

Towards an IPv6-based Security Framework for Distributed Storage Resources

Alessandro Bassi¹ * and Julien Laganier^{2,3}

¹ LoCI Laboratory - University of Tennessee
203 Claxton Building - 37996-3450 Knoxville, TN, USA
abassi@cs.utk.edu

² SUN Microsystems Laboratories Europe
180, avenue de l'Europe 38334 Saint-Ismier Cedex
Julien.Laganier@Sun.COM

³ INRIA Action RESO / Laboratoire de l'Informatique du Parallélisme
École Normale Supérieure de Lyon - 46, allée d'Italie 69364 LYON Cedex 07 - France
Julien.Laganier@ens-lyon.fr

Abstract. Some security problems can be often solved through authorization rather than authentication. Furthermore, certificate-based authorization approach can alleviate usual drawbacks of centralized systems such as bottlenecks or single point of failure. In this paper, we propose a solution that could bring an appropriate security architecture to the Internet Backplane Protocol (IBP), a distributed shared storage protocol. The three basic building blocks are IPsec, Simple Public Key Infrastructure (SPKI) certificates and Crypto-Based Identifiers (CBID). CBID allows entities to prove ownership of their identifiers, SPKI allows entities to prove that they have been authorized to perform specific actions while IPsec provides data origin authentication and confidentiality. We propose to use them to bring some level of 'opportunistic' security in the absence of any trusted central authority. This is particularly tailored to ad-hoc environments where collaborations might be very short-termed.

Keywords: IBP, IPv6, IPsec, authorization certificates, SPKI, CBID, CGA

1 Introduction

In many security approaches the issue of authorization is often overlooked, as the main research focus lies on authentication issues. This is unfortunate, because many security problems have stringent authorization issues rather than authentication problems. Furthermore, an authorization approach avoids usual drawbacks of centralised systems such as bottlenecks or single point of failure, and it's much more suited for high-performance distributed systems.

In this paper, we would like to introduce a solution that could bring an appropriate security architecture to the Internet Backplane Protocol (IBP), a protocol for managing distributed shared storage. The three basic building blocks we are using to provide an acceptable level of security are IPsec, Simple Public Key Infrastructure (SPKI) certificates and Crypto-Based Identifiers (CBID). At present, IBP provides a certain level of security, using an interesting authorization scheme based on cryptographically secure URLs for loading and storing

* This work is supported by the National Science Foundation Next Generation Software Program under grant # 0204007, the Department of Energy Scientific Discovery through Advanced Computing Program under grant # DE-FC02-01ER25465, and by the National Science Foundation Internet Technologies Program under grant # ANI-9980203.

data on a server, but unfortunately not on the most sensitive call of the protocol, the one that allows remote space to be reserved by an end user.

Of those three basic blocks, CBID allows entities to prove ownership of their identifiers, SPKI allows entities to prove that they have been authorized to perform specific actions while IPsec provides data origin authentication and possibly confidentiality. We propose to use them to bring some level of 'opportunistic' security in the absence of any trusted central authority, as the IBP architecture is designed around the concept of using untrusted data depots for holding data for a limited amount of time. As IBP itself does not provide any system to guarantee the integrity and confidentiality of data, these matters have to be taken care by the applications willing to use the IBP infrastructure, and therefore are outside of the scope of this work. We are concentrating especially of potential Denial-of-Service attacks that might occur if an attacker tries to reserve all the available space.

The approach we are following is also particularly tailored to ad-hoc environments where collaborations might be very short-termed.

The paper is organized as follows: in section 2 we will describe the Internet Backplane Protocol. Then, in section 3, we will analyse its security features, and in section 4 we will talk about the basic security building blocks we will use. Section 5 will focus on how those blocks will work together, and, after discussing about the related topics in section 6, we will illustrate the direction of our research in section 7.

2 The Internet Backplane Protocol

The Internet Backplane Protocol [9] (IBP) is a protocol developed by the Logistical Computing and Internetworking (LoCI) Lab from the University of Tennessee to allow an easy sharing of distributed storage resources. The singularity of this protocol is the way of considering those resources completely exposed: any application can allocate some amount of space for a limited amount of time on any server. This key aspect of the IBP storage model, the capacity of allocating space on a shared network resource, can be seen as doing a C-like malloc on an Internet resource, with some outstanding differences, such, for instance, time-limitation. IBP servers, also called *depots* to underline the similarity with the industrial and military logistics, are therefore equipment that allow the sharing of space, either disk or RAM, giving any application the possibility to manage a certain amount of space for a limited time, and therefore allowing end users and applications to explicitly schedule the movement and the position of data.

IBP allocations have to be considered "best-effort", as the server does not guarantee the presence of stored data. Therefore, if reliability of the storage is requested, data replication is necessary, and it must be carried out either through the LoRS tools provided by the same LoCI lab, or directly by the application itself. Because of this particular characteristic, an analogy can be seen between IBP and the Internet Protocol: as IP is a more abstract service based on link-layer datagram delivery, and provides an unreliable, connectionless network service, IBP is a more abstract service based on blocks of data, managed as "byte arrays", providing an unreliable and stateless storage service.

The IBP protocol has not been standardized yet, but efforts in this sense have been made in the realm of the Global Grid Forum, and a final protocol specification is likely to appear towards the end of this year.

IBP Servers have been deployed in around 160 sites, mainly in the United States, providing a publicly available total storage of more than 10 Terabytes. Because of their general-purpose nature, they are well adapted to many different applications, from data staging for scientific calculation, to overlay routing, to

multimedia stream caching. In this last field we can notice several initiatives, the latest one being IBPvo, a mechanism similar to TiVo, a set-up box for recording TV shows on a hard disk support, but based on the IBP infrastructure.

Therefore, we can forecast that this protocol will be broadly used in the future, as its peer-to-peer nature makes it the perfect candidate for the next generation of multimedia sharing software.

3 Security Analysis for IBP

3.1 Threat on *allocate* method

The only mechanism currently implemented to protect depots from Denial-of-Service (DoS) attacks on allocation is Access Control List (ACL). The owner of a depot may choose to define ACL listing IP address list of clients authorized to perform an allocation. Since this verification is based on a longest prefix match of an identifier that has been obtained in an untrusted manner versus some ACL entries, it does not provide a very high level of security, and is particularly vulnerable to IP spoofing and hijacking attacks. An attacker that can snoop a link that carries legitimate and authorized IBP traffic towards a given depot can easily attack this depot by generating either fake read/write/free queries with valid capabilities to destroy other users' data, or fake alloc queries with an authorized source IP address to mount a DoS attack on the exhaustion of the available storage, although, allowing application to retain storage only for a certain amount of time, the risk of having a resource completely taken over for a long amount of time is practically non-existent.

Our belief is that this call (i.e. *allocate*) is the most important, as it allows applications to commit part of a public resource for their private use, and at the same time the more vulnerable one, as apart from the ACL no other mechanism is implemented to protect the unauthorized use of the resource. This work is focused on how to secure this vital phase of the protocol in a manner that both demonstrates scalability and retains a fine-grained resource control.

3.2 Threat on *get* and *put* method

IBP has been designed around the concept of capability, an opaque string returned to a client by a server after a successful allocation, which is the functional equivalent of both a plain-text password and a handler. By providing this capability to his subsequent queries (to put or get data on a depot), the client can prove that either he is the allocator of the storage area, or that the original allocator has authorized him to use this resource by unveiling to him the associated capability. Semantically, those two situations are the same for an IBP server, as the focus is on authorization rather than authentication. This a very practical means of delegating rights to share resources; however, since with the current level of the code no strong cryptographic mechanism (namely authentication and encryption) protect theses exchanges, they could be subject to a wide range of attacks originating from the network layer to the application layer.

As the simple adoption of a publicly available SSL library for any exchange where capabilities have to be passed would provide a sufficient level of security, it would require to modify legacy IBP applications to benefit from. This imply that when each IBP-based application establish a new communication channel intended to carry IBP capabilities, it also needs to perform a SSL/TLS handshake, independently of the fact it may have already perform such an handshake before for protecting a different socket instance. So we decided not to concentrate our attention on this aspect in this paper. Since our scheme use IPsec (IP

packet level), it is almost transparent to applications and allows to factorize the security context establishment between two peers for protecting multiple socket instances.

4 Secure building blocks: IPsec, SPKI and CBID

4.1 IPsec

IPsec [2] is the security architecture for IP. It can provide data-origin authentication, replay protection, non-repudiation and confidentiality to IP datagram delivery.

IPsec processing takes place at the bottom of the IP stack. Each outgoing or incoming packet is matched against the Security Policy Database (SPD) to see which policy must be applied. The selection of the policy is based on the inbound or outbound network interfaces used, source and destination IP addresses, IP protocol carried (e.g. TCP, UDP). The policy specifies if a packet should be dropped, bypass IPsec processing, or secured by an appropriate Security Association. When the appropriate Security Policy has been selected, the kernel looks at the Security Association Database (SAD) to find which algorithms and parameters should be applied to the packet (a two-peers agreement on such parameters is called a Security Association). After the algorithms are applied, the packet continues to flow within the stack.

SAs can be established through manual keying or automatic key exchange with Internet Key Exchange (IKE). The main issue with automatic key exchange is the authentication of so-called "IKE peers". IKE specifies three means of authentication:

- Pre-Shared Keys
- Digital Signatures
- Public Key Encryption

The problem here is that in the absence of a trusted infrastructure, these methods don't allow two previously unknown nodes to successfully authenticate each other: Pre-Shared Keys require prior agreement between the peers, Digital Signatures and Public Key requires both peers to know each other's public key. This knowledge could be achieved either by prior agreement, or by relying on a trusted infrastructure like a Public Key Infrastructure (PKI), a Trusted Third Party (TTP), or a Key Distribution Center (KDC). The approach described in this paper supersedes current limitations on IKE by allowing any two previously unknown nodes to exchange keys.

4.2 SPKI

SPKI [6] stands for Simple Public Key Infrastructure. The vast majority of today's security mechanisms relied on ACLs and PKIs. ACLs define who (or which entity) is authorized to perform a given action (like having read access to a file), while PKIs allow any subject of a common trust domain to bind his name (so called Distinguished Names or DN in X509 nomenclature) to a given public/private key pair.

SPKI specifies a framework which includes the definition of authorization certificates allowing a given public-private key pair to authorize another entity (subject) to perform some actions through the delegation of rights. SPKI authorization certificates differ from standard X509 attributes certificates by the fact that both the issuer (or Certificate Authority or CA in X509 terms) and

the subject are identified by either their public key or a hash of their public key (Whereas with X509, the issuer must be identified by its DN).

A subject may be authorized to do something by an issuer through a chain of several delegatable SPKI authorization 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.

SPKI authorization certificates are very useful objects in a distributed and open system threatened by possible malicious attacks. They can be seen as an ACL entry packed with an in-line PKI certificate. They allow an entity to prove to the controller of a remote resource that he has the right to perform some actions on it, without the intervention of any trusted third party, and possibly without direct contact between the requester and the controller. The delegation feature reduces the management overhead in many different manners, from standard PKI-oriented hierarchical delegation, through small flat groups, to Web-of-Trust Peer-to-Peer communities.

This delegation property is particularly convenient for securing distributed systems because it does not require a centralized management of credentials, thus alleviating some of the potential scalability issues encountered on such large-scale systems (e.g. bottlenecks, single point of failure)

A SPKI authorization certificate has the following general structure:

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

public-key, *signature* and *cert* are objects defined by the SPKI framework. In our experiences, we only uses SPKI authorization certificates in which the *cert* object contains an application-dependent *tag* specifying the attributes of the granted authorization.

4.3 Crypto-Based Identifiers (CBID)

CBID were used by Montenegro et Castellucia ([7]) to help solve the identifier ownership problem in MobileIPv6. Since then, it was also proposed to secure the group management problem for IPv6 multicast/anycast with the help of authorization certificates ([8]), to secure IPv6 Neighbor Discovery, to secure the JXTA peer-to-peer infrastructure ([10]), etc.

The concept of Crypto-Based IDentifiers is quite simple: Starting from a public/private key pair, one could use its public key as one of the input parameters of a secure hash function. The truncated output of the hash is the CBID itself.

$$CBID = h_{128}(PK|Imprint) \quad (1)$$

Where *PK* is the Public Key of the owner of the *CBID* and *Imprint* an input parameter to the secure hash function that allows to restrict the scope of a given CBID. *Imprint* is usually the 64 bits of the node's IPv6 *NetworkPrefix*. h_{128} denotes the truncated output (128 leftmost bits) of the secure hash function h . By doing so, one is able to prove *ownership* of its CBID by computing the digital signature (using its private key) of the message carrying it:

$$msg = CBID|text|PK|SIG_{SK}\{CBID|text|PK\} \quad (2)$$

Where SK is the Private Key associated with PK , $text$ is some data that the owner wants to protect and SIG_{SK} is the digital signature function (usually RSA).

In this scheme, we choose to use IPv6 addresses that are CBIDs, sometimes also called Cryptographically Generated Addresses (CGAs) or Statistically Unique and Cryptographically Verifiable Addresses (SUCV-Address). We construct them by concatenating the IPv6 Network Prefix NP with the 64 leftmost bits of the secure hash output used as an Interface IDentifier (IID):

$$CGIID = h_{64}(PK|Imprint) \quad (3)$$

$$CGA = NP|h_{64}(PK|Imprint) \quad (4)$$

Thus, one would lose half the bits of entropy, but gain a *routable* CBID (i.e. CGA, SUCVAddr) that can be embedded in existent IP protocols like Neighbor Discovery, or, in our case, Internet Key Exchange and Internet Backplane Protocol.

Two peers previously unknown nodes using CGA are now able to authenticate themselves and exchange keys to negotiate some IPsec Security Associations that could be subsequently used by any Upper Layer Protocol (ULP) like IBP.

5 Putting together the building blocks

In this paper we propose the use of existing IPsec security mechanism in conjunction with SPKI certificates and CBID at the network layer to secure the whole stack from the IP network layer, to IBP, used as an ULP for distributed shared storage systems for the Grid.

We assume that the network is no more than a collection of untrusted nodes, and that no trusted infrastructure (e.g. PKI, TTP, KDC) is available. The owner of an IBP depot wants to control who is authorized to allocate storage in his depot, and moreover, doesn't want to be involved in each authorization process because it does not scale with the number of users.

We combine together CBIDs, used as IPv6 addresses (i.e. CGA), with SPKI authorization certificates and IPsec to allow us to verify address ownership, bootstrap a subsequent IPsec Security Association, and verify that a given end-point is indeed authorized to request a given service (like a long term IBP storage allocation).

We propose to authorize the subsequent IBP allocation only if the requesting end-point has transmitted in-line an appropriate SPKI certificates chain. Such a chain must begin by a SPKI certificate signed with the private key of the storage depot, whose issuer is CBID of the depot. If this certificate is not delegatable (*propagate*), then the subject must be the entity trying to allocate space. If the certificate can be delegated, then it can be followed by other certificates which have to be delegatable as well (except the latest). Each certificate subject field of the chain must be the issuer of the next certificate of the chain. These certificate chains allow the maximum flexibility in the representation of the effective trust relationship among a large number of individuals and others entities.

The IBP SPKI authorization certificate (figure 1) includes a tag which defines the maximum space and duration of an allocation on a given depot. The signer can also include an Uniform Resource Identifier (URI) indicating a location in order to perform online revocation check of the certificate. Those revocation check can be conveniently performed toward the issuer of an authorization certificate, as it is probably the most capable entity for doing that. Thus revocation can be performed at the point of authorization, and not on a centralized CRL server. Another solution to avoid the need for a centralized CRL server may be that the

```

(cert
  (issuer (cbid <cbid> )
    (subject (cbid <cbid> )
      (tag
        (ibp-alloc <size> <duration>)
      )
      (propagate)
      (online-test <uri>)
      (not-before <date1>)
      (not-after <date2>)
    )
  )
)

(cert
  (issuer (cbid <3ffe:b89a:10c0:a120:2c48:54ff:fec0:de93>)
    (subject (cbid <2c48:54ff:1ae3:01bb:0ab4:b89a:4f0e:389a>)
      (tag
        (ibp-alloc <5GB> <*>)
      )
      (propagate)
      (not-before <10/1/2002>)
      (not-after <10/31/2002>)
      (online-test
        <http://3ffe:b89a:10c0:a120:2c48:54ff:fec0:de93/crl/latest.der>)
      )
    )
  )
)

```

Fig. 1. IBP SPKI Authorization Certificate

issuer only issues short-lived certificates, thus requiring certificate re-validation from time to time.

Each time an entity wants to perform an action toward a depot, it opens a connection, and packets begin to flow. While the IPsec security policy has been appropriately defined, it should specify that IP packets from protocol TCP and port number IBP_PORT_NUMBER MUST be authenticated and encrypted using ESP transport mode. When the first packet is sent, it match an SPD entry, so the stack will search an appropriate SA to apply it to the packet. If such a SA is not yet in place, IKE would be requested to perform the appropriate key-exchange and SA negotiation using CBID as IPv6 addresses. IKE will then prove address-ownership and bootstrap an ESP transport mode Security Association with the generated Diffie-Hellman shared-secret. This brings both data-origin integrity and confidentiality to the ULP, thus avoiding compromising the IBP capability.

If the security policy is appropriate, the underlying verifiable identifier was already verified when the TCP connection complete three-way handshake. However, it could be safer to verify a second time in the ULP that the host public-key match its CBID or CGA (this can be done through a single call to the CBID library).

In the case of an *allocation* request, the depot verifies that the provided SPKI authorization certificates chains indeed authorizes the requester to allocate the desired space. In the case of *get* and *put*, the depot does not need to verify certificates because the capability proves that the authorization has been given by the owner of the resource (indeed, the owner had given authorization by disclosing a capability).

Since the TCP connection is protected by IPsec encapsulation from IP spoofing, hijacking and snooping, an attacker cannot claim to be someone that is

authorized to allocate nor can he learn some capabilities that would authorize him to read/write/manage associated data buffers.

6 Related work

The approach presented here differs from classical other security approaches targeted at large scale distributed systems by the fact that it does not rely on the existence of a global trusted infrastructure to authenticate users and others entities.

Some of these frameworks try to federate multiple security framework (such as Kerberos, DCE, PKI) under a global PKI, like in Globus Security Infrastructure [11]. Others propose to lay on top of existant remote untrusted storage (e.g. NFS, CIFS, P2P, HTTP) a layer that bring security like in SiRIUS [12]. Both requires an always on-line PKI top work.

Theses approaches have a lot of advantages in terms of re-usability of existing components, but they don't address the major problem in distributed systems which is scalability. Actually, there is no Internet-wide deployed PKI, and some peoples thinks that the nearest thing we could achieve is DNS-SEC. Assuming that this is the case, it severely limits the real scalability of such solutions. Scalability concerns may push users to use their own trusted Certificate Authority, thus fragmenting the unified grid environment.

From another point of view, the fact that these architectures allows local administrator to enforce their locally implemented security policy while joining an existent grid computing domain is extremely valuable in terms of scalability because it gives much more flexibility in deployment (no modification of locally implemented security mechanisms is required). Apparently the scalability of a grid security architecture seems to be a tradeoff between agregation-centralization (reducing management and heterogeneity) and distribution (avoiding possibles bottlenecks, single-point of failures, etc.). We choose to address the second point.

Meanwhile, some researchers have already build completely distributed security systems using SPKI or Keynotes authorization certificates. The most related solutions have been described to implement distributed firewalls [14] and Trust Management in IPsec [13].

Our contribution bring to the existent approaches some interesting way to verify the identifier used within network protocols because we use the public key (in fact, the hash of the public key) as an identifier in both the architecture and the protocols we want to secure. This is particularly convenient for securing existent protocols because as long as protocol's identifiers are as long as needed (about 128 bits), you can replace them by CBIDs and gain a means of verifying identifiers embedded in the protocol itself (without requiring to modify it). This is also convenient by the fact that protocols identifiers are usually generated with the address as an input parameter, and that the address is a CBID in our solution, so in addition to being able to verify the identities of your correspondent in the ULP (using CBIDs and SPKI) and at the network layer (using IPsec and CGA), you can benefit from a Cryptographic Layering Enforcement of the complete protocol stack. The ULP is then securely layered on top of TCP/IP. Hence, an attacker cannot succeed while launching an attack towards the lonely ULP because IPsec prevents IP spoofing and hijacking, and the ULP's identifiers are equivalently tight to both the public key and the IPv6 Cryptographically Generated Address.

7 Future Works

In this paper, we have presented new solutions to secure the most sensitive part of a distributed and shared storage infrastructure, the resource allocation phase.

The Internet Backplane Protocol provides a complete storage infrastructure inside the network in the form of distributed data depots which allow users to deploy and temporarily store their data. Currently more than 160 depots are available worldwide (mainly at Internet2 and PlanetLab nodes, but also at many independent locations), with an aggregate storage capacity of around 10 Terabytes. Depots must rely on replicated authorization mechanisms to move data between them in a secure way.

We showed how to combine together security protocols (IPsec, SPKI and IPsec) for allocation authorization in IBP depots. We are currently implementing this model inside the Internet Backplane Protocol software suite. This implementation will be completely transparent to the protocol itself, and it will be easily usable by IBP servers and clients.

A further step will be to rethink through the capability mechanism that IBP uses for read/write authorization rights and see if our framework could be applicable, and at what price in terms of structural modification to the current scheme.

Because of its elegant architecture, this security model can easily be extended to other Peer to Peer or Content Delivery infrastructure which need fully distributed authorization solutions.

References

1. S. Deering, B. Hinden, "Internet Protocol version 6 (IPv6) Specification", *RFC 2460*, December 1995.
2. S. Kent, R. Atkinson, "Security Architecture for the Internet Protocol", *RFC 2401*, November 1998.
3. S. Kent, R. Atkinson, "IP Authentication Header", *RFC 2402*, November 1998.
4. S. Kent, R. Atkinson, "Encapsulating Security Payload", *RFC 2403*, November 1998.
5. T. Dierks, C. Allen, "The Transport Layer Security (TLS) Protocol", *RFC 2246*, January 1999.
6. C. Ellison et al., "SPKI Certificate Theory", *RFC 2693*, September 1999.
7. G. Montenegro & C. Castellucia, "Statically Unique and Cryptographically Verifiable (SUCV) Identifiers and Addresses", *9th Network and Distributed System Security Symposium (NDSS)*, February 2002.
8. G. Montenegro & C. Castellucia, "Securing Group Management", *ACM Transactions on Security (T-SEC) 2002*, February 2001.
9. J. Plank, A. Bassi, M. Beck & al., "Managing Data Storage in the Network", *IEEE Internet Computing*, September-October 2001.
10. G. Montenegro & D. Bailly The Crypto-ID JXTA project web site, <http://crypto-id.jxta.org>
11. The Globus project web site, <http://www.globus.org>
12. E. Goh, H. Shacham, N. Modadugu, D. Boneh, "SiRIUS: Securing Remote Untrusted Storage", *Proc. Network and Distributed System Security Symposium (NDSS)*, February 2003.
13. J. Ioannidis, A. Keromytis & al., "Trust Management for IPsec", *Proc. Network and Distributed System Security Symposium (NDSS)*, February 2001.
14. J. Ioannidis, A. Keromytis et al., "Implementing a Distributed Firewall", *Proc. ACM Conference on Computer and Communications Security 2000*.