Naming in the Mathematical Mesh.

Mathematical Mesh Naming

<series>draft-hallambaker-mesh-naming

<status>informational

<stream>independent

<ipr>trust200902

<author>Phillip Hallam-Baker

<initials>P. M.

<organization>Comodo Group Inc.

<surname>Hallam-Baker

<email>philliph@comodo.com

<also>http://prismproof.org/Documents/draft-hallambaker-mesh-naming.html

<keyword>Naming

<keyword>Mathematical Mesh

<keyword>Authentication

Abstract

The < importance of naming in information systems is explained with reference to a typical use case. Existing Internet naming systems attempt to strike a balance between usability and machine readability. An alternative approach in which separate classes of identifiers are introduced for human and machine interaction is described. Human Interaction Identifiers are interpreted in the context in which they are used and are thus more compact and offer a superior user experience. Strong Internet Names are a form of Machine Interaction Identifier that are backwards compatible with existing Internet names that include a DNS address and are cryptographically bound to a security policy governing interpretation. The application of these approaches in the Mathematical Mesh is described.

This document is also available online at .

Use Case

Hollywood Alice arrives at the airport. Her hire car is already waiting for her at the curb, every aspect of its function precisely matched to her preferences. The radio is tuned to her favorite station, the seats, mirrors and driving controls adapt to her measurements and of course, the navigation system is already programed with her itinerary.

On the way to her destination, Hollywood Alice places a conference call to Bob and Carol by speaking their name. Bob wants to know if Alice’s new camera arrived in time for her trip. It did, but only just. Alice had only a few minutes to unwrap it and pack it for her trip. Carol has been to the area before and tells Alice she must see the Golden Gate bridge from Baker Beach, the same spot that Ansel Adams took his famous landscapes before and after the bridge was built. The navigator updates its route accordingly and Alice arrives in the golden hour.

Her camera is a large modern DSLR but it doesn’t have a GPS, it doesn’t need one as it automatically captures this data from her mobile phone via Bluetooth. Alice joins the camera to the conference and it uses the same connection to upload thumbnails of her pictures for Bob and Carol. When she gets to her hotel room, the camera will upload the raw files to Alice’s personal cloud using the high-speed connection.

It all works so much better in Hollywood.

An extremely wealthy real-world Alice being followed by a minibus containing a road crew of a half dozen IT support staff might possibly be able to create a similar user experience today but even that is doubtful. Instead of taking pictures of the Golden Gate bridge at sunset, Real-world Alice is far more likely to find herself at the side of the road debugging Perl scripts or more likely, turning all the ‘labor-saving’ technology off because it simply isn’t worth the hassle. We have all the parts but they just don’t fit together properly and even when they appear to be joined, they often come apart. A ‘seamless user experience’ that isn’t actually seamless makes things worse, not better.

There are many reasons that the scenario described above remains a pipe dream, not least the commercial interests of the technology providers. But one of the biggest obstacles to providing the seamless integration experienced by Hollywood Alice is the lack of a secure naming infrastructure. Providing Alice with a seamless user experience required seven data exchanges and ten trust relationships.

The only infrastructures currently capable of managing those trust relationships would require Alice to be in constant communication with some form of ‘identity provider’ and to surrender a large part of her personal control to that infrastructure.

There are proprietary infrastructures that provide an approximation of the user experience if Alice buys all her devices from the same vendor. But these require Alice to surrender even more personal control and in any case, there is no single company that makes phones, DSLR cameras and rents cars. And even if such a company existed, Alice would not be able to achieve seamless connectivity to Bob and Carol unless they live in the same walled garden as Alice.

This document explores the problem of secure naming and describes the approach to naming taken in the Mathematical Mesh, a Personal Public Key Infrastructure (PPKI) that is designed to make computers easier to use by making them more secure.

The central idea is this: Current Internet identifiers are designed to provide a balance between human and machine use. Separating the functions allows a vastly more satisfactory user experience, in most cases the name of the intended referent being implicit from the context in which it is used. The key to making this approach practical being the introduction of a new form of very explicit identifier which is cryptographically bound to the context in which it is interpreted but is nevertheless backwards compatible with existing Internet identifiers through the use of DNS prefixing.

Identifiers and Naming

In the field of semiotics, a name is a symbol whose meaning is purely conventional, that is the meaning of the name is based on nothing more than a common agreement to interpret it in a specific way. Most network identifiers are based on a name of some sort whether they be example.com or 127.0.0.1. The proper functioning of the Internet depends on all the participants agreeing on a common interpretation of these identifiers which in turn gives rise for the need for naming authorities.

Since a name is merely the result of an agreement, any naming scheme must be governed by an authority, even if that authority is just the one person who uses the names. The question of authority is thus an inescapable one when considering the security properties of naming schemes.

One of the core technologies that made the World Wide Web possible was the realization that the means to resolve any network resource may be specified by means of a triple, {what, where, how}. The Uniform Resource Locator (URL) is simply a syntax for expressing this triple in a single identifier: how://where/what. It seems intuitively obvious that whether these are expressed as one single unit or three ‘should’ make no difference. But the lightning success of the World Wide Web proves that combining all the information necessary into a single identifier makes all the difference in the world.

While a URL contains a name component, it is not a pure name: The interpretation of the where component is intersubjectively determined by the choice of ICANN as the ultimate DNS naming authority, the interpretation of what and how is determined by the protocol specifications. Different users attempting to resolve http://example.com/ in different parts of the world may not receive back the same exact sequence of bytes but provided they are connected to ‘the Internet’ they should receive back a representation of the same ‘network resource’.

Interpretation of the other type of Internet identifier users commonly encounter, the email address, is not so straightforward. Unlike a URL, the RFC822 email address does not contain a how component. It is implicit that alice@example.com is an SMTP email address, but this is only one of many ways in which it may be used today:

* Send SMTP email
* Initiate an XMPP chat
* Web site username
* Initiate a proprietary chat
* Comment on a Web forum

From the user’s perspective, it is natural to assume that the address alice@example.com is held by the same individual in each of these contexts. But like many aspects of the Internet today, it is a mistaken assumption. The holder of the email address may or may not be the same as the person who responds when a chat session is initiated.

An RFC822 email address is a combination of two names issued by separate authorities who@where. Since the where component is a DNS name, there is a certain degree of consistency in interpretation through the infrastructure managed by ICANN. No such guarantees are provided for the interpretation of who. A site may have a policy preventing reissue of account names to new users or it may not.

The use of email addresses as identifiers on third party sites introduces a third authority. For example, Alice may use her email address alice@example.com as her account identifier for the chat service provided by example.net. Resolution of the name example.com is now controlled by the unseen third authority example.net but only within the context of the example.net chat application.

While these concerns may appear abstract, the security consequences are anything but. It is the fact that alice@example.com could be the same person or different people that makes the Hollywood Alice experience unreachable today. Alice’s phone can’t integrate with her hire car or her camera or her cloud communications provider because they lack a common language for securely identifying themselves as belonging to Alice.

The identifiers we use in the Internet today represent a compromise between expressiveness and usability. The more information we attempt to pack into an identifier, the more a user must remember or type or read. If we are to introduce a cryptographically secure identifier it must therefore be a subclass of identifier, an identifier which is usually if not always hidden ‘under the covers’ and is separate from the names that the user sees and interacts with. This is the purpose that the Mathematical Mesh PPKI serves.

Identifiers in the Mathematical Mesh.

The Mathematical Mesh is a PPKI that manages two types of identifiers: Human-Interaction identifiers which are primarily designed to support human interaction and Machine- Interaction identifiers which are primarily designed to support interactions between machines.

As we saw in the previous section, these categories are not necessarily exclusive. Most of Internet names used on the Internet today were designed to provide a compromise between human and machine interactions. Such a compromise is not practical in a cryptographic infrastructure such as the Mesh since there is really no way to make an identifier that presents a cryptographic work factor of 2118 or better human friendly except by mapping it to another identifier designed for human use.

This section first describes the two types of identifier used in the Mesh and how one may be mapped to the other. The following sections describe the implementation of these ideas in the current implementation of the Mesh.

Human Interaction Identifiers

Human Interaction identifiers come in many different forms and a given user may use multiple identifiers to refer to the same entity in different contexts. Three different uses are distinguished:

Input

An identifier that the human user uses when entering a command. (e.g. “conference Bob and Carol”)

Output

An identifier that is presented to the human user when requesting user interaction. (e.g. “Alice Cryptographer is calling”)

Comparison

An output identifier that is used only to determine if two things are the same. Comparison identifiers are used in the Mesh when connecting a device to a profile.

Depending on the circumstances in which the identifiers are used, it is generally desirable that input identifiers be as compact as possible while output identifiers may provide more information. If Bob receives the conference request from Carol on his smartphone it is probably desirable for the display to present both the shortcut identifier from his personal contact directory ‘Alice’ and her full personal name ‘Alice Cryptographer’.

While most identifier forms used for input (text, voice) may also be used for output, the reverse is not true. A validated image of a subject’s trademark, as supported by the PKIX logotype extension provides a powerful means of telling a user which party operates the site they are visiting but would be highly impractical as an input format.

The Mathematical Mesh makes use of three different types of human-interaction identifier:

Global Identifier

An identifier that is intended to be globally unambiguous. (e.g. alice@example.com)

Contextual Identifier

An identifier that is only unambiguous within a specific context (e.g. “conference Bob and Carol”)

Implicit Identifier

An identifier whose referent is defined by the context in which it is used (e.g. “localhost”)

For most human interactions, it is desirable to use the shortest identifier possible for input that does not lead to ambiguity. Thus, the use of contextual identifiers is generally preferred over global identifiers and the use of implicit identifiers is almost always best.

Most of the identifiers used in the ‘Hollywood Alice’ scenario were implicit identifiers. The devices Alice used understood the target of the commands she gave by the context in which she used them. As will be seen, the introduction of Strong Internet Names at the machine level allows them to be eliminated at the human level.

The only contextual identifiers that Alice used were ‘Bob’ and ‘Carol’ which were names from Alice’s personal contacts directory. There are many Bobs in the world but only one ‘Bob’ in Alice’s contact book.

The Hollywood Alice scenario only involved three people and a set of devices owned or rented by Alice. There was no need for global identifiers because the scenario did not require Alice to interact with the wider world. But it is the ability to communicate and interact on a global scale that gives the Internet its full power. It is the ability to establish secure communication with practically anyone in the world that makes the Internet the primary engine of international commerce. It is also the capability that gives rise to most Internet security concerns.

User Expectation

One of the chief differences between human interaction identifiers and machine interaction identifiers is that humans interact with certain expectations that may or may not be met. It is the manipulation of the user’s expectations that enables many types of phishing fraud.

If a user sees a message that appears to come from a financial institution that they have a business relationship with and that expectation is not met, the result is likely to be some form of fraud on the user. Such failures are always and only the fault of the designers of the communications infrastructure. The user is never negligent, the user is never at fault if their action is the result of a good faith expectation of a different result.

When the Internet was new, it was often viewed as creating a reality that was distinct and different from the ‘real’ offline world. While this point of view is still a common position among Internet protocol designers, it is no longer the case for an increasing proportion of users. The Internet existed before they did, they have been using the Internet since before they could talk. For the Internet generation, there is no online world, only the world.

It is the fact that human interaction identifiers are bound to expectations that give rise to security concerns in defining the mapping from human interaction identifiers to machine. If we are to avoid the need to deal with expectations in the interpretation of Machine Interaction Identifiers, we must use cryptography.

Machine Interaction Identifiers

Traditional Internet names are designed to achieve a balance between human and machine interaction. As a result, these identifiers omit much of the context that we require at the machine level to avoid the need to address the issue of expectations.

For example, the identifier https://example.com/ specifies a resource that is to be retrieved by means of a TLS secured conversation but not the trust context in which the communication is to be established. While this is an acceptable, and in many cases an unavoidable situation for a human interaction identifier, it is a circumstance that is often unacceptable in the Internet of Things.

Take for example, a high-risk process control application such as the placement of control rods in a nuclear reactor. A control loop critical to plant safety is governed by means of a three-term (PID) controller connected to a temperature sensor, an actuator and the Supervisory Control and Data Acquisition (SCADA) system. We would wish for it to be possible for all these communications to be secured cryptographically but we are required by regulation to account for the correct operation of any infrastructure on which we rely. The introduction of any Trusted Third Party role whether that be a WebPKI CA or a ICANN managing the DNSSEC, is going to be unacceptable.

The Mathematical Mesh introduces a new form of name, the Strong Internet Name (SIN). A SIN is a name that is cryptographically bound to a security policy government the interpretation of the name. The use of a SIN is thus bound to a specific trust context.

The use of SINs in Mesh enabled applications closely resembles the use of DNS names and IP addresses at the network level. In normal circumstances, the user only interacts with DNS names which are a name designed for human interaction. But the Internet core has no understanding or knowledge of DNS names. The only identifiers understood at the narrow waist are IP Addresses and so we must translate the DNS names to IP addresses to establish Internet communications.

Mapping Human Interaction to Machine

To make use of a Human Interaction Identifier, it must be first mapped to Machine Interaction identifier. We consider this mapping to be secure if and only if this translation meets the good-faith expectations of the user. If a user encounters a Human Interaction identifier that leads the user to reasonably expect that they will be interacting with their bank, this expectation must be met or a serious security vulnerability is created.

The terms ‘reasonable expectation’ and ‘good-faith’ are of course subjective but not entirely without meaning. The various financial institutions that support the use of non-cash payments in the US operate under rules that absolve the user from blame or loss in almost any circumstance but nevertheless manage to profitably transfer an average of over $500 billion every day .

The Mesh does not impose a model for mapping human to machine interaction identifiers but it does allow the user to put that mapping under their personal control. Devices connected to a Mesh Personal Profile share the same view of the world; the same set of bookmarks and contacts for defining personal names and the same set of trust roots for Certification Authorities trusted to provide brokered trust.

In the existing model, mapping of the address https://example.com/ to a secure TLS endpoint takes place in two stages. First the DNS infrastructure is used to resolve the address example.com to an IP address. Next, the client begins making a connection to the host at the specified IP address, receives and validates a PKIX trust chain to a recognized authority and if satisfactory, declares the TLS connection trusted:

This approach is serviceable for the intended purpose of the WebPKI: Authenticating the Web sites that the user interacts with. It is less satisfactory as the basis for establishing connections in the Hollywood Alice scenario.

One of the biggest problems with the traditional approach is that TLS is only used to authenticate the server to the client. The user experience for client certificates remains unacceptable leaving usernames and passwords as the only available credentialing mechanism.

Consider the role of the camera in the Hollywood Alice scenario. Alice uses the camera to take pictures but this is only one of four interactions involving the camera. If we are to achieve the Hollywood Alice user experience, these other three interactions must take place without any intervention by Alice. But this is impossible if Alice is required to constantly enter passwords.

Another limitation of this approach is that the naming role of the Certificate Authority is limited to validating DNS names. This is a major constraint if our goal is moving beyond use of DNS for naming.

To achieve a satisfactory user experience, we need to reverse the order of operations. We make use of trusted authorities when mapping the human interaction identifier to a machine interaction identifier whose interpretation does not depend on a trusted authority.

The most important trusted authority for Alice is of course, Alice herself. While a typical Web user may visit hundreds or even thousands of different Web sites in a month, they will only buy from a rather smaller number and will in most cases use one or two financial institutions.

When Alice opens an account with a new financial institution, she adds them to her personal contact directory and (optionally) gives them a shortcut name. In the process, she reviews the credentials presented by the bank, a WebPKI Extended Validation certificate with a logotype extension presenting the Bank trademark. From the point at which the financial institution is added to Alice’s personal contact directory, the role of the Trust Authority is limited to revoking the trust relationship should the need arise.

If this approach is to work, we need a new type of Machine Interpretation Identifier, one that can express both the network addressing and the trust relationship to the network endpoint. Strong Internet Names are one possible approach towards that goal.

Strong Internet Names

A Strong Internet Name (SIN) is a name containing a DNS label that is cryptographically bound to a security policy government the interpretation of the name.

The use of fingerprints of cryptographic keys to establish was introduced in PGP. PGP fingerprints provide a secure means of authenticating the public key(s) of an email user but at the cost of introducing a second identifier for each user that senders must manage in addition to their email address. Like many issues in computing, this is a simple matter until faced with application software that does not support it.

Use of a SIN DNS label permits these two pieces of information to be combined in a single identifier that is compliant with existing application software by employing the DNS prefix approach introduced by punycode . It is established that DNS names of the form xx--<data> (where x is any alphabetic character) are reserved. A SIN DNS label has the form mm--<UDF-of-policy> where <UDF-of-policy> is the Uniform Data Fingerprint (UDF) of the controlling security policy as described in the following section.

UDF Fingerprint

The Uniform Data Fingerprint (UDF) format was designed to provide common format for representing fingerprints of data objects formed using a cryptographic digest function such as SHA-2 that was easier on the eye than existing URI schemes such as ni . A UDF fingerprint is formed using Base32 with optional digit separators to improve readability. The following is an example of a UDF:

pre

Unlike traditional fingerprints calculated from the digest of the data itself, a UDF is a strong function of both the referenced data and the IANA content type.

pre

This approach provides semantic separation between domains. This is necessary to defeat substitution attacks such as presenting an artfully constructed PKIX certificate in a context where a JSON data structure is expected.

The Version-ID parameter specifies both the digest function and the method of application. Version-IDs are currently defined for SHA-2-512 and SHA-3-512. The values of these code points have been intentionally chosen to cause the first digit to be either an M (Merkle-Damgard) or an S (Sponge).

The specification allows for fingerprint compression in the case that the leading 25, 40, 50 or 55 bits are all zero. This allows a fingerprint of a public key represented in 20 characters (120 bits) to present the same work factor to the attacker as a 25 character fingerprint but at the cost of accepting a 225 increase in key generation difficulty.

Strong Email Addresses

A Strong Email Address is an RFC822 compliant email address in which the DNS address part is a SIN bound to a security policy that is relevant to email. For example:

A PKIX End-Entity Certificate

An S/MIME certificate to be used for encryption and/or signature as specified by the certificate keyUsage extensions.

A PKIX Certificate Signing Certificate

A root or intermediary certificate under which end user S/MIME certificates must be validated.

A Mesh/Message Sender Policy

A comprehensive security policy description language that allows the user to specify the use of S/MIME and OpenPGP keys for encryption and signature and the context in which they are to be used. An example of a possible messaging security policy is described below.

For example, Example Inc. holds the domain name example.com and has deployed a private CA whose root of trust is a PKIX certificate with the UDF fingerprint MB2GK-6DUF5-YGYYL-JNY5E-RWSHZ.

Alice is an employee of Example Inc., she uses three email addresses:

alice@example.com

A regular email address (not a SIN).

alice@mm--mb2gk-6duf5-ygyyl-jny5e-rwshz.example.com

A strong email address that is backwards compatible.

alice@example.com.mm--mb2gk-6duf5-ygyyl-jny5e-rwshz

A strong email address that is backwards incompatible.

All three forms of the address are valid RFC822 addresses and may be used in a legacy email client, stored in an address book application, etc. But the ability of a legacy client to make use of the address differs. Addresses of the first type may always be used. Addresses of the second type may only be used if an appropriate MX record is provisioned. Addresses of the third type will always fail unless the resolver understands that it is a SIN requiring special processing.

When specified as the destination address in a Mail User Application (MUA), these addresses have the following interpretations:

alice@example.com

Send mail to Alice without requiring security enhancements.

alice@mm--mb2gk-6duf5-ygyyl-jny5e-rwshz.example.com

Send mail to Alice. If the MUA is SIN-Aware, it MUST resolve the security policy specified by the fingerprint and apply security enhancements as mandated by that policy.

alice@example.com.mm--mb2gk-6duf5-ygyyl-jny5e-rwshz

Only send mail to Alice if the MUA is SIN-Aware, it MUST resolve the security policy specified by the fingerprint and apply security enhancements as mandated by that policy.

These rules allow Bob to send email to Alice with either ‘best effort’ security or mandatory security as the circumstances demand.

Network Administration

Strong names may also be used for network configuration. Example Inc. might enable users to force users to make use of a SIN-aware email client by configuring the SRV records for the inbound and outbound mail servers as follows:

pre

Since the fingerprint mb2gk- is of a PKIX certificate signing certificate, the requirement to use TLS is explicit. This could be specified explicitly by means of a prefixed TXT record as described in .

For example, Alice is a user of the EXAMPLE service, a version management system used at Example Inc. This is a Web Service described by an SRV record as follows:

pre

Since the UDF fingerprint is used here as a parameter rather than as an embedded part of a DNS name, the mm-- prefix is unnecessary and can be omitted. Though it is probably good manners for applications to tolerate its occurrence in cases where it is unnecessary such as the second TXT record.

In this example, there are two separate TXT records describing the EXAMPLE service. The first record applies to all hosts that provide the EXAMPLE service and specifies a PKIX intermediary certificate under which the TLS certificate MUST validate if it is to be accepted. The second TXT record applies only to host1 and contains additional information specific to that host. In this case host1 offers the quic protocol but host2 does not.

It will be noted that this capability allows similar capabilities to the security policy capabilities provided by TLSA records , but in a form that is directly integrated into SRV discovery and offers greater flexibility. A Security Policy specified in a TXT record is not limited to the TLS protocol or even to TLS based key exchanges. The discovery mechanism described in has proven utility and is widely used. It is surely time to recognize this fact and back the winner rather than continuing to ignore it for the sake of a favored son.

The previous examples demonstrated the use of SINs to perform high level, site wide administration tasks. But the security policy specified in a SIN need not be limited to defining a site wide global root of trust. The following configuration file is used in a robotics project to authenticate command signals between the central controller and various control outputs:

pre

Specifying the computer systems controlling the appendages connected to the robot in this way permits all communications to be protected using strong encryption while ensuring that the system can continue to function even if it is impossible to validate the trust paths with respect to an external root of trust.

Security Policy Specification

The UDF format used to construct SINs is calculated over both the content data and the IANA content type. This allows the use of SINs to bind an identify to a security policy described in any language whether currently existing or to be defined in the future. A security policy specification may be explicit or implicit.

If a security policy is a PKIX Certificate Signing Certificate or End-Entity Certificate, the use of a security protocol consistent with the certificate attributes and protocol is required. This approach allows the use of SINs to require the use of an appropriate security protocol with specified credentials in a wide variety of legacy application protocols.

Implicit security policy is convenient but blunt tool. We can establish a baseline for security in the case that an email address SIN authenticates a PKIX end-entity certificate with the dataEncipherment key usage set (i.e. use of S/MIME encryption is required). But once that baseline security is defined, we can only improve on it by decorating the certificate with additional extensions to specify security policy. This approach is unlikely to be satisfactory in the long term.

The introduction of an expressive security policy language defined in an appropriate encoding (e.g. YAML, JSON, XML) offers much more interesting possibilities. For example, we would like an enterprise level security policy to allow specification of security policy parameters such as:

* The default DNS zone
* The UDF of the DNSSEC zone signing key
* The UDF of the PKIX enterprise CA
* The network directory protocols supported (e.g. LDAP)
* The authentication requirements for external network access
* IPSEC profile

At the user level, a security policy would describe the communication identifiers and protocols by which the person could be contacted. It is quite likely that these would be different depending on who is trying to contact them. End-to-end encryption is not an unqualified benefit when it provides an attacker with a channel for bypassing filtering for spam and malware. Thus, a user level security policy is likely to require conditional clauses:

All senders:

Must sign their email messages with public key enrolled in a Mesh notary log

If you are with example.com:

You may send me encrypted messages of any type, including executable code

If I have approved you as a sender:

You may send me encrypted messages but not executable code

If you are vouched for by a party I trust:

You may send me an encrypted contact request message containing no more than 4K of text characters

Otherwise:

Here is the public key of my spam filter.

While such rules are complex, it is complexity that a user would only ever encounter if they were trying to send a message that violated the rules.

As with any configuration language, the specification of a security policy language requires a balance to be struck between simplicity and expressiveness. Discovering the optimal balance is a task left to future work.

Resolving SINs

A SIN provides a mechanism for binding an Internet address to a means of authenticating a security policy under which the name is to be interpreted but does not necessarily provide a means for discovering security policies.

This omission is intentional as there are many circumstances in which we would want authorized parties to apply a security policy without disclosing the security policy to unauthorized parties. A security policy must inevitably disclose information that might interest an attacker and so it is information that we should not disclose to parties without a need to know.

When a synchronous protocol such as VOIP or chat is used, the security policy governing a SIN may be disclosed in-band during the protocol exchange in which the underlying DNS name is used. This approach does not work as well for asynchronous protocols such as email or for network administration.

The Mathematical Mesh provides one possible infrastructure that might be used to resolve SINs to the corresponding security policy, and using a linked notary log approach (aka Blockchain), a mechanism for publishing updates securely. It is not the only infrastructure that might be used, nor is it likely to be the best for every application. For example, a new machine connecting to an enterprise network for the first time might obtain its initial security policy through a DNS CERT record .

Personal Mesh

To complete the explanation of how to realize the Hollywood Alice scenario in practice, we turn to a brief overview of the Mesh itself. At the simplest level, the Mesh is simply a tool that allows a user to bind all their disparate electronic devices into one logical unit by means of cryptographic credentials that are in normal circumstances hidden from the user’s view.

To begin using the Mesh, Alice first creates a personal profile and registers it to a CryptoMesh portal. Alice’s personal profile contains a master profile containing a set of administrative keys that are used to sign updates to the personal profile and master signature key that is used to sign the master profile itself. The fingerprint of the master signature key is the user’s personal mesh fingerprint.

pre

A real-life Hollywood Alice would probably use an app on her smartphone for this purpose. For this paper, the command line tool to illustrate examples is more convenient.

The Cryptomesh is envisaged as an open co-operative infrastructure for management of public Mesh profiles. The CryptoMesh cannot suffer a confidentiality breach as all the data submitted to or created by the CryptoMesh is public. End-to-end confidentiality of private components of personal profiles is achieved by use of strong cryptography.

Use of the CryptoMesh provides the typical user with all the advantages of a cloud service without the usual disadvantage of being tied to a single cloud provider. Users may change their Mesh portal at any time without notice. All the information that is stored in the CryptoMesh is also stored on the user’s personal devices.

A user is not even required to use the CryptoMesh at all. Though any party who is security conscious enough to want to run their own private Mesh portal is likely to appreciate the fact that compromise of the private portal while undesirable will not result in a breach of the applications it is used to support.

Connecting Devices

Having created her personal profile on one of her devices, Alice’s next action is to connect more devices, her new DSLR camera for example. To do this, she simply runs the Mesh administration tool on the new device and specifies the name of the profile to connect to. The tool fetches the personal profile from the portal and reports the UDF fingerprint for Alice to check, should she want to do so.

pre

To complete the process, Alice must confirm the connection request on the first device. The manager provides a list of pending connection requests.

pre

Alice verifies that the connection request has the same comparison identifier. This identifier is a fingerprint of Alice’s personal mesh fingerprint and the fingerprint of the new device profile. Thus if the identifiers match, mutual authentication is achieved and Alice accepts the request:

pre

Given an appropriately secured booking system, Alice’s hire car may be connected to her personal profile automatically but only for the limited purpose of receiving commands and preferences from Alice and her connected devices. For the period of the rental, the hire car responds to Alice as its trusted user.

Connecting Applications

Having connected her devices, Alice begins connecting applications. Alice wants all the devices she has connected thus far to have secure email, SSH credential management and Web credential management:

pre

The Mesh approach to usability is to ask as little of the user as possible. Why bother to ask the user if they want S/MIME or OpenPGP credentials when it is as easy to provide both?

Alice connects her cloud storage provider to her personal profile, thus enabling its use by any of her devices that require data storage.

Contacts Directory

Having briefly described the Mesh itself, we may describe the use of the Mesh to support naming infrastructures. One such application is the use of a shared contacts directory across connected devices. This allows the user to create personal names or shortcuts for all the people and devices they might interact with.

Alice has created shortcuts in her Mesh contacts directory for ‘Bob’ and ‘Carol’. These shortcuts allowed her to establish the conference call with a voice command.

Acknowledgements

The ideas in this paper were developed over several years with the aid of Melih Abdulhayoglu, Robin Alden, Rob Stradling and Egemen Tas.

Security Considerations

This document describes the use of identifiers in the Mathematical Mesh and the security concerns that this gives rise to. The concepts of Expectations, Work Factor and Authority are explored.

Expectations

Whenever humans are required to interact with an identifier, their reasonable expectations must be met. Rather too often, it is the protocol designer’s expectations of the user rather than the user’s expectations of the protocol that have been primary.

Work Factor

No security infrastructure is invulnerable. Given infinite computing resources, even the strongest code can be broken. What matters is how difficult the security infrastructure is to break. When considering the strength of a cryptographic algorithm we consider the work factor measured in operations. Work factors of 2128 or greater are generally considered to be prohibitively difficult to break if the algorithm is secure regardless of future computing advances. Work factors of 2256 or greater present a generous safety margin.

Unfortunately, the cryptography used in a security system is rarely the weakest link. Thus, when considering systems rather than algorithms, an estimate of cost (i.e. in dollars or other currency) is more informative. The architecture of the WebPKI was originally developed using a work factor based on the estimated cost of obtaining a false credential and the expected criminal gain. Deterrence is achieved when the apparent cost is greater than the apparent gain.

As noted in a previous paper , linked notary log technology (aka Blockchain) makes the cost of a backdating attack, near infinite. This property may be used to advantage in developing a naming infrastructure.

Authority

The use of any identifier whose interpretation relies on the action of an external authority raises the problem of delegating trust. The problem is the same whether the authority be a single entity (e.g. ICANN) or multiple entities (e.g. WebPKI Certificate Authorities).

Terms of debate which allow one type of authority to be attacked relentlessly while holding the other as unimpeachable are unacceptable and must be discarded. Any party that proposes to act as an authority in any form of naming scheme must accept that they are accountable to the community they serve.

The approach described in this paper does not eliminate the authority problem but does allow it to be confined to the problem of mapping human interaction identifiers to other forms of identifier. Secure Internet Names are identifiers that are bound to a specific security policy governing their interpretation, allowing the role of any authority to be absolutely circumscribed.

IANA Considerations

This document has no considerations for IANA.