSecLevDomain))
                                {
                                        flag_val=true;
                                        break;
                                }
                                lines = [];
                                } while(hasmore);

                                istream.close();
                                }

if ((wasPetname && prefs.getBoolPref('unipet.useText')) || (flag_val && !prefs.getBoolPref('unipet.useText'))) {
        viewer.showPetname(wasPetname,isSecure,status,firstSecLevDomain);       }
else
{
        if(wasURL.prePath===undefined||wasURL.prePath.match("about:"))viewer.disable();
```

```
            else viewer.null(isSecure,status,firstSecLevDomain);
 }          }              };
} ();

viewer.init();
const SECURE_MASK = Ci.nsIWebProgressListener.STATE_IS_SECURE | Ci.nsIWebProgressListener.STATE_SECURE_HIGH;
const listener = {//listerner keep tracks whenever a new web page is loaded, new tab created, another tab selected or security status changes
onLocationChange: function(progress, request, url) {//this is called whenever a new web page is loaded, new tab created or another tab selected
err && err.logStringMessage('unipet: location change');
try {
            if(prefs.getBoolPref('unipet.useText'))useText(true);
            else useImage(true);
        controller.navigate(url);
        } catch (e) {
            err && err.logStringMessage('unipet: ' + e);
        }
    },

onSecurityChange: function (progress, request, flags) {//this is called whenever a security status changes, HTTP to HTTPS or HTTPS to HTTP
        err && err.logStringMessage('unipet: security change');
        try {
            controller.authenticated(SECURE_MASK === (flags & SECURE_MASK));
        } catch (e) {
            err && err.logStringMessage('unipet: ' + e);
        }
    },

onProgressChange: function() {
        err && err.logStringMessage('unipet: progress change');
    },

onStatusChange: function() {
        err && err.logStringMessage('unipet: status change');
    },

onStateChange: function(webProgress, request, stateFlags, status) {
        err && err.logStringMessage('unipet: state change');
            if(stateFlags & Components.interfaces.nsIWebProgressListener.STATE_STOP)
            {
                    if (stateFlags & Components.interfaces.nsIWebProgressListener.STATE_IS_NETWORK)
                    {
                            var sec,url;
                            url=gBrowser.contentWindow.location+"";
                            var ttitle,tmptitle=document.title;
                            ttitle=tmptitle;
                            title=tmptitle.substring(0,tmptitle.length-37);
```

```
document.title=ttitle;
if(prefs.getBoolPref('unipet.useText') && url.match('https'))
{
	const text=document.getElementById('unipet-text');
	if(text && text.value.match("unknown site"))
	{
		var host=wasURL.prePath;
		var TLD = host.substring(host.lastIndexOf('.')+1, host.length);
		var hostNoTLD = host.substring(0, host.lastIndexOf('.'));
var secLevDomain = hostNoTLD.substring(hostNoTLD.lastIndexOf('.')+1, hostNoTLD.length);
		var firstSecLevDomain = secLevDomain+'.'+TLD;
var file1 = Components.classes["@mozilla.org/file/directory_service;1"].getService(Components.interfaces.nsIProperties).
get("ProfD", Components.interfaces.nsIFile);
file1.append("Petname_suggestion.txt");
if( !file1.exists())
file1.create(Components.interfaces.nsIFile.NORMAL_FILE_TYPE, 0666);

istream = Components.classes["@mozilla.org/network/file-input-stream;1"].
createInstance(Components.interfaces.nsIFileInputStream);
istream.init(file1, 0x01, 0444, 0);
istream.QueryInterface(Components.interfaces.nsILineInputStream);

var flag_val=true,tmp_str;
var line = {}, lines = [], hasmore;
do {
	hasmore = istream.readLine(line);
	lines.push(line.value);
	tmp_str=lines.join();
	if(tmp_str.match(firstSecLevDomain))
	{
		flag_val=false;
		break;
	}
	lines = [];
} while(hasmore);

istream.close();
if(flag_val)
{
	foStream = Components.classes["@mozilla.org/network/file-output-stream;1"].
	createInstance(Components.interfaces.nsIFileOutputStream);
	foStream.init(file1, 0x02 | 0x08 | 0x10, 0666, 0);
	var converter = Components.classes["@mozilla.org/intl/converter-output-
stream;1"].createInstance(Components.interfaces.nsIConverterOutputStream);
	converter.init(foStream, "UTF-8", 0, 0);
	alert("This site carries sensitive information. Apply a Petname for this site");
	var data=firstSecLevDomain+'\r\n';
	converter.writeString(data);
	converter.close();	}		}		}
```

```
else if(!prefs.getBoolPref('unipet.useText') && url.match('https'))
{
        const imageLabel=document.getElementById('imageLabel');
        if(imageLabel && imageLabel.value.match("Add Image"))
        {
                var host=wasURL.prePath;
                var TLD = host.substring(host.lastIndexOf('.')+1, host.length);
                var hostNoTLD = host.substring(0, host.lastIndexOf('.'));
var secLevDomain = hostNoTLD.substring(hostNoTLD.lastIndexOf('.')+1, hostNoTLD.length);
                var firstSecLevDomain = secLevDomain+'.'+TLD;

var file1 = Components.classes["@mozilla.org/file/directory_service;1"].
getService(Components.interfaces.nsIProperties).get("ProfD", Components.interfaces.nsIFile);
file1.append("Petlogo_suggestion.txt");
if( !file1.exists())
file1.create(Components.interfaces.nsIFile.NORMAL_FILE_TYPE, 0666);

istream = Components.classes["@mozilla.org/network/file-input-stream;1"].
createInstance(Components.interfaces.nsIFileInputStream);
istream.init(file1, 0x01, 0444, 0);
istream.QueryInterface(Components.interfaces.nsILineInputStream);

var flag_val=true,tmp_str;
var line = {}, lines = [], hasmore;
do {
        hasmore = istream.readLine(line);
        lines.push(line.value);
        tmp_str=lines.join();
        if(tmp_str.match(firstSecLevDomain))
        {
                flag_val=false;
                break;
        }
lines = [];
} while(hasmore);
istream.close();
if(flag_val)
{
        foStream = Components.classes["@mozilla.org/network/file-output-stream;1"].
        createInstance(Components.interfaces.nsIFileOutputStream);
        foStream.init(file1, 0x02 | 0x08 | 0x10, 0666, 0);
        var converter = Components.classes["@mozilla.org/intl/converter-output-
stream;1"].createInstance(Components.interfaces.nsIConverterOutputStream);
        converter.init(foStream, "UTF-8", 0, 0);

        alert("This site carries sensitive information. Apply a Petlogo for this site");
        var data=firstSecLevDomain+'\r\n';
        converter.writeString(data);
        converter.close();      }       }       }       }       }       },
```

```
      QueryInterface: function(iid) {
        if (iid.equals(Ci.nsIWebProgressListener) ||
            iid.equals(Ci.nsISupportsWeakReference) ||
            iid.equals(Ci.nsISupports)) { return this; }
        throw Components.results.NS_ERROR_NO_INTERFACE;
      }
    };
    listener.onLocationChange(null, null, gBrowser.contentWindow.location);
    listener.onSecurityChange(null, null, gBrowser.securityUI.state);
    gBrowser.addProgressListener(listener);  //this registers the listener with the broswer
    err && err.logStringMessage('unipet: loaded');
}, false);

function bCommand()             //this is called whenever users click the Apply  button...
            {
                        if(prefs.getBoolPref('unipet.useText'))
                        {
                                    var prompts = Components.classes["@mozilla.org/embedcomp/prompt-service;1"]
                                     .getService(Components.interfaces.nsIPromptService);
                                    const text = document.getElementById('unipet-text');
                                    const sug=document.getElementById('unipet-suggest');
                                    if(text.value.length==0)alert("No Site to apply Petname!");
                        else if (text.value.match("unknown site"))alert("Write down your Petname and then click Apply button");
                        else if (text.value.match(sug.value))alert("The Petname is similar to the suggestion. Please change it a bit");
                                    else{
                                                newPetname = model.findPetname(wasURL);
                                                const nsISSLStatusProvider = Components.interfaces.nsISSLStatusProvider;
                                                const nsISSLStatus = Components.interfaces.nsISSLStatus;
                                                var sui = gBrowser.securityUI;
                                                var sp = sui.QueryInterface(nsISSLStatusProvider);
                                                if (sp)var status = sp.SSLStatus;
                                                if (status) status = status.QueryInterface(nsISSLStatus);
                                                var host=wasURL.prePath;
                                                var TLD = host.substring(host.lastIndexOf('.')+1, host.length);
                                                var hostNoTLD = host.substring(0, host.lastIndexOf('.'));
            var secLevDomain = hostNoTLD.substring(hostNoTLD.lastIndexOf('.')+1, hostNoTLD.length);
                                                var firstSecLevDomain = secLevDomain+'.'+TLD;

                                                const similar= model.findSimilarPetnames(text.value, newPetname);
                        {
                                                            if (0 === similar.length) {
                                                                        if(newPetname){
                                                                        model.changePetnameTitle(newPetname, text.value);
                                                                        var tmpPetname=model.findPetname(wasURL);
            viewer.showPetname(tmpPetname,wasSecure,status,firstSecLevDomain);
                        }
                        else {
                                    var tmpPetname = model.createPetname(text.value, wasURL);
                                    viewer.showPetname(tmpPetname,wasSecure,status,firstSecLevDomain);     }
```

```
		} else {
			const pName = similar[0];
		if(text.value.match(pName.title)&&(pName.title.length==text.value.length))viewer.alertSame(text.value, similar);
		else viewer.alertCollisions(text.value, similar);
	}	}		}		}
		else if(!prefs.getBoolPref('unipet.useText'))
		{
			const imageLabel = document.getElementById('imageLabel');
			const petLogo = document.getElementById('petLogo');
		if(imageLabel && imageLabel.value.match("Image Disabled"))alert("No site to apply Petlogo.")
else if(imageLabel && imageLabel.value.match("Add Image"))alert("Select an image by clicking the Browse button and then click
apply.")
		else
		{
			var file = Components.classes["@mozilla.org/file/directory_service;1"].
			getService(Components.interfaces.nsIProperties).get("ProfD", Components.interfaces.nsIFile);
			file.append("petlogo.txt");
			if( !file.exists())   // if it doesn't exist, create
			file.create(Components.interfaces.nsIFile.NORMAL_FILE_TYPE, 0666);

			var data;
			var host=wasURL.prePath;
			var TLD = host.substring(host.lastIndexOf('.')+1, host.length);
			var hostNoTLD = host.substring(0, host.lastIndexOf('.'));
		var secLevDomain = hostNoTLD.substring(hostNoTLD.lastIndexOf('.')+1, hostNoTLD.length);
			var firstSecLevDomain = secLevDomain+'.'+TLD;

			var istream = Components.classes["@mozilla.org/network/file-input-stream;1"].
		createInstance(Components.interfaces.nsIFileInputStream);
			istream.init(file, 0x01, 0444, 0);
			istream.QueryInterface(Components.interfaces.nsILineInputStream);
			var flag_val=false;
			var line = {}, lines = [], hasmore;
			do {
				hasmore = istream.readLine(line);
				lines.push(line.value);
				tmp_str=lines.join();
				if((tmp_str.substring(0,tmp_str.indexOf(" "))).match(firstSecLevDomain))
				{
					flag_val=true;
					break;
				}
				lines = [];
			} while(hasmore);
			istream.close();
			if(!flag_val)
			{
				var foStream = Components.classes["@mozilla.org/network/file-output-stream;1"].
				createInstance(Components.interfaces.nsIFileOutputStream);
```

```
            foStream.init(file, 0x02 | 0x08 | 0x10, 0666, 0);
            data=firstSecLevDomain+" "+petLogo.getAttribute('src')+'\r\n';
            var converter = Components.classes["@mozilla.org/intl/converter-output-
stream;1"].createInstance(Components.interfaces.nsIConverterOutputStream);
            converter.init(foStream, "UTF-8", 0, 0);
            converter.writeString(data);
            converter.close();
            alert("Petlogo applied for this site.");
            }
            else if(flag_val)
            {
            var file2 = Components.classes["@mozilla.org/file/directory_service;1"].
            getService(Components.interfaces.nsIProperties).get("ProfD", Components.interfaces.nsIFile);
            file2.append("petlogotmp.txt");
            if( !file2.exists())
            file2.create(Components.interfaces.nsIFile.NORMAL_FILE_TYPE, 0666);
            var file1 = Components.classes["@mozilla.org/file/directory_service;1"].
            getService(Components.interfaces.nsIProperties).get("ProfD", Components.interfaces.nsIFile);
            file1.append("petlogo.txt");
            if( !file1.exists())
            file1.create(Components.interfaces.nsIFile.NORMAL_FILE_TYPE, 0666);

            istream = Components.classes["@mozilla.org/network/file-input-stream;1"].
            createInstance(Components.interfaces.nsIFileInputStream);
            istream.init(file1, 0x01, 0444, 0);
            istream.QueryInterface(Components.interfaces.nsILineInputStream);
            var flag_val=false;
            var foStream = Components.classes["@mozilla.org/network/file-output-stream;1"].
            createInstance(Components.interfaces.nsIFileOutputStream);
            foStream.init(file2, 0x02 | 0x08 | 0x10, 0666, 0);
            var converter = Components.classes["@mozilla.org/intl/converter-output-
stream;1"].createInstance(Components.interfaces.nsIConverterOutputStream);
            converter.init(foStream, "UTF-8", 0, 0);
            data=firstSecLevDomain+" "+petLogo.getAttribute('src')+'\r\n';
            var line = {}, lines = [], hasmore;
            do {
                        hasmore = istream.readLine(line);
                        lines.push(line.value);
                        tmp_str=lines.join();
                        if(!(tmp_str.substring(0,tmp_str.indexOf(" "))).match(firstSecLevDomain))
                        converter.writeString(tmp_str+'\r\n');
                        lines = [];
            } while(hasmore);
            converter.writeString(data);
            istream.close();
            converter.close();
            file1.remove(true);
            var file1 = Components.classes["@mozilla.org/file/directory_service;1"].
            getService(Components.interfaces.nsIProperties).
```

```
            get("ProfD", Components.interfaces.nsIFile);
            file1.append("petlogo.txt");
            if( !file1.exists())
            file1.create(Components.interfaces.nsIFile.NORMAL_FILE_TYPE, 0666);

            istream = Components.classes["@mozilla.org/network/file-input-stream;1"].
            createInstance(Components.interfaces.nsIFileInputStream);
            istream.init(file2, 0x01, 0444, 0);
            istream.QueryInterface(Components.interfaces.nsILineInputStream);

            foStream = Components.classes["@mozilla.org/network/file-output-stream;1"].
        createInstance(Components.interfaces.nsIFileOutputStream);
            foStream.init(file1, 0x02 | 0x08 | 0x10, 0666, 0);
            var converter = Components.classes["@mozilla.org/intl/converter-output-
stream;1"].createInstance(Components.interfaces.nsIConverterOutputStream);
            converter.init(foStream, "UTF-8", 0, 0);

            line = {}, lines = [], hasmore;
            do {
                        hasmore = istream.readLine(line);
                        lines.push(line.value);
                        tmp_str=lines.join();
                        converter.writeString(tmp_str+'\r\n');
                        lines = [];
            } while(hasmore);

            istream.close();
            converter.close();
            file2.remove(true);
            alert("Petlogo updated a with new logo for this site.");
            }           }           }           }

function tBFocus() //this is called whenever users select the Petname text box
{
            const text = document.getElementById('unipet-text');
            if(text.value.length==0)alert("No Site to apply Petname!");
            else viewer.edit();
}

function useText(tmpFlag)// updates the GUI when the UniPet is in the Petname mode
{
            prefs.setBoolPref('unipet.useText', true);
            const text = document.getElementById('text');
            const image = document.getElementById('image');
            const label = document.getElementById('name');
            var parentDiv = label.parentNode;
            var element = document.getElementById('pet_hbox');
            text.setAttribute('label','Using Text');
            image.setAttribute('label','Use Image');
```

```
		label.setAttribute('value','Petname:');
		if(document.getElementById('imageLabel'))element.removeChild(document.getElementById('imageLabel'));
if(document.getElementById('petLogo'))element.removeChild(document.getElementById('petLogo'));
if(document.getElementById('browse_button'))element.removeChild(document.getElementById('browse_button'));

		if(!document.getElementById('unipet-text'))
		{
			var sp1 = document.createElement("textbox");
			sp1.setAttribute('id','unipet-text');
			sp1.setAttribute('size','18');
			sp1.setAttribute('class','unipet-disabled');
			sp1.setAttribute('readonly','true');
			sp1.setAttribute('onfocus','tBFocus()');
			parentDiv.insertBefore(sp1,document.getElementById('apply_button'));
		}
		if(tmpFlag)
		{
			return;
		}

		var wasPetname = model.findPetname(wasURL);
		const nsISSLStatusProvider = Components.interfaces.nsISSLStatusProvider;
		const nsISSLStatus = Components.interfaces.nsISSLStatus;
		var sui = gBrowser.securityUI;
		var sp = sui.QueryInterface(nsISSLStatusProvider);
		if (sp)var status = sp.SSLStatus;
		if (status) status = status.QueryInterface(nsISSLStatus);

		var host=wasURL.prePath;
		var TLD = host.substring(host.lastIndexOf('.')+1, host.length);
		var hostNoTLD = host.substring(0, host.lastIndexOf('.'));
		var secLevDomain = hostNoTLD.substring(hostNoTLD.lastIndexOf('.')+1, hostNoTLD.length);
		var firstSecLevDomain = secLevDomain+'.'+TLD;

		if(wasURL.prePath===undefined||wasURL.prePath.match("about:"))viewer.disable();
		else if(wasPetname)
		{
			viewer.showPetname(wasPetname,wasSecure,status,firstSecLevDomain);
		}
		else
		{
			const textBox=document.getElementById('unipet-text');
			var url=gBrowser.contentWindow.location+"";
			viewer.null(wasSecure,status,firstSecLevDomain);
		}	}

function useImage(tmpFlag1)//updates the GUI when the UniPet is in the Petlogo mode
{
		prefs.setBoolPref('unipet.useText', false);
```

```
const text = document.getElementById('text');
const image = document.getElementById('image');
const label = document.getElementById('name');
const textBox = document.getElementById('unipet-text');
var element = document.getElementById('pet_hbox');
var parentDiv = label.parentNode;

text.setAttribute('label','Use Text');
image.setAttribute('label','Using Image');
label.setAttribute('value','Petlogo:');
if(textBox)element.removeChild(textBox);
if(document.getElementById('imageLabel'))element.removeChild(document.getElementById('imageLabel'));
if(document.getElementById('petLogo'))element.removeChild(document.getElementById('petLogo'));
if(document.getElementById('browse_button'))element.removeChild(document.getElementById('browse_button'));

if(!document.getElementById('imageLabel'))
{
        var sp1 = document.createElement("label");
        sp1.setAttribute('id','imageLabel');
        sp1.setAttribute('style','border: 1px solid;');
        sp1.setAttribute('value','Image Disabled');
        sp1.setAttribute('tooltiptext','Click the browse button to add an image');
        parentDiv.insertBefore(sp1,document.getElementById('apply_button'));
}


if(!document.getElementById('browse_button'))
{
        var sp1 = document.createElement("button");
        sp1.setAttribute('id','browse_button');
        sp1.setAttribute('style','border: 1px solid;');
        sp1.setAttribute('label','  Browse');
        sp1.setAttribute('tooltiptext','Click here to add an image');
        sp1.setAttribute('image','chrome://unipet/skin/images--plus.png');
        sp1.setAttribute('onclick','browseButton()');
        parentDiv.insertBefore(sp1,document.getElementById('apply_button'));
}

if(tmpFlag1)
{
        return;
}
var wasPetname = model.findPetname(wasURL);
const nsISSLStatusProvider = Components.interfaces.nsISSLStatusProvider;
const nsISSLStatus = Components.interfaces.nsISSLStatus;
var sui = gBrowser.securityUI;
var sp = sui.QueryInterface(nsISSLStatusProvider);
if (sp)var status = sp.SSLStatus;
if (status) status = status.QueryInterface(nsISSLStatus);
```

```
		var host=wasURL.prePath;
		var TLD = host.substring(host.lastIndexOf('.')+1, host.length);
		var hostNoTLD = host.substring(0, host.lastIndexOf('.'));
		var secLevDomain = hostNoTLD.substring(hostNoTLD.lastIndexOf('.')+1, hostNoTLD.length);
		var firstSecLevDomain = secLevDomain+'.'+TLD;
		var flag_val=false;
		var file = Components.classes["@mozilla.org/file/directory_service;1"].
		getService(Components.interfaces.nsIProperties).
		get("ProfD", Components.interfaces.nsIFile);
		file.append("petlogo.txt");
		if( !file.exists())
		file.create(Components.interfaces.nsIFile.NORMAL_FILE_TYPE, 0666);
		var istream = Components.classes["@mozilla.org/network/file-input-stream;1"].
   createInstance(Components.interfaces.nsIFileInputStream);
		istream.init(file, 0x01, 0444, 0);
		istream.QueryInterface(Components.interfaces.nsILineInputStream);
		var tmp_str;

		var line = {}, lines = [], hasmore;
		do {
			hasmore = istream.readLine(line);
			lines.push(line.value);
			tmp_str=lines.join();
			if((tmp_str.substring(0,tmp_str.indexOf(" "))).match(firstSecLevDomain))
				{
					flag_val=true;
					break;
				}
			lines = [];
		} while(hasmore);

		istream.close();

		if(wasURL.prePath===undefined||wasURL.prePath.match("about:"))viewer.disable();
		else if(flag_val)
		{
			viewer.showPetname(wasPetname,wasSecure,status,firstSecLevDomain);
		}
		else
		{
			const imageLabel=document.getElementById('imageLabel');
			var url=gBrowser.contentWindow.location+"";
			viewer.null(wasSecure,status,firstSecLevDomain);
		}		}

function delPetname()//this is called when users click the Delete All Petnames item from the Options menu
{
		var prompts = Components.classes["@mozilla.org/embedcomp/prompt-service;1"]
		    .getService(Components.interfaces.nsIPromptService);
```

```
	const bookmarks = Cc['@mozilla.org/browser/nav-bookmarks-service;1'].getService(Ci.nsINavBookmarksService);
	var result = prompts.confirm(null, "You are trying to delete all Petnames", "Do you really want to delete all the Petname?");
	if(result)
	{
		bookmarks.removeFolderChildren(rootFold);
		prompts.alert(null, "Petnames Deleted", "All Petnames have been deleted.");
		const nsISSLStatusProvider = Components.interfaces.nsISSLStatusProvider;
		const nsISSLStatus = Components.interfaces.nsISSLStatus;
		var sui = gBrowser.securityUI;
		var sp = sui.QueryInterface(nsISSLStatusProvider);
		if (sp)var status = sp.SSLStatus;
		if (status) status = status.QueryInterface(nsISSLStatus);

		var host=wasURL.prePath;
		var TLD = host.substring(host.lastIndexOf('.')+1, host.length);
		var hostNoTLD = host.substring(0, host.lastIndexOf('.'));
		var secLevDomain = hostNoTLD.substring(hostNoTLD.lastIndexOf('.')+1, hostNoTLD.length);
		var firstSecLevDomain = secLevDomain+'.'+TLD;
		const text = document.getElementById('text');
		if(prefs.getBoolPref('unipet.useText'))
		{
			if(text && text.value.match(""))viewer.disable();
			else viewer.null(wasSecure,status,firstSecLevDomain);
		}	}	}

function delPetlogo()//this is called when users click the Delete All Petlogos item from the Options menu
{
	var prompts = Components.classes["@mozilla.org/embedcomp/prompt-service;1"]
	    .getService(Components.interfaces.nsIPromptService);
	var result = prompts.confirm(null, "You are trying to delete all Petlogos", "Do you really want to delete all the Petlogo?");
	if(result)
	{
		prompts.alert(null, "Petlogos Deleted", "All Petlogos have been deleted.");
		const nsISSLStatusProvider = Components.interfaces.nsISSLStatusProvider;
		const nsISSLStatus = Components.interfaces.nsISSLStatus;
		var sui = gBrowser.securityUI;
		var sp = sui.QueryInterface(nsISSLStatusProvider);
		if (sp)var status = sp.SSLStatus;
		if (status) status = status.QueryInterface(nsISSLStatus);
		var host=wasURL.prePath;
		var TLD = host.substring(host.lastIndexOf('.')+1, host.length);
		var hostNoTLD = host.substring(0, host.lastIndexOf('.'));
	var secLevDomain = hostNoTLD.substring(hostNoTLD.lastIndexOf('.')+1, hostNoTLD.length);
		var firstSecLevDomain = secLevDomain+'.'+TLD;
		var file = Components.classes["@mozilla.org/file/directory_service;1"].
		getService(Components.interfaces.nsIProperties).
		get("ProfD", Components.interfaces.nsIFile);
		file.append("petlogo.txt");
```

```
                        if( !file.exists())
                        file.create(Components.interfaces.nsIFile.NORMAL_FILE_TYPE, 0666);
                        file.remove(true);
                        const imageLabel = document.getElementById('imageLabel');
                        if(!prefs.getBoolPref('unipet.useText'))
                        {
                                if(imageLabel && imageLabel.value.match("Image Disabled"))viewer.disable();
        else viewer.null(wasSecure,status,firstSecLevDomain);
                        }       }       }
function help()//this is called when users click the Help item from the options menu
{
        gBrowser.selectedTab = gBrowser.addTab("chrome://unipet/content/Help on Toolbar.html");
}
function about()//this is called when users click the About item from the options menu
{
        window.openDialog("chrome://unipet/content/aboutDialog.xul", "About UniPet", "centerscreen");
}
function browseButton()//this is called when users click the Browse button
{
                        const text = document.getElementById('imageLabel');
                        var element = document.getElementById('pet_hbox');
                        const label = document.getElementById('name');
                        var parentDiv = label.parentNode;
                        if(text && text.value.match("Image Disabled"))alert("No Site to apply Petlogo!");
                        else
                        {
                                const nsIFilePicker = Components.interfaces.nsIFilePicker;
                                var fp = Components.classes["@mozilla.org/filepicker;1"].createInstance(nsIFilePicker);
                                fp.init(window, "Select an image file", nsIFilePicker.modeOpen);
                                fp.appendFilter("Image Files","*.bmp; *.jpg; *.png");
                                var rv = fp.show();
                                if (rv == nsIFilePicker.returnOK) {
                                var file = fp.file;
                                var path = fp.file.path;
                                var ios = Components.classes["@mozilla.org/network/io-
                                service;1"].getService(Components.interfaces.nsIIOService);
                                var URL = ios.newFileURI(file);
                                var file = Components.classes["@mozilla.org/file/directory_service;1"].
                                getService(Components.interfaces.nsIProperties).
                                get("ProfD", Components.interfaces.nsIFile);
                                file.append("petlogo.txt");
                                if( !file.exists())   // if it doesn't exist, create
                                file.create(Components.interfaces.nsIFile.NORMAL_FILE_TYPE, 0666);
                                var istream = Components.classes["@mozilla.org/network/file-input-stream;1"].
                                createInstance(Components.interfaces.nsIFileInputStream);
                                istream.init(file, 0x01, 0444, 0);
                                istream.QueryInterface(Components.interfaces.nsILineInputStream);
                                var tmp_str,flag_val=false;
                                var line = {}, lines = [], hasmore;
```

```
        do {
                hasmore = istream.readLine(line);
                lines.push(line.value);
                tmp_str=lines.join();
                if((tmp_str.substring(tmp_str.indexOf(" "),tmp_str.length)).match(URL.spec))
                {
                        flag_val=true;
                        break;
                }
                lines = [];
                } while(hasmore);

                istream.close();
                if(flag_val)
                {
                        alert("You're trying to assign a same Petlogo that was assigned for other site. Please choose another
Petlogo");
                        browseButton();
                }
                else
                {
                        if(text)element.removeChild(text);
                        const pLogo = document.getElementById('petLogo');
                        if(pLogo)
                        {
                                pLogo.setAttribute('src',URL.spec);
                        }
                        else
                        {
                                var sp1 = document.createElement("image");
                                sp1.setAttribute('id','petLogo');
                                sp1.setAttribute('height','26');
                                sp1.setAttribute('width','26');
                                sp1.setAttribute('src',URL.spec);
                                parentDiv.insertBefore(sp1,document.getElementById('browse_button'));
                        }
                        }
                        }
                }
}
```