Managing Trust: Real World Examples

Jerrold Poh
The University Of Auckland
jpoh004@ec.auckland.ac.nz

Abstract

Security on the Internet has always been a big concern, especially now a days with the popularity of e-commerce, where goods can be purchased online and where banking and share trading can be done with the click of a mouse. Structures though, have been put in place to help prevent online fraud, but a lot of these structures are built on a misunderstanding of the word trust. In this paper, I will be exploring the definition of the word trust and how systems can be put in place to take into account these new meanings.

1. Introduction

With more and more financial transactions occurring over the Internet, trust is starting to become more and more important to us online. Before businesses started setting up shop on the Internet, residents of the Internet mainly consisted of research facilities and Universities, so not a lot of emphasis was put on trust. For example, back then the SMTP protocol [1] didn't contain any authentication requirements whatsoever. This allowed anyone on the Internet to access any SMTP server they could find online to send their own personal email. This also allowed anyone to forge their own *from* address, making it extremely easy to forge an identity via email. Businesses then came on the Internet and usually, when money starts getting involved in anything, the rules of the game start to change [2].

What people have tried to do to make the Internet more secure is set up structures online which are similar to structures which are set up in real life. For example, most bars or clubs that I try to enter usually ask me for ID before I go into them. For the majority of these places, taking my word that I'm over 18 isn't usually an option. What the bouncers need is a form of identification from a trusted third party to verify my identity. The trusted third party in this case is the government, and the form of identification is a drivers licence.

To relate this example to something online, a trusted third party is usually a Certificate Authority (CA for short) and the form of identification is usually an identity certificate issued by the CA.

So before we go any further with this example, what exactly is a trusted third party and what does it mean to say they are trusted?

2. Trust

Most people think the statement "I trust Joe Bloggs" is a complete statement. Trust in actual fact is not a complete word, but a word which can be used to represent a subset of words. They're varying forms of trust, and they're also varying degrees of trust. Rarely are we able to use the word trust without qualifiers, especially when they're humans involved [3]. For example, I may recommend a certain moving company to a friend because I've used them before, and I might say to my friend that I trust them. On his end, he may think that I trust them to move his goods without damaging them, but on my end that trust might be that I trust them not to steal anything. They could very well be that they are the clumsiest moving company in the world, but without adding qualifiers to my trust statement to say exactly what I trust them in doing, there has been a huge breakdown in communication and in the financially centred e-commerce world, communications breakdown isn't a good thing.

So going back to the drivers licence example. I mentioned that the trusted third party was the government, but what exactly is it about the government that we trust? We trust that their main motivation for doing anything is to look out for the interest of the people, and that they would do everything in their power to have everyone fairly represented. From that we can conclude that the certificates that they generate would be genuine. Do we also trust the government to keep our identities safe from people who might fraudulently misuse them? What about their drivers licence manufacturing facilities, do we trust them to be well guarded so as to stop people printing their own drivers licences? And what about the drivers licence delivery system. Are they reliable in matching the correct people with the correct drivers licences?

Now that I've listed the different types of trust that we have when we say we trust the government, I can now talk about the levels of trust. If there was to be an inconsistency found between what is printed on a drivers licence and what a customer was saying, most people would trust what the drivers licence says instead of the customer. This is because from past dealings with the government, they have shown to

be in the wrong for only a very small proportion of the time, so the level of trust that we have with the government (despite the many things which could go wrong) is much higher compared to the level of trust we have in the customer.

In the case of a CA though, the examples above might not hold true due to the many differences which exist between a CA and the government. The main difference being that the government is set up by the people, for the people, whereas a CA's existence is created due to a collection of people who have found a common interest, which is that of making a profit. There is also not just one CA like there is just one government, they're multiple CAs on the Internet, each fighting for your dollar.

Note that the system above was created to emulate a system already established in real life, but as you can see, what works in real life doesn't necessarily translate well onto the Internet. What we need is another way to certify someone's identity, or at least improve this method to remove some of the ambiguity of what exactly it is we trust.

3. Web of Trust

3.1 Description

From description above, it we can see why some people might distrust the idea of having CAs, which happens to be one of the solutions proposed to solve this problem, which is to get rid of CAs altogether. One of the systems in place which remove the need for CAs is PGP. PGP stands for Pretty Good Privacy, and is a system in which emails can be encrypted and sent securely to people using public key encryption. The ingenuity of this protocol is not in it's encryption method, but of it's way of establishing identity.

The process that occurs goes like this [4]. Lets say Bob wishes to speak to Alice. What Bob will do is send a message to Alice encrypted with Alice's public key. This message can now only be decrypted with Alice's private key (which Alice would hold the only copy of, and who would also keep private). If Alice's public key was given to Bob without interception (for example in person) then even if there was an interception of Alice's email, the intruder still can't read Alice's email because Alice would be the only one who would have the private key to decrypt the email. So as long as the public key can be guaranteed to have been exchanged without interception, the system should be relatively safe. This doesn't include the factors which plague every other security system though, for example social engineering, death threats, or more

common, money.

So how does PGP establish identity? What you need is an introducer. Lets say Alice wishes to talk to someone called Carl but Alice can't get Carl's public key in person due to geographical impairments. Bob on the other hand has met Carl once while he was overseas, and while overseas, was able to secure a copy of Carl's public key. So to get that public key to Alice, what Bob can do is sign Carl's key (to say that it's genuine), encrypt it with Alice's public key, and send it to Alice. On Alice's end, she now knows that the public key that she is getting will be the correct public key for Carl, and since it's signed with Bob's signature (and verifiable to be Bob's signature) Alice can now partake in secure communications with Carl.

Imagine now, this whole system replicated around the whole of the Internet, with everyone managing their own system of who they trust and don't, with links not in a hierarchy, but in the form of a web, which is probably why this structure is called the Web of Trust.

This managing of individual trust systems also allow some fancy tricks depending on how paranoid you are. For example, instead of Alice trusting Carl's public key which Bob sent, Alice can say that she needs at least two signatures from people she trust to sign Carl's key first before she's willing to use it. That way it isn't as limiting as with CAs, where each certificate you get from each CA costs money. This way you can have multiple people to authenticate a public key.

3.1 Problems

As good as this system sounds right now, they are some problems, not just in the translating of this into an e-commerce type situation, where the identity of a merchant and vendor need to be verified, but also in the existing email infrastructure.

At the moment we're assuming that the community on the Internet is rather small but in larger situations it is much harder to imagine a web of trust structure like this working very well. For example lets say that I wish to speak to an old friend who I haven't seen in years and I manage to find his email address and public key off a personal web site which he created years ago. So I download his public key and find that it isn't signed by anyone I know. What's the next step which I proceed to verify his identity? I could ask him personal questions which only he and I would know the answers to, but that still doesn't stop a man in the middle attack who has intercepted my old friend's personal key and replaced it with his own. Lets say that I do track a couple more links in the chain to link to my old friend, but the thing is I'm unsure of how

credible each of the links in the chain are. If one link the chain is broken, then the whole trust system falls apart. That doesn't mean that there isn't another link to my friend, but what if my friend is new to the web of trust and hasn't established any other links yet?

3.2 Solutions

One proposed solution [5] is to calculate trust percentages for each path from one entity to another. That way it is possible to set a threshold of how much trust is required before it is deemed safe to ensure that the public key is mapped to the right person. Say Darryl, Edgar and Fred are people in this web of trust, and Darryl wants to talk to Fred. So lets say Darryl trusts Edgar about 95%, and Edgar trusts Fred 95%, so using the same maths that we would if we were calculating probabilities, we can see that the link from Daryl to Fred would be about 90% trustworthy (0.95 x 0.95). Lets say instead, I have two links from Daryl to Fred, and they're all 95% trustworthy as well. After doing some maths, we can work out that the link from Daryl to Fred is now about 99% trustworthy 1 - (1 - (0.95 x 0.95)) (1 - (0.95 x 0.95)). Obviously having more than one path to a person would increase their percentages, and it's good to see that this system works just like a probability tree, because in essence, this is what it is (which is both good and bad).

Note, for this to work the trust percentages would also have to be uniform across the whole of the Internet, so there would have be standards in place [5] of what percentages are for how well you know this person and how you obtained this person's key (i.e. in person).

The vital flaw in this idea though is that trust isn't transitive [6]. Just because Alice trusts Bob doesn't mean that that trust is transferable to Carl. It also fails to address the different forms of trust which are present in the system. In PGP (much like the system above), you are able to set how much you trust a party, with the levels you can set being: *unknown, un-trusted, marginally trusted,* or *completely trusted*. This seems to just address only one form of trust, even though at minimum there is at least one other form which has been missed out [6].

So using the same example as we used previously, we can see that Alice trusts Bob to be a good person. With this in mind, Alice can put the level of trust which she has towards Bob as *completely trusted*. Because Bob is *completely trusted*, the key which Bob has passed to Alice has a high probably of being the correct public key which maps to Carl.

The trust that Alice has towards Carl though might not necessarily be the same as the trust she has towards Bob. Alice might just trust that the public key for Carl is the right one which maps onto Carl, but that doesn't necessarily mean that she trusts Carl to be a good person. So what level of trust does Alice put for Carl? Maybe it would be more convenient to have two scales, so one would be used to say how trustworthy a person is as an introducer, and another scale which says how trustworthy is that public key which is actually mapped to that person.

Maybe the previous trust percentages [5] could be calculated with these two types of trust in mind, but even still, that still doesn't take into account that trust isn't transitive. If the formulas that calculate the trust percentages can though, decrease the percentages for each transitive form of trust encountered, it could be a good system to get a general feel of how much you can trust a person.

All of these systems seem to be doing different things, but it feels like they're making changes to the same layer. This is kind of what Trust Management Systems address in the next section.

4. Trust Management Systems

4.1 Description

So to get a better understanding, lets go back to the original problem which was presented in the *Trust* section of this paper. What can we do to ensure that we can correctly identify people on the Internet?

Trust management systems are something proposed about seven years ago by M. Blaze, J. Feigenbaum, and J. Lacy in their seminal paper titled "Decentralized Trust Management" [6]. This paper was one of the first to address that they are different forms of trust, and that none of the existing identification methods have addressed these forms of trust.

The system which they proposed was called PolicyMaker, and it wasn't anything concrete with a definite set of rules which would guarantee the identity of a person. Instead, it provided a tool which you could use to write these rules yourself. It, in a sense, created another layer of abstraction in the existing security model. With the presence of PolicyMaker, people were forced to be made aware of this layer and once aware, they could then understand that they were different forms of trust and have new policies written in the PolicyMaker layer to strengthen their existing security policies.

Below is a step by step process of what a typical application must go through to process an identity certificate [6].

1. Obtain certificates, verify signatures on certificates and on application request, determine public key of original signer(s).
2. Verify that certificates are un-revoked.
3. Attempt to find "trust path" from trusted certifier to certificate of public key in question.
4. Extract names from certificates.
5. Lookup names in database that maps names to the actions that they are trusted to perform.
6. Determine whether requested action is legal, based on the names extracted from certificate and on whether the certification authorities are permitted to authorize such actions according to local policy
7. Proceed if everything appears valid.

What policy maker does is replace steps 3 to 6, so that it turns the above steps into this:

1. Obtain certificates, verify signatures on certificates and on application request, determine public key of original signer(s).
2. Verify that certificates are un-revoked.
3. Submit request, certificates, and description of local policy to local "trust management engine"
4. Proceed if everything appears valid.

You'll notice that steps 3 to 6 are steps which need to be performed by every application which processes certificates. Every application implements these steps differently though and these are the steps which are usually the most dangerous if they are implemented incorrectly. "The problem of reliably mapping names to the actions they are trusted to perform can represent as much of a security risk as the problem of mapping public keys to names, yet the certificates do not help the application map names to actions" [6]. So by eliminating steps 3 to 6, what PolicyMaker does is instead of binding keys to names, it binds the keys to what they can sign for.

For instance I can set up a policy so anyone can use their key to browse through my online bookstore. I don't need to know who they are, and for the time being, nothing complicated needs to be done which would compromise the integrity of the

system. If they want to buy something though, just their own signature on the purchase order wouldn't be enough since I can't take the word of a complete stranger. I would set up a policy in PolicyMaker so that if they were to buy something, not only is their own key required so sign a purchase order, but another key from a trusted third party has to sign as well to ensure that they'll pay.

What if they would want to put up a book to sell on my web site? I could set up another policy so that this person not only needs another key from a trusted third party, but five other keys who are also signed up on my web site to sign it. Lets say I'm also part of an online community, so signatures from other parts of the community can be trusted as well.

4.1.2 Advantages

So like I said before, it's advantage is that you can set up very complex trust structures which you otherwise wouldn't be able to do with the existing applications which the PKI is built on. Only trivial modifications need to be done to existing applications to take advantage of the huge gains you get from using a trust management system. Theoretically, you could even implement the trust percentages I discussed earlier with this system in place.

Trust management systems also appear to provide the right layer of abstraction to the developer and doesn't take on more tasks that it needs to. Everything is modularized and each module is responsible for it's own job, which follows the UNIX design philosophy quite well. That is that everything should only have one job to do, and do that job very well. The SMTP protocol is responsible for the delivery, the MUA is responsible for reading the mail, PGP is responsible for decrypting the keys, PolicyMaker is responsible for enforcing the policies, vim is used for editing the emails, etc.

It is also good in that you can run PolicyMaker on a different machine which is separate from the machine which receives requests. For instance a voting system of some sort could be set up via a web site, where what votes each user voted is kept secret, but only users who have registered to begin with can vote.

So lets say machine A is the web site with the voting program on it. The user first sends in a vote with his signature to machine A, and machine A then sends the vote machine B which has the PolicyMaker daemon running on it. PolicyMaker approves that the signature is allowed to vote, and then sends an approved to machine A. So machine A never knows the identity of the user and the user can vote in complete

anonymity.

The final thing that I like about it is that it isn't built in a hierarchy structure, or a web structure, but a combination of the both. It isn't in danger of falling over if the root node falls over in the hierarchy structure, and it isn't in danger using random ad hoc certificates as in a Web of Trust structure.

For example, it is possible to set up rules to trust Bob of the @bob.com domain to sign certificates for people in the @bob.com domain, but maybe not trust everyone in the @bob.com to sign other people's certificates. It is also possible to trust people in the @alice.com domain if Bob has signed a user's key there, but not necessarily trust Alice when she signs someone in the @alice.com domain.

4.1.3 Disadvantages

There is only really one disadvantage which I can think of with trust management systems, and that is that it doesn't really provide any rules to build a solid system, but instead it provides a tools to build these systems. Trust management systems provides lots of rope which either can be used by an experienced user to build a good system, or by a inexperienced user to hang himself with.

Conclusion

In the abstract of this paper I mentioned that I was going to be "[...] exploring the definition of the word trust and how systems can be put in place to take into account these new meanings".

Just by reading the different methods which I have discussed, it looks to me that when you trust someone you're, at some fundamental level, delegating responsibility to that person. When you trust someone as an introducer of a public key, you're delegating the responsibility to that person in providing you the correct public key so if something goes wrong, you have someone to blame. Depending on which way you look at it, the systems which have been discussed aren't just reducing the risk of people taking advantage of misplaced trust, but also giving a better picture of who blame can be placed on.

Another funny thing that I noticed as well is that instead of getting more and more specific with rules on how to create an infrastructure which insures correct identification, the solutions have shown themselves to became more and more abstract and flexible. The same unexpected solution has come up with the structure of the

infrastructure as well. Instead of the structure being more hierarchical or more web like, the solution tended to be a mix of both solutions.

In the end, if trust management systems become popular, we can hope to see more discussions in the future regarding security written in the language of trust management systems. With the boundaries now defined with this layer of abstraction, hopefully instead of fighting without knowing where the fronts are, we can finally look the enemy in the face and start to make improvements to the overall security of the Internet.

References

[1] J.B. Postel. "RFC821 - Simple Mail Transfer Protocol", August 1982

[2] J. Myers. "RFC2554 - SMTP Service Extension for Authentication", March 1999

[3] C. Ellison. "The Trust Shell Game". *LNCS 1550*: *Security Protocols 6th International Workshop, Cambridge, UK, April 15-17, 1998, Proceedings* (1998): 36-40.

[4] "How PGP Works". *http://www.pgpi.org/doc/pgpintro/.* Network Associates, Inc. and its Affiliated Companies.

[5] G. Caronni. "Walking the Web of Trust". Paper presented as part of the IEEE 9th International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, Los Alamitos, CA., USA, 14-16 June 2000.

[6] M. Blaze, J. Feigenbaum, J. Lacy. "Decentralized Trust Management". Paper presented as part of the 1996 IEEE Symposium on Security and Privacy, Oakland, CA., USA, 6-8 May 1996.