HIPernet: A Decentralized Security Infrastructure for Large Scale Grid Environments

Julien Laganier * †, Pascale Vicat-Blanc Primet †
* DoCoMo Communications Laboratories Europe
† Laboratoire de l'Informatique du Parallelisme (LIP)
laganier@docomolab-euro.com, pascale.primet@ens-lyon.fr

Abstract

Security in Grid environments appeals for fundamental primitives like the secure establishment of dynamic and isolated virtual trust domains. The security mechanisms currently used are generally based on a Public Key Infrastructure global to the grid environment, and a mix of global and local access control policies used to make an authorization decision. Such approaches do not scale well with the number of participating domains and entities. In this paper we propose a decentralized approach for securing grid environments that better cope with their inherently distributed nature. The combination of network and operating system virtualization (Supernets) with the Host Identity Protocol (HIP) and Simple Public Key Infrastructure (SPKI) delegation/authorization certificates allows to create virtual trust domains onto multiple shared computer nodes connected by an untrusted network. We analyse how this approach adapts the vast diversity of trust relationships in the real world and has a better scalability with respect to the number of entities involved.

Keywords: Grid Security, Authorization, Delegation, Resource isolation, Host Identity Protocol.

I. Introduction

The generic problem solved by the grid computing concept is coordinated resource sharing and problem solving in dynamic, multi-institutional, virtual organization [1]. One major requirement is to control precisely how shared resources are used. Over the past years, the Grid community has proposed security solutions that support management of credentials and policies when computations span multiple institutions, and resource management protocols and services that support secure remote access to computing and data resources. The security solutions for Grid and grid services [2] are generally a combination of a global PKI for the grid environment and per-resource *ACL* (Access Control List): the resource would perform *authentication* of the user based on its PKI certificate, and make the *authorization* decision based on the resource *ACL* and the user identity (e.g. a login name or a Distinguished Name). But this kind of solution has a major drawback in terms of scalability with respect to the number of participating domains. In particular, the burden require to reconfigure the grid environment (e.g. addition or removal of a Virtual Organization (VO), a resource, a user, an organization) is still relatively high compared to the promises of ubiquitous and dynamic resource sharing, as envisioned in grid and peer-to-peer communities.

Taking into account the recent innovations in system and TCP/IP technologies, we explore a decentralized and distributed approach to the security in multi-domain grid environment. The basic argument behind the model is: although it might be legitimate to require user authentication at a local site, *authentication* at the shared resource is not required per se to make an *authorization* decision. Our approach reuse and combine existing technologies which provide the security building blocks of a fully distributed security architecture: network virtualization, operating system virtualization and delegation/authorization certificates. The paper is organized as follows: section II presents the foundations on which our work is laid. Then section III describes the HIPernet design model and discuss how it fits the security requirements. Section IV relates experiences with the implementation of HIPernets. Finally, an overview of related works is given in section V, before we conclude in section VI.

Note: Part of this work was done while Julien Laganier was employed by Sun Microstems Laboratories and doing a PhD Thesis at the LIP.

II. Foundations

A. Authentication vs. Authorization

Sharing distributed resources amongst a multi-domain grid environment raises complex security and policy issues, at the heart of which lies the ability to make an *authorization* decision when a shared resource is accessed.

When looking at authorization issues involved with the implementation of network and distributed systems security, there is quite a lot of examples of such issues being solved by following an *authorization*-centric approach [3] rather than following a one-size-fits-all *authentication*-centric approach. All these solutions are architected around a distributed security infrastructure model, like the Simple Public Key Infrastructure (SPKI) [4] or the Simple Distributed Security Infrastructure (SDSI). When an entity make a resource accessible to users, it is rather interested in verifying that an entity (e.g. user, group) accessing this resource is authorized to do so than in knowing what its name is. Moreover, the knowledge of the entity name is, in general, insufficient to make the authorization decision. It is often required to look up in an Access Control List (ACL) defining *who* is authorized to do *what*. The ACL is indeed binding a *name* to specific *rights* (a.k.a. *authorization*, *capability* or *permission*.)

For instance, in a distributed firewall implementation [3], the firewall needs to know whether or not a particular packet is authorized to pass through. If the authorization decision process relies on authentication, obviously, knowing the name (e.g. John Smith) of the sender of a packet is not sufficient to make the authorization decision. The firewall must also be configured to look up in an Access Control List (ACL) defining *who* (e.g. John Smith) is allowed to do *what* (e.g. send through the firewall TCP packets with destination port number 80).

A major drawback of *authentication*-centric approaches in Grids is that they often require a global Public Key Infrastructure, which in turn requires the entities of the grid to trust all the Certificate Authorities (CA) of the security domains composing it. Hence, if one of the domain is compromised, the entire grid can possibly be compromised, not only those in relation with the compromised node. The flexibility of those grids is also affected by the fact that adding or removing a domain from the grid incur a huge administrative overhead, and often formal face-to-face meetings. In other words while the size multi-domain grid environment is increasing, its robustness, flexibility and overall security are diminishing, thus severely limiting the scalability of these grid environments.

In the authorization approach, the only entity name required is the public key. The real name (e.g. John Smith, Foo Corporation) is not used to make the decision when an entity tries to access a resource. Hence, when the entity initiate a transaction towards the resource, it sends its public key and signs the message, along with a certificates chain. This certificates chain is made of multiple ordered SPKI [4] Authorization Certificates, each of them issued by a public key (the issuer) to another public key (the subject) and granting specific rights (e.g. file permission mask `rwxr-xr-x`, privilege `operator.reboot.*`, etc.). For the authorization to be granted by the resource controller, such a certificates chain must start from the public key of the resource controller, end with the public key of the entity accessing the resource, with the subject and issuer of each intermediate certificate being, respectively, the issuer of the next certificate of the chain, or the subject of the previous certificate of the chain. The intersection of rights granted by certificates of the chain must be a superset of the rights required per the access policy of the resource: If "entity X delegates to entity Y the right R" is denoted by the notation $X \overrightarrow{R} Y$, then $A \overrightarrow{S} B$ and $B \overrightarrow{T} C$ implies that $A \overrightarrow{S \cap T} C$, i.e., entity A delegates to entity C the intersection of rights S and T. For instance, if $A \xrightarrow{rwxr-xr-x} B$ and $B \xrightarrow{r-xr-x---} C$, then $A \xrightarrow{r-xr-x---} C$.

B. Secure Communication Channels

A grid environment provides the same three fundamental functionnalities of a computer: computation, storage and communication. The communication channels constitute the backbone of the grid: geographically distributed computation and storage units can collaborate only if they are able to communicate between each others. Consequently, and because of some basic security principles which we will discuss later, secure communication channels ought to be the foundation onto which a secure grid can be built. Moreover, the availability of those secure communications channels should be available in an environment where:

- Relationships between grid entities (e.g. users, services, resources, organization, etc.) is dynamic, and possibly short-lived.
- The network interconnect might grow, diminish, move, etc. It is not a fixed entity w.r.t. location and constitution.
- The entities use the communication infrastructure transparently: end-use, applications, tools and APIs behave like if they were using a regular TCP/IP Internet or intranet.

C. Domain Virtualization and Isolation

Virtualization has proved to be a very powerful mechanism to solve various security and administration problems involved with sharing a single computing or network platform between multiple entities or administrative domains.

OS virtualization allows to securely partition, manage and isolate separate virtual views of the underlying operating system while network virtualization allows to build a private network within a public network infrastructure.

Network virtualization exists both at layer 2 (the link layer) and 3 (the network layer) of the OSI reference model. IEEE 802.1Q Virtual Local Area Network (VLAN) is an example of layer 2 virtualization: Multiple logical VLAN instances can cohabit on a single physical LAN. IPsec Virtual Private Network (VPN) is an example of layer 3 virtualization: an IPsec tunnel gives the illusion of a secure point-to-point link while packets are actually traveling through a multi-hop and insecure path. Both solutions also provides access controls features: A VLAN-capable switch port or network interface can be configured to process (i.e. receive, send, forward) only frames tagged with particulars VLAN identifiers, therefore keeping different VLANs isolated from each others. An IPsec stack has a Security Policy Database (SPD) which specifies the access control policy of the node; this is similar to a routing table, or firewall table, but rule entries are typically indexed by a 5-tuple (source address, source port, destination address, destination port, protocol number), and specifies which action a packet should be subject to: further IPsec processing (e.g. encryption and/or integrity protection), bypass, or drop.

Traditionally, VPN solutions have addressed security of point-to-point communications. These communications were typically taking place between two distant locations belonging to the same administrative domain (e.g. a corporation) like, for instance, a VPN linking security gateways of two distant intranets, or VPN linking a mobile road-warrior to its corporate intranet security gateway.

If the communications to be secured occurs between many distinct entities belonging to many administrative domains, then point-to-point VPN model doesn't offer a convenient solution because the number of tunnels to manage is the square of the number of entities. Hence new technologies have been designed at layer 2 and 3 to allow management and deployment of so-called Provider-Provisioned VPN (PPVPN), in which a service provider is responsible to centrally manage and deploy between multiple remote sites a mesh overlay networks whose links are point-to-points VPN tunnels.

However, management and deployment of VPN overlays between very dynamic coalition of nodes (like grids) can be better tackled by the communicating end-points themselves (as opposed to trusted third parties) because they have a better view of what the user and applications communications needs are. The Host Identity Protocol is the key-enabler of end-to-end VPN overlays.

D. Building IPsec VPN Overlays with the Host Identity Protocol

In the traditional TCP/IP protocol stack, the IP address play two independent roles: The *locator* and the *identifier* role. Network Layer Protocols (NLPs, e.g. IPv4, IPv6) use the *locator* role of IP addresses to route packets, while the Upper Layer Protocols (ULPs, e.g., TCP, UDP) use the *identifier* role of IP addresses to name end-points (e.g. sockets). Because of this deliberate confusion between these different roles, ULPs are dependent on location, and broke when network mobility and multi-homing cause a modification of the IP address.

The Host Identity Protocol (HIP) [5] proposed by the IETF decouples these two roles while maintaining a binding between *identifiers* and set of associated *locators*. This is a virtualization of the network infrastructure from an upper layer or application standpoint. The *identifier* namespace defined by HIP is made of the public key of the host, and is called a *Host Identity*. Because sometimes an identifier needs to be embedded in a fixed size field of an existing protocol or API, the HIP specification also defines the *Host Identity Tag (HIT)*, a 128 bits long truncated hash of the public key. A typical application would no longer use directly IP addresses as end-point identifiers, but rather the Host Identity (HI) or the Host Identity Tag (HIT). The HIP layer is in charge of mapping HIs and HITs into appropriate locator IP address for the node.

The HITs belongs to the Crypto-Based Identifiers (CBIDs) family, which was used to help solve several issues in the IPv6 world: identifier ownership for mobility, neighbor discovery, and multicast group membership as well as infrastructure-less opportunistic encryption [6]. The next section presents the HIPernet model based on these four foundations.

III. HIPernet Design Model

A. Goals and Principles

Ideally, any user of a Grid should have the illusion that he is using its own system, while in reality it is using multiple systems part of the Grid. To highly control this sharing we propose to combine the virtualization of both the network and the operating system. The OS is virtualized to permit multiple virtual nodes to cohabit on the same physical host, and the network is virtualized to permit multiple virtual overlay networks to cohabit on a shared communication infrastructure. The resulting virtual network and OS instances are kept isolated from each others.

The HIPernet model is an implementation of the Supernet security model [7]. and is based on the HIP and

SPKI architectures. The members of a HIPernet have a consistent view of a single private TCP/IP VLAN overlay, independently from the underlying physical topology. The figure 1 shows how topology-independent HIPernet VPN overlays are built across the Internet. The HIPernet can span multiple networks belonging to disparate administrative domains. A user can join from any location, and uses the same TCP/IP applications he was using on the internet or its intranet. The HIPernet is transparent to upper layers in the large sense of the term: Upper Layer Protocols (e.g. TCP, UDP), APIs (e.g. sockets), middleware (e.g. DCE, Globus), applications, services and users. Hence, the HIPernet model maintains backward compatibility with existing APIs, Middlewares and Applications which were designed for UNIX and TCP/IP APIs. Therefore, users of the Grid do not need to learn new tools, developers do not need to port applications, legacy user authentication can still be used to enroll an user into a HIPernet, and a middleware laid on the HIPernet secure communication paradigm can securely provide the remaining services constituting the grid: Secure Storage and Secure Computation.

However, the transparency feature does not prevent some of these upper layers to interact explicitly with the new security services through new APIs, like the HIP native API. These interactions includes: setting an application or user-specific Host Identity, tune the cryptographic parameters, etc.

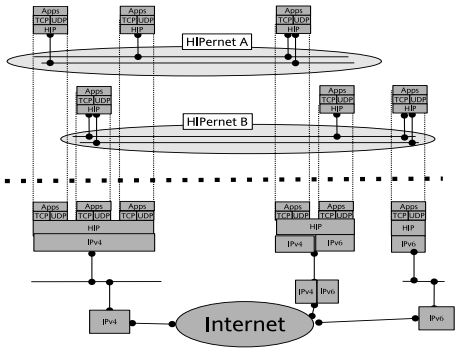

Fig. 1. HIPernets laid across the Internet

The HIPernet design is backed by some of the seminal security principles of J.H. Saltzer [8].

The *economy of mechanism* principle translates into a recombination and integration of existing architectures and protocols: The essence of the HIPernet architecture lies in the Supernet architecture [7]. It is is implemented on top of the `zone(5)` [9] and `jail(8)` [10] frameworks of modern UNIX-like OS. The network security architecture is based on the the Host Identity Protocol architecture [5], and delegation certificates from the SPKI Certificate Theory [4]. The *fail-safe defaults* principle translates into an access decision process which requires explicit authorization via a combination of local security policy and entity-to-entity delegation certificates chains. The *separation of privilege* and *least common mechanism* principles translate into the virtual resource instance assignment model: Each trust domain is assigned its own virtual OS instance and its own virtual communication channels. These virtual instances are kept isolated from each others (both at the network and OS level), and relationship is constrained per the access control security policy and delegation certificates chains. The *least privilege* principle translates into delegate certificates issuing to entities the least set of privileges they need to complete their jobs. The *psychological acceptability* principle translates into the preservation of the APIs in use today (e.g. UNIX and TCP/IP sockets). The protection mechanisms are therefore applied transparently to applications, tools, and end-users, maximizing their acceptability.

B. Definitions

An HIPernet is a collection of HIPernet nodes linked together by HIPernet channels. An HIPernet channel constitute a trust domain in which services (communication, computation, storage) can be securely offered to *authorized* consumers (i.e. channel members). An HIPernet is formed of many channels hosting services, which collaborates to accomplish their duties over the Internet in a secure manner.

a) HIPernet Entity:: The fundamental unit of trust domain involved within a HIPernet. It can be a user, a group of users, an application, a service, an organization, a computer node, etc. It is uniquely identified by a public key (HI) or its compact representation (HIT), and can delegate, or be delegated, specific access rights via SPKI authorization certificates.

b) HIPernet Node:: The smallest security container for a *communicating execution* environment. This is typically a `zone` (or `jail`) instance running into a HIP-enabled UNIX node, and attached to one or more HIPernet Channels.

c) HIPernet Channel:: A security and trust domain defining a set of HIPernet nodes which can communicate together. This is the smallest security container to isolate security perimeter in a HIPernet.

d) HIPernet Registration Service:: A service allowing HIPernet Nodes to join and leave a HIPernet Channel. Each HIPernet Channel relies on this service to issue membership authorization certificates to the nodes willing to attach to it. Once authenticated (by a new mechanism, or a legacy mechanism, e.g. remote login and `/etc/passwd`, LDAP, Kerberos, etc.) and registered with a HIPernet Registrar, the entity is authorized to access the HIPernet

by receiving capabilities to use it in a manner which is consistent with its policies. A HIPernet node might leave, time-out or be banned from a HIPernet. There might be a revocation/revalidation of a node channel membership via an online certificate revocation/revalidation check, or the authorization certificate might just not be renewed.

C. Entity Identification and Rights Delegation

Each grid entity is uniquely identified by a HIT. Each application, service, user, group of user, or (virtual) organization might be an entity itself, or belong to a larger entity, and is thus similarly identified by a HIT. The HIT is therefore the primary name for an entity and is used for communication endpoint (e.g. sockets) naming, entity-to-entity right delegation (SPKI authorization certificates), as well as access control (HIPernet Security Policy Database).

An authorization (or delegation) certificate assert a set of authorization rather than a name. The authorization is issued by the the holder of the issuer public key (`issuer`) to the holder of the subject public key (`subject`). The most renown authorization certificates scheme is SPKI [4], and stands for Simple Public Key Infrastructure.

e) : SPKI [4] specifies a framework of authorization certificates that allows an entity (`issuer`) to authorize another entity (`subject`) to perform some actions through the delegation of rights. SPKI certificates differs from usual certificates by the fact that both the `issuer` and the `subject` are identified by either their public key or a hash of it (As opposed to usual certificates in which the issuer is identified by its Distinguished Name). A `subject` may be authorized through a chain of several delegable SPKI certificates, each of them having a `subject` field corresponding to the `issuer` of the next certificate in the chain. The final subject gains the intersection of the rights granted by each certificate of the chain. These certificates chains allows to better adapt the HIPernet trust model to the vast diversity of trust relationships in the real world.

f) : SPKI certificates are particularly appealing to secure distributed systems involving many entities and administrative domains because they allow entities to assert each others with proof of authorization, without on-line intervention of any trusted third party. The delegation feature allows to express trust relationships in many different manners, by allowing a mix of topologies like hierarchical tree, small flat groups and Web-of-Trust. The delegation property is particularly convenient because it allows to avoid centralized credentials management, bottlenecks in authorization enforcement process, and single point of failure, thus alleviating scalability issues.

A SPKI certificate has the following general structure:

```
(sequence
(public-key object)
(cert object)
(signature object)
)
```

public-key and *signature* are objects defined by the SPKI framework, while the *cert* object is application-dependent.

g) : In accordance with the SPKI theory, we define a HIPernet authorization certificates for HIPernet entities. Each user, group, application or service, real or virtual organization, or even resource has a HIT, and its associated public-private key pair. The HIT defines a trust domain, and can issue, or be issued, HIPernet authorization certificates.

```
(cert
   (issuer (hit <hit>) )
   (subject (hit <hit>) )
   (tag <capability-name_1> (arg <arg_1>)
                              ...
                            (arg <arg_i>) )
   (tag <capability-name_2> (arg <arg_1>)
                              ...
                            (arg <arg_j>) )
   (propagate)
   (online <online-type> <uris>)
   (not-before <date1>)
   (not-after <date2>)
)
```

The authorization certificate we use includes a `tag` which defines several capability granted, as well as multiple arguments (`arg` field) used to limit or tune the usage of the capability. The signer can optionally include an Uniform Resource Identifier (URI) indicating the location of an online certificate revocation/revalidation check server (`online` field).

We propose to authorize further HIP-enabled communication between two entities if and only if both entities have transmitted in-line an SPKI certificate chain authorizing each of them to access a common channel. Such a chain must begin with a SPKI certificate issued by the channel HIT (or a delegated entity HIT, which is trusted by the channel to issue channel membership certificates in a manner consistent with the channel policy). If this certificate is not delegable (no `propagate` flag), then the `subject` must be the entity asserting channel membership. On the other hand, if the certificate can be delegated (presence of the `propagate` flag), then it can be followed by a chain of other certificates which have to be delegable as well, except the latest. Each certificate `subject` field of the chain must be the `issuer` of the next certificate of the chain. The latest certificate `subject` must be the entity asserting channel membership. The intersection of the rights granted by each certificate of the chain must include the rights required to access the channel, as specified per the local security policy (i.e. `tag channel` for the appropriate `channel`). These certificate chains allow the

maximum flexibility in the representation of the effective trust relationship among a large number of individuals and others entities, in a manner independent from the actual placement and topology of the entities and resources, and do not require updates when entities change their location.

After validation of the credentials, the entities configures two unidirectional end-to-end IPsec Security Associations to protect further data flows.

Each HIPernet entity should be provisioned with the Authorization Certificates chains it needs to perform its duties. For example, if a HIPernet node is attached to a HIPernet channel, then the channel HIT may issue to the node HIT a certificate granting the `hipernet.channel.register` privilege:

```
(cert
   (issuer (hit 43fe:0fec:a120:2c48:de93))
   (subject (hit 4c48:54ff:1ae3:01bb:0ab4))
   (tag channel (hit 43fe:0fec:a120:2c48:de93))
   (not-before 4/25/2005)
   (not-after 4/26/2005)
)
```

And this is an example of a certificate which permits a HIPernet registration server to act as a delegated registration server for the same channel (note the `propagate` field which enables further delegation and the `online` field allowing to check the certificates revocation/revalidation status):

```
(cert
   (issuer (hit 43fe:0fec:a120:2c48:de93))
   (subject (hit 4ae3:01bb:0ab4:b89a:4f0e))
   (tag channel (hit 43fe:0fec:a120:2c48:de93))
   (propagate)
   (online crl http://www.example.com/crl/latest)
   (not-before 1/1/2004)
   (not-after 12/31/2009)
)
```

The above certificate would allow the delegated registration server to delegate further the rights to attach to this channel: It would form with the preceding and following certificate a valid chain for `tag channel`:

```
(cert
   (issuer (hit 4ae3:01bb:0ab4:b89a:4f0e))
   (subject (hit 4fb9:56d4:9ab4:6f0a:a3b5))
   (tag channel (hit 43fe:0fec:a120:2c48:de93))
   (not-before 4/25/2005)
   (not-after 4/26/2005)
)
```

IV. Experiences and implementation consideration

The implementation of the HIPernet architecture on a modern UNIX-like operating system like FreeBSD proved to be straightforward. We choose to leverage on the vanilla IPsec stack and OS virtualization mechanisms (`jails(9)`) implemented in the native OS. In addition to these existing mechanisms, we implemented a HIPernet agent, implemented as a user space daemon. This agent is in charge of the management of the HIP and the HIPernet layer. The HIPernet layer maintains two databases: the HIPernet channel registration policies database and the HIPernet channel registration database. The HIP layer also maintains two databases, the HIP policy database and the HIP association database, and is in charge of key exchange, name (i.e. HIT) binding, and readdressing (i.e. mobility and multi-homing) management. The IPsec layer maintains the IPsec security policy database (SPD) and security association database (SAD). The OS virtualization tools and IPsec stack are in charge of the enforcement of the above described security policies and associations. This simple and modular design is implemented as a user-space daemon comprising the HIPernet and HIP databases, as well as the base exchange and registration protocol, which communicates with the in-kernel IPsec stack via the `PF_KEYv2` message-oriented API. The agent source code consists of less than than ten thousand lines of code.

h) : The HIPernet agent acts upon triggers (`SADB_ACQUIRE` messages sent through a `PF_KEYv2 socket`) sent by the IPsec stack when an application needs to send data, but no appropriate IPsec security association is found to encrypt and/or authenticate the corresponding data packets.

The trigger contains the source and destination HIT of the data packet. The agent then looks into the HIP security policy database to find the matching policy and see if a functional HIP association exists between these two HITs. If it found a HIP and HIPernet association but no IPsec state exists, the two unidirectional IPsec security associations are re-established. If it did not find a HIP association, then the agent look into the HIPernet channel security policy database and channel registration association database. If an appropriate HIPernet channel registration exists for the channel (in the form of a verified SPKI HIPernet certificate), then the node tries to re-establish a HIP association. If no appropriate HIPernet channel registration exists for the channel, the node tries to establish a HIP association, while it sends its channel certificate and verify the one send by its remote peer. Then, a HIPernet registration entry is created in the HIPernet channel registration database, and the HIP security policy matching the source and destination HITs in put in place in the HIP SPD, while the corresponding HIP association entry is populated into the HIP association database. The same treatment applies to the IPsec SPD and SAD, which are populated with security policy and association entries matching the source and destination HIT of the original data packet.

The data packet can then flow further down the IPsec stack and be subject to encryption and data integrity protection.

Because the architecture preserves the end-to-end properties of the original TCP/IP, and the UNIX and socket APIs, we were able to run unmodified basic UNIX applications. For example, two HIPernet nodes connected together by a HIPernet channel have communicated with unmodified `ping` and `telnet` using secure identifiers in the legacy `socket` API. Inside this channel, the nodes are isolated from the outside, and `ping`, `telnet` or `rsh` are secured.

i) : Using this architecture, we were able to create multiple virtual HIPernet nodes on a single physical node hosting a single operating system image (figure 2). These virtual HIPernet nodes can be securely allocated on-demand to entities which needs remote access or computation. And finally, an existing middleware (e.g. [11]) hosted by HIPernet channels and nodes can provide the remaining services of the grid (e.g. storage).

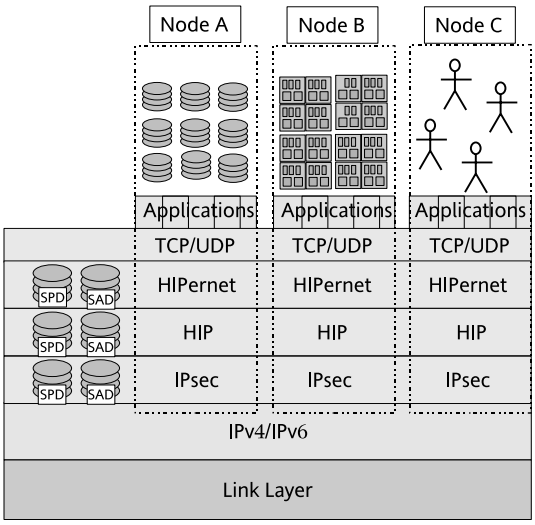

Fig. 2. Consolidation of HIPernet Nodes

The corresponding HIPernet channels security policy entries, as well as channel and node identification were assigned for once, despite the fact that some nodes changed their IP address.

V. Related Work

Examples of existing middleware security layer which rely on a grid-wide Public Key Infrastructure are the Grid Security Infrastructure (GSI) [2] and SiRIUS. They are both laid on top of existing mechanisms and provide their own new API, and require an always on-line PKI to work. GSI tries to federates existing security mechanisms such as Kerberos or PKI under a global Grid PKI, while SiRIUS introduces a new security layer on top of existing untrusted storage services (e.g. NFS, CIFS, P2P, HTTP).

j) : Our proposal differs from GSI from an implementation point of view, although the architectures are very similar from a functional point of view. Functionally, both GSI and HIPernet allows for the dynamic deployment of secure virtual organization (or trust domain) overlays, while handling entity identification and rights delegation. When it comes to implementation, GSI is based on transport layer security (TLS, layer 4) and PKIX/X509 certificates, as opposed to HIPernet which leverage on network (and sub-transport) layer security (IPsec, layer 3 and HIP, layer 3.5), self-certifying naming (i.e. CBID/HIT) and SPKI delegation certificates. We believe our approach allows for the maximum of generality, because its "lowest layer" implementation allows it to constitute a least common denominator to deployed grid security infrastructure. Nevertheless, the HIPernet architecture can also act as an additional security barrier complementing existing security mechanisms (e.g. GSI).

k) : The Supernet architecture [7] was an attempt to provide secure communication services for the Public Utility Computing Environment, a concept which encompasses grid computing and its usage scenarios. We similarly propose to run existing applications on a collection of UNIX and TCP/IP nodes while enforcing a pre-established security policy, which is independent of the low-level configuration details of the multi-domain grid environment (i.e. IP numbering, mobile agent placement).

Systems providing UNIX and TCP/IP APIs on top of a multi-domain grid environment like MOSIX [12] are already implemented and running, but they do not provide strong security: Most of them relies on firewalling and closed private network for their security. Hence, the compromission of one of the nodes of the systems often causes compromission of the whole system.

VI. Conclusion and Future Works

In this paper, we have presented a solution to secure distributed computing platform infrastructures and their services. The HIPernet architecture backed by the Host Identity Protocol and SPKI authorization certificates provides location-independent security.

We show how to combine together existing security building blocks (HIP, operating system sand-boxes and authorization certificates) for enforcing a pre-established security policy in a large-scale multi-domain grid environment. We have implemented this model for UNIX-like operating systems (e.g. FreeBSD and Solaris). To prove the validity of this approach we will deploy it

within the Grid5000 testbed, an experimental plateform that aims at gathering more than 3000 nodes in France for grid middleware evaluation. We believe that the HIPernet security model can easily be extended to secure any kind of distributed system infrastructure which needs fully distributed authorization solutions.

References

[1] I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," *International J. Supercomputer Applications*, no. 15(3), 2001.

[2] V. Welch, F. Siebenlist, I. Foster, J. Bresnahan, K. Cajkowski, J. Gawor, C. Kesselman, S. Meder, L. Pearlman, and S. Tuecke, "Security for grid services," in *Proc. of* 12^{th} *IEEE International Symposium on High Performance Distributed Computing (HPDC 2003)*, June 2003.

[3] J. Ioannidis, A. Keromytis, and al., "Implementing a Distributed Firewall," in *ACM Conference on Computer and Communications Security*, 2000.

[4] C. Ellison and etc., *SPKI Certificate Theory*, IETF, RFC 2693, September 1999.

[5] R. Moskowitz and P. Nikander, *Host Identity Protocol Architecture*, IETF, draft-ietf-hip-arch-02.txt, January 2005.

[6] C. Castellucia, G. Montenegro, J. Laganier, and C. Neumann, "IPv6 Opportunistic Encryption," in *Proc. of* 9^{th} *European Symposium on Research in Computer Security*. LNCS, Springer, September 2004.

[7] C. G., K. S., S. Ch., and S. G., "Supernetworking: The Next Generation of Secure Enterprise Networking," in *ACSAC 2000*, March 2000.

[8] J. Saltzer and M. Schroeder, "The Protection of Information in Computer Systems," 1975.

[9] D. Price and A. Tucker, "Solaris Zones: Operating System Support for Consolidating Commercial Workloads," in *USENIX Large Installation System Administration Conference (LISA'04)*, November 2004.

[10] P.-H. Kamp and R. Watson, "Jails: Confining the omnipotent root," in *Proceedings of the 2nd Intl. SANE Conference*, May 2000.

[11] A. Bassi and J. Laganier, "Towards an IPv6-based Security Framework for Distributed Shared Storage," in *Proc. of* 7^{th} *IFIP Conference on Communications and Multimedia Security*. LNCS, Springer, December 2003.

[12] L. Barak and O. La'adan, "The Mosix Multicomputer Operating System for High Performance Cluster Computing," *Journal of Future Generation Computer Systems*, March 1998.