Samu Varjonen, Andrei Gurtov (2010):
Separating Friends from Spitters.
In Gunnar Stevens (Eds.),
International Reports on Socio-Informatics (IRSI),
Workshop Proceedings of 9th International Conference on the Design of Cooperative Systems - COOP 2010, Aix-en-Provence, France, May 2010
(Vol. 7, Iss. 1, pp. 141-150)

Separating Friends from Spitters

Samu Varjonen
Helsinki Institute for Information Technology, Espoo, Finland,
Andrei Gurtov
Helsinki Institute for Information Technology, Espoo, Finland
and CWC, University of Oulu, Finland
<firstname.lastname>@hiit.fi

Abstract. Undesired mail, such as commercials, is called spam in mail services. Spit is what spam is for mail services, unsolicited communications. The difference in spit and spam is that spam can be checked before the delivery to the recipient and spit can be reliably detected only after the call is made.
Voice over IP (VoIP) community has adopted a more peer-to-peer approach, in which the registrars and proxies are located on the participating nodes, rather than on separate servers. Industry has also been quite keen in the development of such approach, especially ones that use identifier-locator split protocols. The lack of centralized authorities and the usage of long trust paths make detecting spit even harder a task.
In this paper we describe a system to disseminate information of friendships that are based on an end-host based identifier-locator split protocol. In our solution the existing *buddy lists* are used to introduce one-hop connections in the system.
In our system we rather detect friends than spit or spitters. Moreover, our solution can be generalized for situations where the before-hand inspection of the content is impossible or otherwise hard to implement.

1 Introduction

Everyone knows how annoying it is to open a mailbox and see it littered with unsolicited mail. We have seen this problem grow in proportion over the years and now we see how it spreads across different mediums. Everything that draws in large crowds of users will eventually draw in hordes of spammers in a form or another.

VoIP is one of many new technologies that draw in users as well as spammers. Spam over Internet Telephony (SPIT), the equivalent to spam in VoIP, is more intrusive as a call effectively disrupts whatever the user was doing at the moment and the spam does not. Blacklisting is the most prominent way to fight against spam but it has its downsides and there is no sure way of knowing is the call SPIT before answering it.

In VoIP, Session Initiation Protocol (SIP) is the signalling protocol used to create the sessions between clients. In order to get a more scalable and less vulnerable SIP has Internet Engineering Task Force and networking industry been designing a pure peer-to-peer alternative that does not need any centralized servers, i.e. peer-to-peer SIP (P2PSIP). As the architecture moves away from the centralization, the research community has proposed the usage of trust paths in order to identify friends (Heikkilä 2009)

In our paper we argue that trust paths longer than one-hop are too long to retain trust. Moreover, we argue that hiding of the path structure from the search result, for privacy reasons, enables Sybil attacks without any fear of retaliation for the attackers.

Our solution is based on host based identifier-locator split with self-certifying cryptographic identities. These identities are used as the entities in the certificates that are used to communicate the one-hop paths between participants. Moreover, we show how this information can be used in the GUI to provide more information for the users to make better trust decisions.

It should be noted that the solution is also valid for other situations where the before-hand inspection of the content is impossible or inconvenient. First, the content such as live audio and video streams cannot be inspected before receiving it as it does not exist prior to the receival. Second, the inspection of a large file transfer can be impossible, difficult or even unwanted waste of resources as the file has to be transferred and stored on the inspecting host or middlebox.

Rest of this paper is structured as follows; In the Section 2 we describe the basics of P2PSIP, identifier-locator split protocols, and in more details a host-based approach, i.e. Host Identity Protocol (HIP). In the Section 3 we discuss about the usage of trust paths to identify friends amongst spitters. In Section 4 we describe the overall system including what is distributed and by what means. Also the flow of control is described in the system when a connection is made between participants unknown to each other. In Section 5 we evaluate the behaviour of the system, i.e. latencies of how long it takes to gather the one-hop trust path information, and how much space on the wire does the trust information take. Section 6 concludes the paper.

2 Background

SIP is a signalling protocol for conferencing, telephony, instant messaging and presence. SIP can create, modify and terminate two- or multi-party sessions. In SIP the user equipment (user agent, UA) is the network endpoint that creates or receives the calls. The actual architecture comprises of three elements: proxy servers, registrars, and redirect servers. Proxies handle the routing of SIP messages between UAs and they can implement access control. Registrars maintain the location information for the UAs, i.e. registrars translate SIP URIs into one or more IP addresses. Redirect servers can be used to redirect SIP session invitations to external domains.

For scalability and security reasons the SIP research community has introduced a peer-to-peer alternative for SIP called the P2PSIP (Jennings 2010). The infrastructure elements in SIP are defined as logical entities and are usually co-located on the same hardware. This distinction makes it easier to move the infrastructure elements to the UAs as P2PSIP does. The host Identity Protocol (HIP) (IETF RFC 4423, IETF RFC 5201, Gurtov 2008) has been proposed to be used for the connection maintenance and transport protocol for the P2PSIP because of its support for mobility, multihoming, NAT traversal, and security features (Keränen 2010). For VoIP applications NAT traversal is a major concern and by offloading it to HIP the connection management in VoIP applications becomes simpler. Moreover, by using HIP VoIP applications gain support for transparent mobility without any modification to the application.

It could be argued that why to use host identities in access controlling, as the modern operating systems are multi-user systems. First, in our opinion most of the machines, such as laptops, smart phones, etc. are more personal than ever and they even require additional authentication methods, such as PIN codes on smart phones, and account passwords on laptops, etc. Second, the usage of lower level identifiers to access control the connection avoids the need for deep packet inspection on higher layers. For example, using SIP URIs for access control would need inspection of SIP URIs in the SIP messages.

HIP introduces a new cryptographic namespace between the transport and the IP-layer. The namespace is based on public-key cryptography and consists of so-called Host Identities, which are RSA and DSA public keys. Using full-length public-keys in packet headers can produce too much overhead and would be incompatible with unmodified (legacy) applications. For this reason public-keys in HIP are also represented in a shorter 128-bit (IPv6-compatible) format, called the Host Identity Tag. HITs can be used directly with IPv6-enabled applications because of their size and format. Since HIP uses cryptographic keys as identifiers, host authentication and the establishment of a secure channel between HIP hosts is very simple. Moreover, HIP is designed to be extensible. A modular packet and parameter concept allows the addition of new functionality to HIP easily. In essence, the HIP Base EXchange (BEX) is a four-way handshake and key negotiation phase to create IPsec security associations between hosts HIP has an update procedure that is used to maintain the ongoing connection.

Digital certificates bind a piece of information to a public key by means of a digital signature, and thus, enable the holder of a private key to generate cryptographically verifiable statements. HIP has a container to transport X.509.v3 and SPKI certificates (Heer 2010). This container is an organizational parameter that can be grouped to transmit semantically grouped certificates in a systematic way.

HIP has support for the XML-RPC interface (Ahrenholz 2009). The XML-RPC interface provides basic features, such as *put*, *put-rm*, *rm* and *get* operations. Put operation inserts a key and a value to the storage and get retrieves the value, matching the key, from the storage. Remove operation is protected by a hash of a secret that is inserted with the value and revealed in plain text when removed. The XML-RPC uses also a time-to-live value in order to remove stale information from the storage.

HIP for Linux[1] implementation has an identity management GUI that filters all HIP based control traffic through it. Incoming connections are filtered upon receival of I1 control packets. Incoming connection prompts the user to make a decision on accepting or dropping the connection. The GUI is used also used to group known HIs in to groups and give them group based attributes. GUI can also be used to show dropped HIs.

3 Requirements for the trust paths

Web of trust (WOT) originates from the Pretty Good Privacy and is the starting point for most trust path related solutions. WOT is a decentralised trust model that represents the trust that the users have for each other. A path through the WOT from the initiator to the responder of the communication is presented as a token of trust to the responder. We argue that long trust paths are: complicated and do not represent real trust.

First, long trust paths increase complexity of the solutions and can have unforeseen problems. Heikkilä et al. allow long trust paths scheme in their Pathfinder (Heikkilä 2009) approach for protection against spit. Pathfinder uses one or more centralized servers as privacy protected search engines. The information that the pathfinder servers use is protected with a hash scheme. However, we argue that by protecting the privacy of the links the Pathfinder allows Sybil attacks.

A malicious user can set up a node that acts trustworthily among other nodes. While, acting nice, the node then links the real spitters to the WOT. Now, when

[1] http://www.infrahip.net/

the spitter tells the Pathfinder to search for a path from itself to the target, the path is found. What makes this effective is the fact that the malicious user connecting the spitters to the network does not have to fear of punishment as the path is hidden along the responsible node.

Second, long trust paths do not represent real trust. To be trustworthy, long trust paths require that every node on the path are honest and do not lie in their statements. The longer the path grows, the harder it becomes to trust every nodes decision on the path. In a country, such as Finland, it may be possible to connect the authors to the President with a reasonably short path. However, it does not say anything about the trust between the President and the authors in either way. In our opinion trust degrades quite fast, in the matter of few hops.

We propose based on the discussion above that one-hop trust paths are sufficient.

4 Our Solution

HIP BEX creates an IPsec tunnel between the participating hosts and the traffic is transported inside the tunnel between the hosts. Due to the nature of IPsec being a host-to-host connection, we access control the traffic coming from the tunnel by using the HITs from the packet headers. This way we bind the tunnel for certain usage, VoIP in our case.

In our prototype we used OpenLookup v2 as our credential storage system. Although, any storage that implements anonymous key-value storage service with the XML-RPC interface can be used, such as Bamboo DHT[1] or OpenLookup v1[2]. The replacement does not have to be a DHT but could be a centralized system or even less centralized system than DHT. But it should be noted that by using a centralized system it may become bottleneck for the system and in less centralized systems the revocation of the certificates may become overly complicated.

Upon an incoming connection from an unknown host the user is prompted with a dialogue, in which a question is asked, whether to accept or drop the new connection. If the connection is accepted, a certificate is created and uploaded into the storage system. This certificates semantic statement is that the subject has trusted the host enough to accept a connection from the host. Upon subsequent connections from the subject, the certificate's can be checked and the user is not bothered with the dialog.

The certificate contains the HITs of the participants as the issuer and subject. Moskowitz et al. (IETF RFC 5201) say that HIT collision maybe possible, while improbable. For this reason the certificates contain also the full HIs. The certificate contains also a short time frame in which the certificate is valid. This

[1] http://bamboo-dht.org/

[2] http://code.google.com/appengine/

makes the revocation easier, as the issuer can just stop renewing and uploading the certificate. For the host the information contained in the certificate is enough but in the *new HIT* dialogue of HIPL the HITs are also presented to the user. Long HEX strings are hard for the user to recognize and for this reason we added an issuer given name into the certificate. Moreover, we believe that because the names are given by first-hop friends they most probably have a meaning for the receiver. As the certificates do not contain any location information they are also suitable for mobile clients as there is no need to update the certificates in the system upon mobility events.

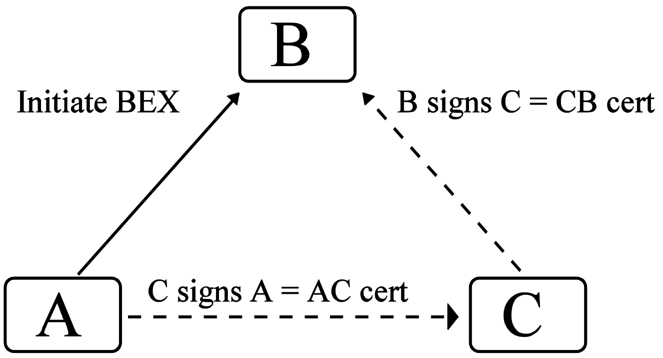
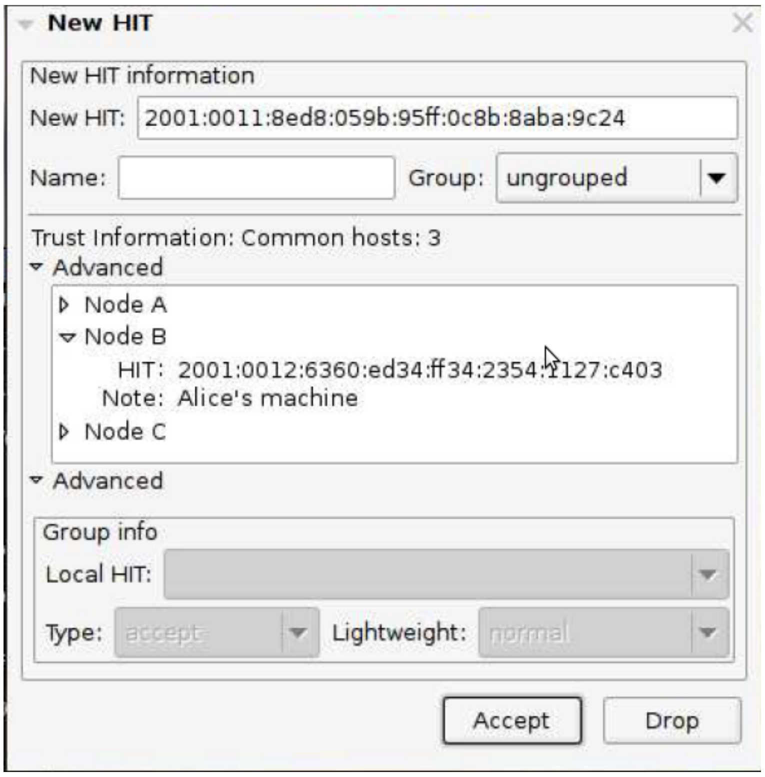


Figure 1 Forming of a trust relation ship between host A and host B by presenting the certificate given to host C to host A.

In our system malicious users can try to lie about their trust but the lying would be noticed easily and the lying friend could be punished. Moreover, the system could have an additional rating (e.g. a floating point value from 0 to 1, where 1 is complete trust) for the trust that could be increased or decreased based on the observed behaviour. In practice this rating could be enforced by increasing the puzzle sizes and/or by throttling the connection by limiting the bandwidth of host with low ratings and vice versa for hosts with high ratings. In the end, the meaning of the rating is left as a local policy, this way the hosts can make independent choices on the level of enforcement. In the worst case the host could deny all connections from the subject and remove its certificate from the system. By tying the used puzzle size in the BEX to the used trust rating, the responder could also choose the puzzle sizes for the initiators based on their ratings. The used computational cycles to solve the puzzle would constitute a payment for the service needed by the initiator of the connection.

During our implementation efforts we changed the triggering point for the access control of incoming HIP control packets. Previously the trigger point was in the receiving of I1 control packets and we moved the trigger point after the handling of I2 control packet. At this stage the user would not see prompts for all easily forged I1 control packets. Instead the responder will see the incoming connection prompt only after the initiator has solved the puzzle successfully. Moreover, the signature in the I2 control packet has proofed that the initiator actually owns the corresponding private key, from which the used HIT was created.


New HIT

New HIT information

New HIT: 2001:0011:8ed8:059b:95ff:0c8b:8aba:9c24

Name: Group: ungrouped

Trust Information: Common hosts: 3

Advanced

- Node A
- Node B
  - HIT: 2001:0012:6360:ed34:ff34:2354:1127:c403
  - Note: Alice's machine
- Node C

Advanced

Group info

Local HIT:

Type: accept Lightweight: normal

Figure 2 Dialogue presented upon an incoming connection with our modifications.

In the initialization phase, i.e. when the host starts the identity management GUI, the hosts upload the certificates to the used storage service. The upload can happen sequentially or in parallel but for our purposes uploading sequentially was adequate. In the example, host B has previously accepted a connection from the host C and uploaded the certificate to the storage system (CB cert in the Fig 1.). Moreover, host C has previously accepted a connection from host A and has uploaded the certificate to the storage system (AC cert in the Fig 1.) When the initiator starts the connection it queries all the possible concatenations of its own HIT used in I1 and I2 control packets as the destination HIT and its friends HITs and tries to find one or more suitable certificates to be presented to the responder in the BEX. The certificates are stored by using the hash of the concatenation of the issuer and subject HITs from the certificate as the key. This way we retain some privacy as the key cannot be directly guessed. If only the issuers HIT was used the malicious host could easily gather the friend list of a host. Using the hash of the concatenation of HITs the key is obfuscated so the malicious user has to guess what the HITs of the friends are and while the malicious user would guess one friend it would not reveal other friends of the host.

In our example, the host A finds two certificates: one given by host B for host C, and one given by host C for host A. Host A transports the certificate AC in the BEX to the host B. There is no need to transport the certificate CB as the host C is already in the friend list of host B and is trusted.

Upon receiving the I2 control packet host B verifies its signature and checks that the puzzle is solved correctly. If these checks were performed successfully the identity management GUI prompts the user asking to accept or drop the connection (see Fig. 2.). In this prompt, in addition to the used HITs, the user is presented with the user friendly name of host A given to it by host C.

5 Evaluation

We measured the mean latency of OpenLookup v2, using both IPv4 and IPv6, to determine the query performance of the system. We used a quad core Intel Xeon 5130 running at 2 GHz with 2 GB of main memory as the server and we used a laptop with Intel Core 2, 2 GHz CPU processor with 2 GB of main memory as the client. All machines involved in our measurements were located in our local Gigabit network with a mean round-trip latency of 0.88 ms (std.dev. 0.03 ms). We concentrated more on the query performance and not on the update performance, since the friendships seldom change and the frequency to refresh the credentials in the storage can be even a week. The measurements were done by querying random keys with values containing certificates of varying sizes.

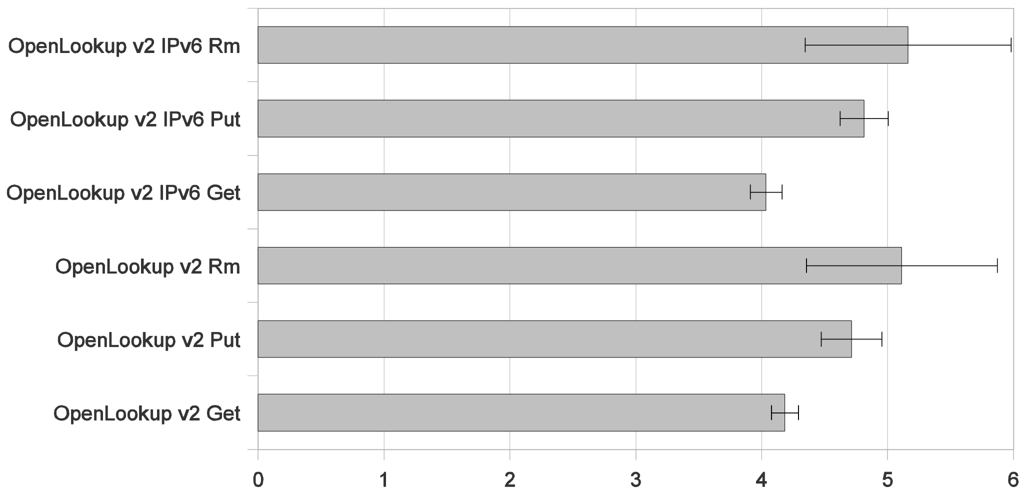


Figure 3 Average latencies measured from the storage systems in milliseconds.

From the results depicted in the Figure 3 it can be easily calculated that an initiator, with 100 friends, can sequentially query the system in maximum of circa 418 milliseconds. In the Internet the RTT times between the client and the server increase the latency. With RTT of, for example, 70 milliseconds between the client and the server will the total time be circa 7418 milliseconds. In our opinion even the longer latencies are acceptable because it bothers only the initiator of the connection. Moreover, the query performance can be optimized by querying in parallel and by caching the results locally on the client and thus avoiding subsequent queries of credentials.

In our experiments we noticed that certificates can pose a size problem for the control packets. If multiple suitable certificates are found, we could send multiple certificates in the control packets to the responder so that the information could be prompted to the user. However, the average size of a I2 control packet, using 1024 bit RSA keys, is circa 850 bytes and it occupies most of the minimum MTU of IPv6 (1024 bytes) and exceeds the minimum MTU of IPv4 (512 bytes). When we add one or more certificates to the I2 control packet will exceed even the IPv6 minimum MTU. This makes it very probable that the packets are fragmented on the wire. The solution for the size problem is left for further study.

6 Conclusions

In this paper we presented a discussion on how to separate friends from spitters, using SIP as an example. Based on our observations, the discussion identifies two problems in the proposed trust path solutions: a) trust path solutions that hide the path details from the users allows Sybil attacks, b) real life trust does not extend over multiple hops.

We addressed these problems by using the self-certifying cryptographic identities of Host Identity Protocols (HIP) to create one-hop trust paths to be used to access control the incoming calls. Our solution is in a sense a distributed white list based on host identifiers that identify the incoming connections from friends. We also provided measurements from live storage systems in order to estimate the time required to gather sufficient trust information and discussed about the size issue caused by the addition of certificates to the control packets.

7 References

Heikkilä, J. and Gurtov, A. (2009): Filtering SPAM in P2PSIP communities with web of trust, in Proceedings of the MobiSec'09, Jun 2009.

Jennings, C., Lowekamp B., Rescorla, E., Baset, S., and Schulzrinne, H. (2010): Resource location and discovery (reload) base protocol: draft-ietfp2psip-base-08, Mar. 2010, work in progress.

Moskowitz, R. and Nikander, P. (2006): RFC 4423: Host Identity Protocol (HIP) Architecture," Apr. 2006.

Moskowitz, R., Nikander, P., Jokela, P., and Henderson, T. R (2008): RFC 5201: Host Identity Protocol, Apr. 2008.

Gurtov, A. (2008): Host Identity Protocol (HIP): Towards the Secure Mobile Internet. Wiley and Sons, 2008.

Keränen, A., Camarillo, G., and Mäenpää, J. (2010): Host identity protocol-based overlay networking environment (hip bone) instance specification for resource location and discovery (reload): draft-ietf-hip-reload-instance-01.txt, Mar. 2010, work in progress.

Heer, T. and Varjonen, S. (2010): HIP certificates: draft-ietf-hip-cert-03, Apr. 2010, work in progress.

Ahrenholz, J. (2009): HIP DHT interface: draft-ahrenholz-hiprg-dht-06, Nov. 2009, work in progress.