Self-Authentication

David Irvine, email: david.irvine@maidsafe.net, LifeStuff: David

maidsafe.net limited (registered in Scotland Sc 297540)

September, 2010 ***Abstract*—Today all known mechanisms that grant access to distributed or shared services and resources require central authoritative control in some form, raising issues in regard to security, trust and privacy. This paper presents a system of authentication that not only abolishes the requirements for any centrally stored user credential records, it also negates the necessity for any server based systems as a login entity for users to connect with prior to gaining access to a system.**

***Index Terms*—security, freedom, privacy, authentication**

CONTENTS

I. INTRODUCTION

AUTHENTICATION allows access to a system at a certain level or privilege. This is generally accepted as the privilege as granted by an authoritative 3rd party who owns or manages the particular service or resource being accessed. In cloud-computing or personal computing this is a limiting factor and a significant security risk. Trust in 3rd parties with personal or confidential data is something that is arguably impossible without any radical change in the very structure and fabric of modern society. This paper presents a system where there is no requirement for such a 3rd-party involvement.

A significant shift in thinking nowadays is to spread data across multiple locations for security and ease of access. This has surfaced privacy concerns, and increased awareness of ownership of data, something that is often disregarded very easily. But rarely all personal data is surrendered in this way; many systems offer some level of encryption to ensure privacy of data, but none offer any system of personal access to personal data, privately. In almost every case there is some form of contract, whether implied or actual between the 3rd-party service provider and the client. Furthermore the supplier may independently decide or be forced to act on the data, whether deleting encrypted data, going out of business or becoming a victim of damage or theft.

This situation is a crucial impediment to personal freedom, and without a change in technical capabilities that allow the mindset change that appears more prevalent as time goes by, there will be a significant gulf between people's individual desires and technology's ability to deliver. This in itself may impede progress and innovation, which would be an enormous failure of Science and Engineering to take responsibility for.

This paper will outline and detail a significant mind-shift in access controls that not only answer these issues, but take the current situation and dramatically alter our relationship with technology, particularly in regard to storing, sharing and developing our most personal thoughts, aspirations and desires.

II. IMPLEMENTATION

A. Issues to be solved

Given a traditional resource exchange, the bargain between involved parties tends to be direct and physically local. However, the de-facto replacement of barter by monetary systems in modern societies introduced the requirement of trust in third parties providing and controlling the necessary infrastructure, such as banks and financial authorities.

It is an illogical consideration to have created a technology based solution which requires this demand of trust, and to do so in a manner that is almost uncontrolled. Technology tends to be based on logic, thus it would appear obvious that creating, sharing and retrieving information fed into a computing device

by a person should not require that computer to connect to a network of computers with a controller or guardian that is not a system of pure logic.

A significant reason for the current situation is the inability for identities to be created, managed and personally controlled. This is a reasonable request from people to make from their technology, but so far has been regarded as impossible by technology professionals. A system of personal identity management is fundamental for the removal of the illogical situation of today.

B. Conventions

There is scope for confusion when using the term "key", as sometimes it refers to a cryptographic key, and at other times it is in respect to the key of a DHT "key, value" pair. In order to avoid confusion, cryptographic keys will be referred to as K, and DHT keys simply as keys.

- $\mathsf{H} \equiv$ Hash function such as SHA, MD5, etc.
- $\mathsf{PBKDF2}_{[\mathsf{Passphrase}][\mathsf{Salt}]} \equiv$ Password-Based Key Derivation Function or similar
- $\mathsf{SymEnc}_{[\mathsf{K}]}(\mathsf{Data}) \equiv$ Symmetrically encrypt Data using K
- $\mathsf{SymDec}_{[\mathsf{K}]}(\mathsf{Data}) \equiv$ Symmetrically decrypt Data using K
- $+ \equiv$ Concatenation
- $\mathsf{PutV}_{[\mathsf{Key}]}(\mathsf{Value}) \equiv$ Store a Value under the given Key
- $\mathsf{GetV}_{[\mathsf{Key}]} \equiv$ Retrieve the Value identified by Key
- $\mathsf{DelV}_{[\mathsf{Key}]} \equiv$ Delete the Value identified by Key

C. Overview of self-authentication

1) Requirements: Self-authentication requires a storage mechanism accessible by the users of the system. This may be public (such as a peer-to-peer network) or private (such a a storage area network); this paper assumes that it should also be a key addressable storage system.

2) Methodology : Self-authentication relies on a system where an entity can create a unique key to store a value in the storage system. The value stored with this key should contain an encrypted passport to data. This passport may be cryptographically secure keys and or a list of other keys to make use of the information to be stored and or shared as well as any additional components required.

The location of this initial key should be masked or at least not obvious in the storage mechanism. Further masking should be considered. This simplified approach is the basis for self authentication and is extended into a system that is capable of security in a manner that allows access data to be stored publically and with no additional requirement such as firewalls or access controls.

III. Detailed implementation

A. Creation of an account

Here we will assume there are two inputs from the user of the system: user-name U, password W. A salt S is calculated by hashing the U and W (twice) and then reading the result and generate a number between 1000 and 5000 e.g. $ReadHash()$ reads a bit from the hash $H(H(U+W)) = P \quad X = ReadHash(P)\ While(X\%4000 \leq 0)\{X;\}\ X + 1000 = S.$

To generate a unique identifier, we hash the concatenation of the user-name and the salt, $\mathsf{H}(\mathsf{U}+\mathsf{S})$.

PBKDF2 is used here to strengthen any password keys used. This is required as user selected passwords are inevitably weak and the user may not know the user-name is also used as a password in the system. Account specifies session data, like user details or an index of references to further data. This packet is located through an Access Packet holding a random number Rnd#.

1) $\mathsf{GetV}_{[\mathsf{H}(\mathsf{U}+\mathsf{S})]} \equiv \mathsf{False}$ (Ensure uniqueness)
2) Generate random number Rnd#
3) $\mathsf{PutV}_{[\mathsf{H}(\mathsf{U}+\mathsf{S})]}\left(\mathsf{SymEnc}_{[\mathsf{PBKDF2}_{[\mathsf{U}][\mathsf{S}]}]}(\mathsf{Rnd}\#)\right)$ (Store Access Packet)
4) $\mathsf{PutV}_{[\mathsf{H}(\mathsf{U}+\mathsf{S}+\mathsf{Rnd}\#)]}\left(\mathsf{SymEnc}_{[\mathsf{PBKDF2}_{[\mathsf{W}][\mathsf{S}]}]}(\mathsf{Account})\right)$ (Store Account Packet)

B. Login / load session process

1) $\mathsf{SymDec}_{[\mathsf{PBKDF2}_{[\mathsf{U}][\mathsf{S}]}]}\left(\mathsf{GetV}_{[\mathsf{H}(\mathsf{U}+\mathsf{S})]}\right) \equiv \mathsf{Rnd}\#$
2) $\mathsf{SymDec}_{[\mathsf{PBKDF2}_{[\mathsf{W}][\mathsf{S}]}]}(\mathsf{GetV}_{[\mathsf{H}(\mathsf{U}+\mathsf{S}+\mathsf{Rnd}\#)]}) \equiv \mathsf{Account}$

For the following operation, Rnd# should be kept locally for the duration of the session.

C. Logout / save session process

1) Generate new random number $\mathsf{Rnd}\#_{\mathsf{new}}$
2) $\mathsf{PutV}_{[\mathsf{H}(\mathsf{U}+\mathsf{S}+\mathsf{Rnd}\#_{\mathsf{new}})]}\left(\mathsf{SymEnc}_{[\mathsf{PBKDF2}_{[\mathsf{W}][\mathsf{S}]}]}(\mathsf{Account})\right)$ (Store new Account Packet)
3) $\mathsf{PutV}_{[\mathsf{H}(\mathsf{U}+\mathsf{S})]}\left(\mathsf{SymEnc}_{[\mathsf{PBKDF2}_{[\mathsf{U}][\mathsf{S}]}]}(\mathsf{Rnd}\#_{\mathsf{new}})\right)$ (Update Access Packet)
4) $\mathsf{DelV}_{[\mathsf{H}(\mathsf{U}+\mathsf{S}+\mathsf{Rnd}\#)]}$ (Delete old Account Packet)

D. Fallback Account Packets

The previous sections outlined the basic system of authentication. However, this can be extended for safety reasons. As with any system, the serialisation and store operation of the account data can fail for any reason, resulting in unreadable data upon retrieval. This would be catastrophic as access to the user's data on the system may be rendered impossible. To reduce any such risk, a fallback copy of an Account Packet (and its Access Packet) is kept, to allow reverting to the previous version in case the current version can't be restored or turns out to have been generated erroneously.

1) Updated account creation process:

1) $\mathsf{GetV}_{[\mathsf{H}(\mathsf{U}+\mathsf{S})]} \equiv \mathsf{False}$ (Ensure uniqueness of Access Packet)
2) $\mathsf{GetV}_{[\mathsf{H}(\mathsf{U}+(\mathsf{S}-1))]} \equiv \mathsf{False}$ (Ensure uniqueness of Fallback Access Packet)
3) Generate random number Rnd# and copy $\mathsf{Rnd}\# \rightarrow \mathsf{Rnd}\#_{\mathsf{fallback}}$
4) $\mathsf{PutV}_{[\mathsf{H}(\mathsf{U}+\mathsf{S})]}\left(\mathsf{SymEnc}_{[\mathsf{PBKDF2}_{[\mathsf{U}][\mathsf{S}]}]}(\mathsf{Rnd}\#)\right)$
5) $\mathsf{PutV}_{[\mathsf{H}(\mathsf{U}+(\mathsf{S}-1))]}\left(\mathsf{SymEnc}_{[\mathsf{PBKDF2}_{[\mathsf{U}][\mathsf{S}-1]}]}(\mathsf{Rnd}\#)\right)$
6) $\mathsf{PutV}_{[\mathsf{H}(\mathsf{U}+\mathsf{S}+\mathsf{Rnd}\#)]}\left(\mathsf{SymEnc}_{[\mathsf{PBKDF2}_{[\mathsf{W}][\mathsf{S}]}]}(\mathsf{Account})\right)$

In this case, the random numbers in the Access Packets are the same, thus point to the same (unique) Account Packet. Fallback packets are only kept once the Account Packet is updated.

2) Updated Login process:

1) if ($\mathsf{GetV}_{[\mathsf{H(U+S)}]} \equiv \mathsf{EncRnd\#}$)
   a) $\mathsf{SymDec}_{[\mathsf{PBKDF2}_{[\mathsf{U}][\mathsf{S}]}]}(\mathsf{EncRnd\#}) \equiv \mathsf{Rnd\#}$
2) else (or if previous attempt failed)
   a) $\mathsf{SymDec}_{[\mathsf{PBKDF2}_{[\mathsf{U}][\mathsf{S}-1]}]}\left(\mathsf{GetV}_{[\mathsf{H(U+(S-1))}]}\right) \equiv \mathsf{Rnd\#}$
3) $\mathsf{SymDec}_{[\mathsf{PBKDF2}_{[\mathsf{W}][\mathsf{S}]}]}\left(\mathsf{GetV}_{[\mathsf{H(U+S+Rnd\#)}]}\right) \equiv \mathsf{Account}$

3) Logout / save session process:

1) Generate new random number $\mathsf{Rnd\#_{new}}$
2) $\mathsf{PutV}_{[\mathsf{H(U+S+Rnd\#_{new})}]}\left(\mathsf{SymEnc}_{[\mathsf{PBKDF2}_{[\mathsf{W}][\mathsf{S}]}]}(\mathsf{Account})\right)$
3) $\mathsf{PutV}_{[\mathsf{H(U+(S-1)}]}\left(\mathsf{SymEnc}_{[\mathsf{PBKDF2}_{[\mathsf{U}][\mathsf{S}]}]}(\mathsf{Rnd\#})\right)$
4) $\mathsf{PutV}_{[\mathsf{H(U+S)}]}\left(\mathsf{SymEnc}_{[\mathsf{PBKDF2}_{[\mathsf{U}][\mathsf{S}-1]}]}(\mathsf{Rnd\#_{new}})\right)$
5) $\mathsf{DelV}_{[\mathsf{H(U+S+Rnd\#_{fallback})}]}$
6) $\mathsf{Rnd\#} \rightarrow \mathsf{Rnd\#_{fallback}}$
7) $\mathsf{Rnd\#_{new}} \rightarrow \mathsf{Rnd\#}$

The previous Account Packet remains untouched, instead the Fallback Access Packet is redirected to it and the normal Access Packet points to the new Account Packet. The previous Fallback Account Packet is deleted. This is a security measure to hinder slow brute-force attacks on decrypting the Access Packet, which by the time the clear-text random number is obtained would make it obsolete.

E. Further enhancements

1) Time based obfuscation: In the previous sections a system of self authentication was detailed which is very effective. A potential failure point, however, may be the Access Packets, as those are never altering their location (key) and can pose a target for any attacker who is monitoring data traffic between the user and the system.

To remove this weakness, a predictably altering piece of data can be introduced, such as time (e.g. week number, day of year or similar). In this case in III-B the GetV call would be iterative, starting at the current time slot and decrementing until it returns with a value. Access Packets at "outdated" locations would be deleted on detection, and updates always stored in the current location, resulting in regularly "moving" packets.

2) Distributed storage system: This system can be enhanced with the introduction of a distributed storage network as described in [4]. This has many advantages including the ability to mask any account data in a large address space and protect with cryptographically secure privileges that can prevent unauthorised deletion or any loss of data packets.

IV. Conclusions

The system presented is one version of a server-less authentication system. There is a strong case for this to enable true cloud based services with users being in control of their own data and access privileges. Such a system can be applied in a multitude of situations and may be particularly useful with paid services, which would not require the customer to divulge any personal information or create yet another identity specifically for a single service, providing a shared infrastructure exists.

We demonstrated a working version of such a system for the first time in April 2008 in our offices in Troon, Scotland, and as far as we are aware this was the first time in history a person created their own identity, stored it and managed all their actions without any server requirement and without any 3rd party control.

References

[1] as described by Van Jacobson in this link below, August 30, 2006 http://video.Google.com/videoplay?docid=-6972678839686672840
[2] David Irvine, Self Encrypting Data, david.irvine@maidsafe.net
[3] David Irvine, "Peer to Peer" Public Key Infrastructure, david.irvine@maidsafe.net
[4] David Irvine, maidsafe: A new networking paradigm, david.irvine@maidsafe.net

David Irvine is a Scottish Engineer and innovator who has spent the last 12 years researching ways to make computers function in a more efficient manner.

He is an Inventor listed on more than 20 patent submissions and was Designer / Project Manager of one of the World's largest private networks (Saudi Aramco, over $300M). He is an experienced Project Manager and has been involved in start up businesses since 1995 and has provided business consultancy to corporates and SMEs in many sectors.

He has presented technology at Google (Seattle), British Computer Society (Christmas Lecture) and many others.

He has spent many years as a lifeboat Helmsman and is a keen sailor when time permits.