Petname Systems

Marc Stiegler
Mobile and Media Systems Laboratory
HP Laboratories Palo Alto
HPL-2005-148
August 15, 2005*

computer security, phishing

It has been repeatedly observed [Zooko, Shirky, PNML, Close] that global namespaces suffer from a variety of difficulties. While different analyses have focused on different problems, the conclusion emerges that global names are often overloaded with too many purposes, purposes that come into conflict as the system reaches global scale. One representation of the conflict is, global namespaces attempt to achieve the following trio of properties all at the same time: each name should be global, memorable, and securely collision free. Domain names are an example of a system that attempts to achieve this trinity: domain names are global, and memorable, but as the rapid rise of phishing demonstrates, they are not securely collision free.

Though it may not be possible for any single namespace to have all three properties, *petname systems* do embody all three properties. Informal experiments with petname-like systems suggest that petnames can be both intuitive and effective. Experimental implementations already exist for simple extensions to existing browsers that could alleviate problems with phishing. As phishers gain sophistication, experimenting with petname systems as part of the solution seems compelling.

* Internal Accession Date Only
Approved for External Publication

Petname Systems

by Marc Stiegler

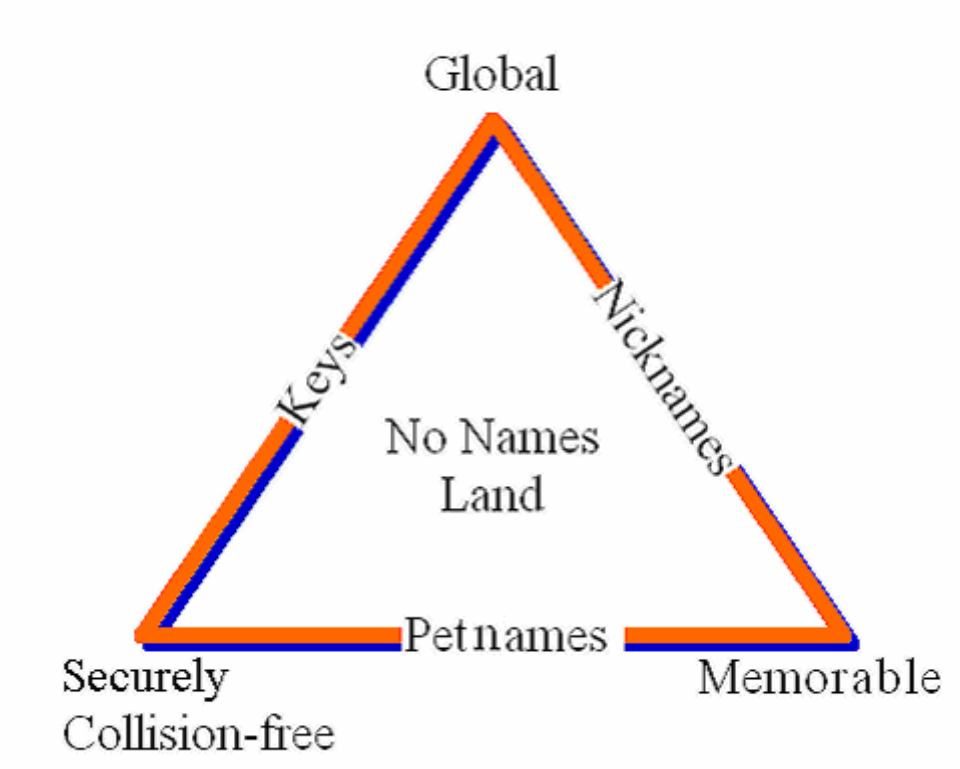
Abstract

It has been repeatedly observed [Zooko, Shirky, PNML, Close] that global namespaces suffer from a variety of difficulties. While different analyses have focused on different problems, the conclusion emerges that global names are often overloaded with too many purposes, purposes that come into conflict as the system reaches global scale. One representation of the conflict is, global namespaces attempt to achieve the following trio of properties all at the same time: each name should be global, memorable, and securely collision free. Domain names are an example of a system that attempts to achieve this trinity: domain names are global, and memorable, but as the rapid rise of phishing demonstrates, they are not securely collision free.

Though it may not be possible for any single namespace to have all three properties, *petname systems* do embody all three properties. Informal experiments with petname-like systems suggest that petnames can be both intuitive and effective. Experimental implementations already exist for simple extensions to existing browsers that could alleviate problems with phishing. As phishers gain sophistication, experimenting with petname systems as part of the solution seems compelling.

Problem Description

The figure shows the properties we'd like a name system to have overlaid with a petname system:



The properties at the points of the triangle are:

- **Memorable:** a human being should be able to remember the name. Memorable names pass the "moving bus test": if you see the name on the side of a bus as it drives past you, you should be able to remember the name long enough to use it when you get home.
- **Global:** the name should mean the same thing no matter where it is used or who is using it. A key goal of marketing and advertising is to capture memorable names in such a fashion that the memorable name is globally locked to a particular entity.
- **Securely Collision Free:** This is means that the name cannot be *forged* or *mimicked*. A name can be forged if one can manufacture an exact duplicate of the name such that neither man nor machine can tell the difference. A name can be mimicked if one can make a name similar enough to fool the human being. In general, phishing depends on mimicry, not forgery.

Attempts to endow a single name with all three properties may lead, at sufficiently large scale, to any and all of the following problems:

- **Dependency** upon a trusted third party that may make grievous errors or be subverted.
- **Political and legal conflict** over particular names, or even over names that are merely similar. In the best case, resolution will be considered unfair by some parties in the conflict. In other cases resolution may be unachievable, as the dispute crosses jurisdictional boundaries.
- **Simple confusion** periodically leading to both humorous and lethal miscommunications: a cancer biopsy report emailed to john.doe@domain.net instead of to john.doe@domain.com could cause loss of life.
- **Malicious confusion**, as demonstrated by phishing attacks that intentionally use the similarities of names to lure victims into surrender of valuable private information.

Basic Petname Layout

A petname system uses three interrelated types of names that, together, achieve all three of the desirable properties. The types of names are: *keys* that are global and securely collision-free (but not necessarily memorable); *nicknames* that are global and memorable (but often collide), and *petnames* that are securely collision free and memorable (but private, not global):

- **Keys** lie at the heart of the security properties of the petname system. A key is a globally unique, unforgeable designator of some specific entity. The security of the system can be no stronger than the unforgeability of the keys. Self-authenticating public/private key pairs make good keys since they have strong unforgeability properties. But there are other ways of achieving unforgeability. A trusted path can also work well as the key: the full pathname to a file on a specific computer is also unforgeable. Nicknames and petnames exist to make it easy for human beings to manipulate keys. It makes no difference in a petname system whether a key can be mimicked: keys are handled only by the computer, the human being handles the keys only indirectly via petnames. For a particular person, for a particular application, there is a one-to-one mapping between a key and a petname.

- **Nicknames** can be used to assist in discovery of keys, and for help in selecting a petname. Nicknames are chosen by the owners of keys in hopes of creating a distinctive, if not unique, mapping from the memorable nickname to the key. Such nicknames often are promulgated throughout the world in the hopes of making the nickname stick in the mind as a reference to the key. Since there are strong incentives to "take ownership" of a nickname, even though true ownership is not possible, nicknames may be the most misunderstood part of a petname system.

  In the simple case, a nickname has a one-to-many mapping to keys. The name John Smith is obviously a nickname: there are many John Smiths. Other nicknames produce the illusion of being globally unique: the name Argus Billaby

appears to be collision free at the time of this writing. But there is no security property in this accident of global uniqueness. The uniqueness of the name Argus Billaby would change quite quickly if, through human whimsy, the name suddenly became desirable. Sometimes the desirability of a nickname is not whimsical, but venal. It is already desirable for some applications to call themselves Quicken, and draw windows that request the Quicken password.

- **Petnames** are our private bidirectional references to keys. There are many dogs named Rover, but for the owner of a dog with the petname Rover, there is one specific Rover to whom the name refers. In the computer setting, for a specific person with a specific application, petnames are unique, each petname refers to exactly one key, and each key is represented by exactly one petname. In all places in the application where the application wants to designate the key, the petname is displayed -- which is to say, *a true petname is a bidirectional one-to-one mapping to a key*. All references to the key by the user interface are represented by the petname. A key cannot have two petnames; if a single key had two petnames, under what circumstances would the user interface display Petname-1 as the representation of the key, and under what circumstances would it display Petname-2?

The security of a petname system depends on the keys to prevent forgery, and on the petnames to prevent mimicry.

More Detail, and Interactions

A good example of a nickname management system is Google. Type in a name, and Google will return a list that includes all the entities Google knows, to which the name refers. Google makes a mapping between these nicknames and their keys (if we think of the URL of a page as a key). Often enough to be interesting, the first item in the list will be the desired item. But it fails often enough, and endless pages of other choices appear often enough, to never leave us in doubt that these identifiers are not securely collision free mappings to single keys. As is already true in the current world, in a world filled with petname systems, a key goal of marketing would be to get your URL listed at the top of the Google rankings for your nickname.

Nicknames are conveniences that may serve as good starting points for petnames. If Argus Billaby sends someone his key and his nickname, often his nickname will work quite well as a petname. But the nickname-as-proposal must not be confused with the petname-as-decided. Never in a true petname system is the nickname presented or employed as if it were a petname. After all, the recipient might already have assigned that petname to another Argus Billaby.

Alleged names are similar to nicknames. An alleged name is the name for an entity proposed by a third party, typically in an introduction. An alleged name can also be useful as a starting place for picking a petname. Alleged names, like nicknames, are usually memorable, often global, and never securely collision free. Alleged names are

often based on nicknames, though this is unreliable enough, if one really cares about the nickname then one really needs to ask the entity designated by the key, not the third party performing the introduction.

In action, keys and alleged names tend to be transferred together. We refer henceforth to such key/alleged-name pairs as *referrals.*

It is crucial not to confuse private petnames with global nicknames that temporarily happen to have a unique mapping to a key. Experience to date suggests that the term "petname" is attractive, leading people to desire to use it. People can thence easily fall into the trap of referring to momentarily collision-free nicknames as "petnames". This error then leads them inevitably to draw fatally confused conclusions about the possibility of petnames with global meaning. The security properties of a petname derive from its privacy. Public nicknames are trivially vulnerable to both forgery and mimicry; they have few interesting security properties. (They do have one interesting security property: If you do discovery on the nickname “John” and an introduction agent refers you to Jane, you can authenticate the alleged introduction by asking Jane “Do you go by the nickname ‘Joe’?”. This is irrelevant against phishing attacks, but is useful in other contexts)

Petnames are guessable. Most people will accept Paypal's nickname as the petname. This can only impact the security of the system if the user interface fails to unambiguously distinguish the user-specified petnames from suggested nicknames.

The term "petname" suggests that this name is embodied as text. This is not necessary. Petnames can be graphical as well. Indeed, some of the petname systems listed later use both *pet texts* and *pet graphics.*

Petnames must be repeatably editable by the human being so that the set of petnames can evolve as the user's set of associations grow. You might use the petname "John Smith" for the one and only John Smith that you know. But then if you meet another John Smith you will have to distinguish, possibly by editing the first one: Instead of the single entry "John Smith" you may now have "John Smith Security Guru" and "John Smith Dentist".

Petnames convey power: since the petname is the user’s representation of the key, it is through the petname that the human being uses the key, communicates with the key owner, and conveys authority to the key owner based on the user's *purposeful trust* relationship with that owner (*purposeful trust* is the type of trust needed to engage in action: I trust (i.e., I am willing to be vulnerable to) PayPal to hold N number of dollars on my behalf, and to engage in transfers of that money based on orders I give).

Another way of thinking about the relationship between a key and a petname is as follows. The key is used to authenticate the entity that owns the key. The petname is used as a handle upon which to hang the trust/reliance/vulnerability data used by the human being to make authorization decisions for that entity. If the entity represented by the petname My Phone Company asks for a credit card, if the justification sounds reasonable, one may choose to release the data. If the entity represented by the petname Deadbeat

Brother asks for a credit card, the recipient of the request will probably not release the card number no matter what the justification (even though the recipient may trust the Deadbeat Brother to teach soccer to the recipient's daughter without supervision -- the trust relationship with a deadbeat brother is not exactly positive nor negative, it is more complex).

Having leaned so heavily upon the term "entity" in the above paragraph, a brief attempt to define it seems in order. Entities are those objects with which human beings form distinct trust relationships. Each human being has his own model of what constitutes an entity, and this model evolves over time. Let us look at three examples.

- A Web example: if the owners of the car.com domain take over the car.net domain and duplicate the pages there, car.net and car.com are the same entity.
- A human example: If John creates the pseudonyms Carol and Harley, and Bob meets both Carol and Harley without realizing they both represent John, then Bob will treat each separate persona as a different entity. Upon discovering that the multiple persona represent the same individual, Bob may decide to treat all the persona as a single entity. Or he may not, if the persona have sufficiently different characteristics (if Bob/Carol responds to email quickly but Bob/Harley responds erratically).
- An organizational example: We may have a strong trust relationship with the repair department of Smith's Used Cars without trusting the sales department at all, even though both are part of the same "entity", and may even share the same public key from a Certificate Authority.

An idealized petname would have a one to one mapping to an entity, and that mapping would survive despite changes in the key, despite multiple keys for the same entity, and even despite single keys that represent multiple entities. For the purposes of this paper, we assume that there is a one to one mapping between the entity and the key, and therefore transitively a one to one mapping between the petname and the entity.

Petnames In Action

Informal experimentation suggests that a petname system is much easier to use than to explain (see examples below). We will create a single example for this introduction, and give some hint as to the wide diversity of variations in the Examples. Suppose Argus Billaby sends Carol a referral to John Smith's OpenPGP public key in email. Argus says, "here is John Smith's public key." Argus has sent both a key and an alleged name (John Smith). Implicit in the transmission of the alleged name is the proposal that one might want to consider "John Smith" as the petname. Whether one actually chooses John Smith as a petname depends entirely on the recipient's context. If one knows this particular John Smith in other contexts as "John", one may choose "John" as the name referring to this key in the list of public keys. If one thinks this might be the same John Smith as in other contexts, but is not willing to be vulnerable to Argus as the sole source of such a powerful mapping, one might use the petname "Argus Billaby's John Smith".

If a newly received public key already exists in one's list of public keys, the software shouldn't give you the choice of adding it: the software should point out that this public key is already listed and point out the current petname. If one receives a message signed with the private key for the "John" petname's public key, the software should display the petname John. If one sends a message to John, the software should pick the encryption key based on the petname.

The above example has the security properties of a petname system, but OpenPGP systems often do not demonstrate the usability properties a petname system needs. Instant messaging systems with buddy lists demonstrate the usability properties, but discard all the security properties. See the examples section for more details on buddy lists as petname systems.

Key Issues with Petname Systems

Two elements of full-fledged petname systems seem to be principle sources of controversy. One question is, "how do I get the keys transferred around the system?" The other is, "how easily can Darth Vader mimic a petname?"

Transferring Keys and Purposeful Trust

Transferring keys around the universe is easy; one could, for example, plaster the keys on all the web sites in the world that will let one do so. The hard part is transferring a key with an association to purposeful trust. It is useless to both PayPal and the phisher who wants your Paypal account if you just know Paypal's key. You have to be willing to make yourself vulnerable to the PayPal key owner to hold your credit card, trusting that entity to engage in only transfers that you specify, before either PayPal or the phisher can pursue any benefits.

The question, "how do I transfer a purposeful trust association?" has no single answer. Rather, there are many answers, each of which works in narrow circumstances. The question is made even more difficult to answer because the mechanisms by which humans determine an appropriate purposeful trust relationship is subtle, complex, powerful, and completely subconscious: the question of how one transfers the association can easily slide into a much more difficult discussion of how to create purposeful trust in the first place. Here we outline some general ideas for transferring key/purposeful-trust mappings, then in the Examples point out some practical approaches in specific narrow contexts.

Transfers of purposeful trust often start with direct physical contact. One gets a combination of a nickname and a key in a file from your best friend, who says, "this Google thing is a great search engine", or "this Consumer Reports site will not lead you astray". You stick these referrals in your browser, assign petnames, and make yourself vulnerable to them for the purposes stated because your friend said so. Then when the side of the bus says PayPal, you might go and see what Google thinks Paypal means. Since a relationship with PayPal is a serious vulnerability decision, serious enough so that

people in general will not jump at the first site just because Google said so, we'll ask a few of our friends to email referrals to the entities they use for online money. If the referrals they send corroborate the Google pick (which is easy to tell, because trying to add each new key/petname mapping will produce the alert that the key matches one you've already got), one's willingness to be vulnerable to the key petnamed PayPal increases.

The process described above is pretty similar to how people started using PayPal even without petname systems: people joined when enough of their friends and organizations that they trusted for recommendations about financial matters concurred. The only difference in the petname version of the story is that the friends explicitly give referrals rather than easily mimicked domain names, and we explicitly set a petname (perhaps by just clicking an Accept key when the alleged name was proposed as the petname).

While a full-fledged, purebred petname system could in principle supplant the entire DNS system, we have DNS now. It can be used for bootstrapping. The ability to type google.com and paypal.com is adequate to get started.

Regardless of how one bootstraps, one can get referrals by email, thumbdrive, web page, chat, and even by telephone.

Converting From Nickname to Petname

The other part of the system that cannot be easily quantified is the ease or difficulty of mimicking pet names. Let us assume a poorly built petname system in the clutches of a clueless user. Our user has a money transfer site on the Web that has been petnamed PayPal. The user gets an email telling him to update his PayPal account, he clicks the link, and goes to a domain that has given itself the nickname "PayPa1" (that last character, "1", is a one). Our poorly built hypothetical petname system is so poorly built, the nickname is put into the field where the petnames go with only the slightest indication that this is a nickname rather than a petname. The user sees no distinction and is phished.

Solutions to this problem are application and context specific, though some good ideas seem to have wide applicability. In the Waterken Petname Toolbar proposal, the alleged name is always "untrusted". It's hard to fail to recognize that this isn't PayPal, though a sufficiently unobservant user might completely disregard the petname information and get phished anyway.

There are a couple of user interface issues. The petnames must be unambiguously distinct from nicknames. This seems easy to do, through colors, fonts, additional text, and separate fields for the nickname as examples of pieces of strategy.

More difficult is the following problem: Petname creation must be both painless (or people will reject the whole idea) and reliably mimicry-free (it would be a disaster to have both PayPal and PayPa1 as petnames!).

Here are two example ideas for petname creation user interface that seem generally applicable. First is to compose the default choice for the petname out of a combination of contextual information and nickname information. Suppose we click on a link to "PayPal" in the Consumer Reports site (that is, the site that we have assigned the nickname, "Consumer Reports"). This takes us to a new site that proposes the nickname "PayPal". The system clearly marks that we do not have a petname for this site and proposes "Consumer Reports' PayPal". The user can press a button to accept this name, edit it, or press a second button that says, "let me use the raw nickname “PayPal” as the petname." This system still depends on the user remembering the petnames he has already assigned and noticing at the time of creation of the new petname, whether he already has a similar name in his list. In many circumstances this will not be a problem -- most of us who gave PayPal a petname would have no trouble remembering we had done so, and if we saw a suggested nickname that looked like "PayPal", we'd notice we were at risk of confusing ourselves if we accepted that similar name as the petname…but again, we are dealing with humans, so the process is imperfect. To support the human being, we'd want to use a font that was as ambiguous as possible during petname creation, mixing up 1 and l and I in a hopeless mess, so that one could be confident that the resulting petnames looked unique no matter what font was used later.

A second idea for keeping petnames reliably mimicry-free is to have a weak algorithm for comparison matching a candidate petname against the existing petnames. We explicitly call this a *weak* algorithm because it can be quite poor. It is quite acceptable for the algorithm to pop a list of "similar petnames" that is overly extensive, i.e., it is fine to show names that the human easily recognizes as distinct. The serious error is to fail to show names that the human might confuse. Comparing Paypal to Paypa1, a sample algorithm might notice that the names are of similar length and have three letters in common ("p", "a", and "y"), and say, "that's similar enough to be worrisome." The algorithm for noticing similarity between private petnames is under much less pressure to be perfect than is the algorithm for a Certificate Authority when deciding whether to award the name "pawpal" when the name "paypal" already exists. A CA might like to prevent mimicry, but to do so the CA must tread a difficult line between enabling many customers to have desirable names, versus abolishing huge swaths of namespace to ensure similarities don't arise.

Examples, Near Examples, and Comparisons

Physical World Petnames

Humans have been using parts of petname systems since before the invention of the written word. Human faces were used as keys. These keys resisted forgery far better than many things that pass for security today on computers (indeed, faces make such good keys, their forgery is so difficult and rare, that such forgery is used for entertainment, as in episodes of Mission Impossible, and the occasional Shakespearian comedy like 12th Night). The referral, "Og, this is my son Oop, he's great with a club," transferred both a key/alleged-name pair and a purposeful trust recommendation. The recipient of this referral typically accepts the alleged name as a petname.

These physical world petname systems are sufficiently different from computer-based petname systems that it is dangerous to draw too many conclusions from them. But the similarities are intriguing. More comprehensive comparison and contrasting of physical petnaming to computer-based petnaming is left as an exercise for the reader.

Trademark Law

Trademark law is not a petname system. When civilization started creating entities that did not have unforgeable faces (like Apple Computer), we settled on a legal system that attempted, with fair success, to enforce (that is, secure) purpose-unique memorable global IDs for small numbers of entities. It is hard to map trademarks onto petname systems for comparison, but an attempt seems in order. The trademark-purpose pair is the key, made unforgeable by government coercion. It is important to note that the trademark itself is not the key: Apple Computer and Apple Music both used the trademark Apple for decades, without conflict, until Apple Computer entered the music business.

The trademark by itself is the nickname: Apple Computer thinks of itself as "Apple". Petnames are absent. Mimicry is prevented by the same government action as forgery, and indeed the trademark system makes no distinction between forgery and mimicry (which perhaps helps explain why the distinction is so blurred in most discussions).

Trademark law depends on the legal system to disambiguate "similar purpose". This is expensive, and consequently trademark law can only apply to "small" numbers of "big" entities. The name John Smith is covered by trademark law, but only in explicitly recognizing that all people who have that name may use it, i.e., trademark law recognizes non-uniqueness in this case. On the Web, the number of entities with whom we would like to associate trust/vulnerability relationships is extremely large; indeed, one of the failures of the Web today is that we cannot construct as many such associations as we would like. Those relationships span multiple legal jurisdictions, further complicating the trademark system.

Instant Messaging Buddy Lists

Buddy lists for instant messengers follow the logic of petname systems. Each entity gets a globally unique id, rooted in the domain name of the messaging service, which fills the role of "key". Once the user puts a petname into the buddy list, all references to the id are displayed using the petname: you can connect to the entity using the petname, and when the entity connects to you, the petname appears, not the id.

While buddy lists are petname implementations, some instant messengers lose the security properties a petname offers because of efforts to overload the id with additional purposes. A weak effort is made to make the id both human memorable on the one hand, and unforgeable, on the other. The id is used as both a nickname and a key even though it is often hard to remember and easy to forge (either through man in the middle attacks or password attacks).

Despite these complications, buddy lists demonstrate the utility and usability of petname systems. Buddy lists are intuitive. People learn to use them with neither instruction nor documentation.

An instant messenger that used true keys, true nicknames, and enforced good security properties would be virtually indistinguishable in user-interface presentation from existing systems. Indeed, if one used an object-capability style of key, the biggest difference would be the absence of passwords, an actual usability improvement.

CapDesk and Polaris

CapDesk [CapDesk] and Polaris [Polaris] are desktop systems that explicitly flesh out petname systems to enforce security properties. CapDesk is a point and click desktop that combines usability, security, and functionality, to a degree often found surprising by those unfamiliar with it. In CapDesk, at application installation time the application proposes a pet text and a pet graphic (the icon for the top left corner of its windows, and the text in the title bar that is immediately adjacent). The user may accept this petname or modify it. Windows launched thereafter by the installed application are unforgeably marked with the petname. Limited informal experimentation suggested that the CapDesk petname system was is intuitive and easy to use, like the buddy lists.

Polaris is a derivative of CapDesk that defends the Windows user against several interesting classes of attack. Polaris uses pet texts similar to the CapDesk pet texts for marking the windows. Polaris is today being used in a larger set of pilot programs than CapDesk ever experienced. One result of the pilots that proved a pleasant surprise is that people are aware of and sensitive to the petname markings. This supports the hope that petnames could indeed strongly impact phishing.

Domain Names

DNS attempts to create memorable keys. It makes a striking example of the difficulties that result. The namespace being managed is so large that mimicry occurs, not only by malicious intent, but by accident. Mimicry becomes increasingly more problematic as systems grow to global scale since it is less likely that people can reliably remember all the DNS addresses they use.

Several of the other examples here treat the domain name as a key. Domain names are forgeable, but in practice they seem resistant enough to forgery to be useful. Judging by the prevalence of mimicry-based phishing over DNS forgery, it seems that at the time of this writing forgery is not the weakest link in DNS; mimicry is.

Browser Bookmarks

Browser bookmarks combined with URLs have many similarities to petname systems...with a significant flaw. Think of the URL as a key and a page title as a

nickname. The bookmark can then be thought of as a private name that points at the key, suggested by the nickname. It sounds like a petname system.

However, the bookmark is not a petname. As noted earlier, a true petname is a *two*-way mapping: any reference to the key is represented in the user's world as the petname. However, bookmarks only map from the private name to the key, with no mapping back. When you follow a bookmark to a page, or take any other path to get to the page, the domain name rather than the bookmark is used throughout the user interface as the "name" for presentation to the user, a fundamental violation of petname logic. Despite this violation, bookmarks demonstrate how even a partial implementation of petnames will deliver some defense against mimicry. Any person who reads an email allegedly from PayPal, and then clicks on an existing bookmark to go to PayPal rather than following the email-embedded link, gets a security benefit derived from the partial implementation of petnames afforded by bookmarks.

OpenPGP and the Web of Trust

OpenPGP keys carry nicknames with them, and in some implementations the user can replace nicknames with a name of the user's choosing, which would be a petname. When an entity's key is observed by the software, the pet name is properly presented, i.e., the petname is properly bidirectional. The Web of Trust supplies an interesting way to associate these keys with purposeful trust, by asking other entities who have vouched for the new entity, what they recommend as a trust relationship.

With all these features, OpenPGP supplies a true petname system architecture. OpenPGP has not been tested by phishing attacks yet. Since all the basic elements are there, the biggest question would be, how must the user interfaces for applications using OpenPGP evolve to face such a threat? This is another reminder that user interface is as critical for any practical security architecture as is the crypto. A security system whose user interface is written by cryptographers is no more likely to succeed than a security system whose authentication machinery is written by user interface designers.

Waterken Petname Tool

The Petname Tool [Waterken] is a FireFox extension explicitly based on petname architecture, explicitly to prevent phishing. A CA public key plus an organization name is treated as the key. The petname is a true two-way mapping between key and private name. For those sites to which the user assigns petnames, the tool supplies markings that make it easy for the user to unambiguously distinguish the site. Limited informal experimentation suggests that the petname tool is as intuitive as the buddy lists and desktop systems described earlier.

Certificate Authorities

Certificate Authorities create nickname/key pairs. The certificates share with PGP keys the cryptographic strength to ensure unforgeability. The claim is made that, because the

nickname is unique within the CA, interesting security properties may be ascribed to the nickname. Petnames are not included in the scheme. It looks like DNS, with the CA playing the role of the DNS root servers. In a world where web pages cross jurisdictional boundaries at the speed of light, it is hard to see how the standard CA concept could fix what DNS demonstrates so vividly is broken.

Trustbar

The Trustbar [trustbar] is a Certificate Authority-based FireFox extension that allows user construction of petnames for the certified entity. Like the Petname Tool, the TrustBar has been developed as a defense against phishing. In the 0.1 implementation, the distinction between a nickname (based on the certificate) and the petname is implied by the popup of a dialog box when the certificate is first encountered and no petname has yet been assigned. The petname and the key are not quite fully bidirectional: the key is properly represented by the petname in user interactions, but the petname cannot be used to get to the key.

Pet Name Markup Language

PNML is an XML proposal for using petname systems ubiquitously. In a chat system, if Bob made a reference to “Alice” in the text he wrote to Ted, and if “Alice” is Bob's petname for a person known to Ted with petname “Carol”, the sent reference to “Alice” would be converted on transmission to Alice’s key, and then converted on reception into Ted’s petname “Carol”. It would take significantly more effort to build PNML into an existing browser than to integrate the Petname Tool.

Conclusions

The informal experiments with petname-like systems identified here suggest that they can be intuitive and easy to use. A user who understands his petname system and is alert to the information it conveys can be quite resistant to mimicry, making that user a difficult target for phishing. Experimentation is required to determine how much less vulnerable to phishing the typical user would become given a petname system. Experimentation with petnames for web browsers does not have to be expensive; both the Trustbar and the Petname Tool are ready now, both for usage and for further experimentation by building variations derived from their open-source code.

Implementation Notes/Requirements

Following are important features of a petname system. If an implementation of a naming system does not include these properties, it is not fully following the logic of petnames:

- The key must be resistant enough to forgery to survive in the context of the application threat model.

- The petname binding and its corresponding key should endure and continue to designate the same entity for as long as that entity exists within the user's mental model.
- There can be at most one petname per key per user per application.
- There can be at most one key per petname (per user per application).
- In the application user interface, all references to the key are represented by the petname.
- The user must be able to assign a private petname to any key.
- The petname must be assigned to the key only by explicit user action.
- The user must be able to repeatedly edit the petname of any key.
- The user interface shall assist the user in assuring that two petnames are not similar enough to enable mimicry, to the extent necessitated by the complexity of the application context in which the petnames are selected and manipulated. If the number of petnames needed by the application is small and they are easily remembered, no assistance may be required. If the number of petnames is large, and/or difficult to remember and/or likely to be similar, and the resultant forms of mimicry, accidental or intentional, leads to vulnerability inside the threat model, assistance is required.
- Nicknames and alleged names must be unambiguously visually distinct from petnames.

Nicknames are optional.

Acknowledgements

We thank everyone on the cap-calk mailing list for their help, especially Ian Grigg for his deliciously relentless criticism, but also notably including David Hopwood, Alan Karp, Mark Miller, Tyler Close, Trevor Perrin, and Charles Landau, each of whom made comments that directly caused modification to the early draft. Petnames were invented at Electric Communities. Tyler Close was the first to recognize their usefulness as a defense against phishing. Thank you also to Amir Herzberg for his assistance in understanding the Trustbar.

References

[Zooko] http://zooko.com/distnames.html
[Shirky] http://shirky.com/writings/domain_names.html
[Close] http://www.waterken.com/dev/YURL/Name/
[Trustbar] http://eprint.iacr.org/2004/155/ or
http://www.cs.biu.ac.il/~herzbea//Papers/ecommerce/spoofing.htm
[CapDesk] http://www.skyhunter.com/marcs/CapDeskSpec.html or
http://www.combex.com/tech/edesk.html
[Polaris] http://www.hpl.hp.com/techreports/2004/HPL-2004-221.html

[Waterken] http://www.waterken.com/user/PetnameTool/
[PNML] http://www.erights.org/elib/capability/pnml.html