Security in Programmable Network Infrastructures: the Integration of Network and Application Solutions

Paolo Bellavista[1], Antonio Corradi[1], Rebecca Montanari[1], Cesare Stefanelli[2]

[1]Dipartimento di Elettronica, Informatica e Sistemistica
Università di Bologna
Viale Risorgimento, 2 - 40136 Bologna - Italy
{pbellavista, acorradi, rmontanari}@deis.unibo.it
[2]Dipartimento di Ingegneria
Università di Ferrara
Via Saragat, 1 - 44100 Ferrara - Italy
cstefanelli@ing.unife.it

Abstract. Programming the network infrastructure significantly enhances its flexibility and favors fast deployment of new protocols, but also introduces serious security risks. It is crucial to protect the whole distributed infrastructure, especially its availability in case of denial-of-service attacks. A security framework for programmable networks may provide security solutions at different levels of abstraction. Active networks mainly propose a network-layer approach, by extending the packet format to include security information. Mobile code technologies tend to provide security tools at the application layer to integrate with standard external infrastructures, such as public key ones. The paper describes the security frameworks of several programmable network proposals and points out the dis/advantages related to the adopted abstraction level. This comparison suggests to consider an integrated security framework capable of choosing the service-specific balance between application-layer flexibility and network efficiency. To this purpose, the paper presents the architecture of a Programmable Network Component (PNC) that integrates security solutions at different layers and that has been implemented by using a mobile agent programming environment.

1 Introduction

The convergence of telecommunication systems and the Internet proposes a global shared network infrastructure with new value-added services for all participants (final users, service providers, and network operators) [1]. The management of the infrastructure is increasingly complex, because of its global dimension, of network resources heterogeneity, of the request for dynamicity in offered services, and of increasing user requirements and expectations. To satisfy these requirements, the traditional end-to-end model of interaction in the network is evolving toward an alternative scenario where the network infrastructure can play an active execution role. In particular, in Programmable Networks (PN), interconnection components can perform computations on transmitted data and can be programmed by dynamically injecting service/user-specific code [2]. Several approaches and technologies have been proposed for the realization of PN, and can be roughly classified on the basis of the principal abstraction layer: the term Active Networks (AN) usually identify the approaches that achieve programmability by working mainly at the network layer, whereas we consider Mobile Agents as an enabling technology that achieves programmability at the application layer [2].

Many research groups have recently claimed PN suitability for a wide spectrum of applications [3] [2]. PN can help in fast prototyping and deploying new network-layer protocols (e.g., for congestion control and topology-aware reliable multicast). Other proposals employ network programmability to deal with application-specific requirements, as in Web caching and in dynamic adaptation of multimedia streaming to currently available resources [4] [2]. All application scenarios require that PN environments provide adequate answers to the security issues raised by network programmability. The main security concern is to achieve a full protection of the shared network infrastructure against illegal accesses and denial-of-service attacks.

The paper discusses some different security solutions in the PN research area depending on their specific level of abstraction. Some approaches in the AN area suggest the adoption of security mechanisms at the network layer. They usually tend to standardize security data by directly enclosing them into packets [5] [6]. Other approaches propose solutions at a higher level of abstraction, to exploit the flexibility and extensibility typical of the application layer [3]. On the one hand, network-layer approaches focus on efficiency but often lack flexibility and dynamicity. On the other hand, application-layer solutions permit to integrate with existing infrastructures for rapid prototyping and deployment, but do not often achieve performance.

The paper presents the architecture of a Programmable Network Component (PNC), designed to fast prototype and deploy protocols/services in the global, heterogeneous and untrusted Internet environment. In particular, the paper focuses on security aspects and proposes the integration of network- and application-layer solutions. An integrated approach to security permits service designers and system managers to satisfy different security requirements, from high dynamicity in the modification of security data to strict respect of timing constraints, from interoperability with existing infrastructures to scalability, crucial for handling a large number of users. We claim that only a solution that integrates mechanisms and tools at both layers can achieve the efficiency of the network layer together with the flexibility of application-layer solutions. The proposed PNC architecture has been implemented by using a Mobile Agent (MA) framework called Secure and Open Mobile Agent (SOMA) [7]. The SOMA platform exploits the Java technology for agent serialization, dynamic class loading, networking support and for the ubiquitous availability of the Java virtual machine.

2 Security Issues in Programmable Networks

Network programmability raises significant concerns from the security point of view. Possible threats and attacks to PN are far more critical than in passive network infrastructures. In fact, the possibility of injecting code to modify the behavior of network components can compromise not only the correct operations of one node but also the availability of the whole network. To face these threats, the design of a general PN environment should grant an adequate level of security since its first phases, and security cannot be considered an add-on to insert only a posteriori.

The security framework of a PN environment should be based on a thorough security model to protect all involved entities, both network infrastructure (the set of all programmable nodes) and active packets (the single pieces of code injected into the network). More in detail, it is necessary to protect:

- the network resources against malicious behavior of active packets, to maintain the availability of the shared network infrastructure;
- the active packets against attacks from malicious network nodes, to grant the correctness of the service provided by active packets over the whole network path between service users and providers;
- the active packets when transiting in the network, to detect possible modification and to prevent malicious sniffing;
- the active packets from interfering with each other, to avoid the possibility of combined attacks performed by colluding active packets.

The PN security framework should answer the fundamental issues of authentication, authorization, secrecy, and integrity and should provide the requested models of trust. Any trust model defines who or what in the system is considered trusted, in what way, and to what extent [8].

Authentication permits to associate active packets with responsible principals, where principals represent the subjects that request the operations, e.g., an individual, a corporation, a service provider, and a network administrator. In practice, any principal can be associated with personal public/private keys and digitally signs packets to ensure the correct identification of their responsible party. The authentication process safely verifies the correspondence between principal identities and keys. Most authentication solutions delegate key lifecycle management to Public Key Infrastructures (PKIs) [9]. Authentication also ascertains the paternity of active packets by associating them with either their principal or their responsible role. A role models a collection of rights and duties that characterizes a particular position within an organization. A role-based model facilitates the administration and management of a large number of principals, by simplifying the dynamic handling of principals and permissions [3].

Authorization grants active packets the permissions to operate on the resources of the network infrastructure. Several authorization models are possible: the most common is the Access Control Lists (ACL) model that describes and enforces the access rights of principals/roles on a resource. More generalized models, such as the

Trust Management model, can provide a unified framework for the specification and interpretation of security policies in distributed systems [10].

In addition, the security infrastructure should prevent the possibility of modifying and inspecting active packet contents (integrity and secrecy issues) while migrating over untrusted networks and executing in malicious nodes. When considering the protection of an active packet in transit over an end-to-end communication channel, traditional cryptographic techniques can establish secure channels to ensure both integrity and secrecy between end-to-end network nodes. This approach is not sufficient in the PN area where intermediate hosts have to verify incoming active packets before their execution. This requires a hop-by-hop control that implies the establishment of a trust relationship between all involved intermediate nodes [2].

In PN environments another issue concerns the possibility to control the behavior of incoming active packets while in execution. Several PN infrastructures confine the execution of different active packets into isolated environments to prevent reciprocal interference, and to avoid possible collusion against the hosting network node and provide monitoring services to exclude excessive resource consumption that can lead to possible denial of service attacks [11].

A general security framework for PN environments should provide strategies and mechanisms of solution for all the above issues. The same infrastructure can offer different solution implementations to make available different qualities of security service. In any case, some general properties have to be considered to deal with global and heterogeneous distributed systems such as PN.

The basic requirement to satisfy is the *durability* of design efforts. Whenever a system has been completely deployed, its lifetime strictly depends on its capacity of following the evolving needs. In other words, the security model should be flexible enough to accommodate any suitable variation and should extend easily to embody reasonable additions to components. These *extensibility* and *flexibility* properties can be achieved along synergic guidelines, preventive solutions and design technologies that favor the addition/substitution of system modules. For instance, the association of one principal with several roles can help in changing the principal permissions to adapt to different and evolving environments. The same is for the versioning of security tools, which can coexist in different versions within the same system at the same time, if the design maintains sufficient information to distinguish between the different installations.

Another requirement is *dynamicity*. PN are global systems and the availability of the network infrastructure is a necessary condition. For this reason, all security solutions have to maintain system effectiveness while incorporating variations. For instance, while a programmable router is receiving a new protocol version that affects the handling of specific streams, not only routing operations should go on, but also no packets (either active or normal) should be lost.

A final but fundamental consideration for the implementation of a PN security infrastructure, which influences all design choices, is to meet an adequate level of *performance*. PN call for security solutions capable of meeting cost requirements and of achieving a suitable trade-off between the necessary security degree and the usage of time and resources.

3 Security Solutions in Programmable Networks

The aim of this section is not to provide a general survey on the state-of-the-art of the PN research, but to organize and give some technical insights about the projects that have specifically worked on solutions at different levels of abstraction for the PN security issues.

According to this guideline, we first present two architectures that base their security solutions on the insertion of security data directly within transmitted packets. They are the Secure Active Network Environment (SANE), employed in the SwitchWare project at the University of Pennsylvania [12] and the Smart Packets (SP) proposal of the BBN [13]. Other PN approaches tend to rely on security mechanisms and tools that are more at the application layer: Section 3.2 presents some trends emerging in these PN architectures and gives some insights of the Agent-Based Security Architecture for the Active Network Infrastructure (ABSANI) developed at the GMD Fokus [3].

3.1 Security Data within PN Packets

Several research efforts have addressed the issue of defining new formats for network packets to include security-relevant information [5] [6]. These activities propose a structure of packets that permits efficient security processing at packet forwarding/reception, on the basis of the security data contained in packet headers.

The most recognized work toward the standardization of the PN packet format is the Active Network Encapsulation Protocol (ANEP) [5] that proposes a common base to increase interoperability among different PN projects. The main purpose of ANEP is to fast identify the environment in which to evaluate incoming active packets by examining the content of specific fields in packet headers. ANEP packets can be transmitted directly over the link layer, or they can be encapsulated within an existing network protocol such as IP.

The current security support in ANEP is limited to the provision of one-way authentication with X.509 and SPKI self-signed certificates [9]. All the network-layer security approaches in the PN area have proposed specific extensions to the ANEP header to provide and manage security issues in a more general way.

3.1.1 Secure Active Network Environment

SANE provides a layered security architecture to ensure the correct behavior of incoming active packets. At the lower layers, SANE guarantees that its PN components start in an expected state by exploiting a secure bootstrap mechanism, called AEGIS [12]. The higher layers are responsible for active packet authentication/encryption, for the provision of a restricted execution environment based on a type-safe dedicated language for active packets [3], and for the safe partitioning of separate name spaces to the different node services.

SANE extends the ANEP format to support packet authentication and secrecy. The approach is similar to the IPsec protocol and to its provided security associations [6].

The SANE packet includes a Security Parameter Index (SPI) to identify uniquely the corresponding security association. Figure 1 shows the SANE authentication header that provides a first basic mechanism to detect replay attacks and that ensures packet origin and data integrity.

SANE employs a secret-key scheme that requires a preliminary application-layer negotiation, called Key Establishment Protocol (KEP), to determine security associations between all involved PN nodes and between the PN infrastructure and its users. At bootstrap, the PN nodes verify the accessible network topology and perform KEP steps with adjacent nodes. After two PN parties have achieved mutual authentication and agreed on the utilization of specified secret keys and cryptographic algorithms, they can start to exchange authenticated and encrypted active packets. Any PN node stores negotiated parameters locally until the corresponding security association is broken. In particular, active packets can follow a path that involves a large number of PN nodes, such as in multiple-hop PN protocols, if all involved PN nodes have mutually established security associations.

SANE exploits secret-key-based security for the sake of performance and limits the public-key usage only during the security association phase. In addition, the information required to perform security checks is maintained locally at the active nodes: when receiving an active packet, the SANE node locally retrieves the security parameters indexed by the SPI to complete the verification of the authenticator integrity.

The SANE authorization support exchanges information about user permissions over authenticated and encrypted channels, established by using ANEP-compliant packets. In particular, KeyNote-based credentials [10] specify the policies to rule the operations of active packets on system resources.

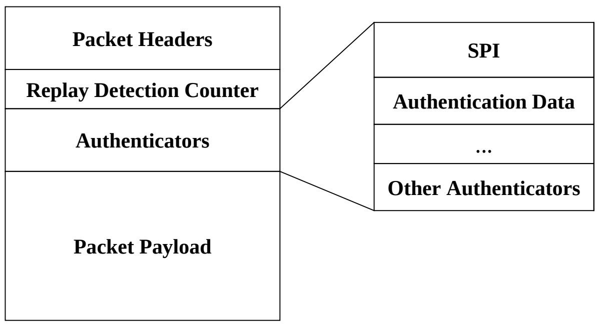

Fig. 1. The SANE packet format

3.1.2 Smart Packets

SPs have been proposed in a DARPA project that focuses on the PN application to network management. SP distinguishes two different modes, end-to-end and hop-by–hop. In the end-to-end mode, only SP endpoints can execute SP protocols, while in the hop-by-hop mode the source, the destination and all SP intermediates actively participate to deploy the active protocol.

To concentrate on the adopted security solutions, SP authenticates the origin of active packets by checking their integrity and by providing a confined and controlled execution environment. A dedicated specialized language, Sprocket, limits the operations permitted to SP active packets.

SP are encapsulated within ANEP via the definition of a specific SP header (*basic authenticator*), shown in Figure 2. The authenticator permits both to identify the origin of the packet and to verify the integrity of its non-mutable portions, by exploiting public-key algorithms. SP designers extended the ANEP header to accommodate the hop-by-hop mode, which models the case of intermediate SP nodes that operate and transform the active packet contents. To avoid the need of an integrity check at any hop, SP carry an authenticator that omits the payload and the packet length field in the ANEP header.

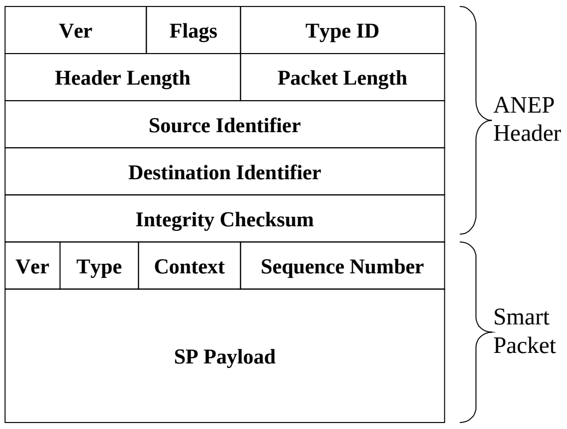

Fig. 2. The SmartPacket format

At the reception of one SP packet, its origin and the integrity of its non-mutable parts are verified. Any SP node checks the received authenticators on the basis of user certificates, either included in the SP payload or requested to an external application-layer PKI. There is no limit to the number of certificates that can be included within an SP packet, obviously apart from the maximum packet size. X.509 standard certificates directly enclosed into SP packets reduce the space left for the code, but may significantly improve the efficiency of security controls. The SP performance, in fact, is tightly connected to the local availability of needed certificates at the intermediate SP nodes.

If the packet origin and integrity verification process fails, the packet is discarded; otherwise, the packet enters the authorization process that employs ACL mechanisms to control active packet execution.

3.2 Security Solutions at the Application Layer

Most PN proposals implement security solutions completely at the application level, without affecting the content of the transmitted active packets. Main motivations are the integration with mechanisms, tools and infrastructures already developed for securing distributed services, the fast prototyping of security solutions via software simulations of network components, and the simplified support for flexible and programmable security in PN.

All these PN proposals have chosen Java as the implementation technology. This choice is motivated not only by Java portability over heterogeneous platforms, but also by its security-related properties, such as safeness (strong typing, lack of pointers, automatic memory management) and availability of security mechanisms both at the language level and at the run-time support one [14].

For instance, the Intel framework for PN (Phoenix) exploits Java authorization and access control to rule the access to active node resources by mobile agents that implement congestion analysis and intrusion detection active protocols [15]. The Java authorization mechanisms are extended with proprietary monitoring and management functions that permit to change agent priority levels and to dynamically reject resource requests depending on current resource load. Another example is the Lucent PN prototype for distributed network management where legacy routers are enhanced with a Java-based active engine that runs on a general-purpose workstation [16]. The Lucent system exploits the standard Java SecurityManager to avoid possible interference between different flows of active packets and to control the associated session environments at run-time, by preventing the access to native methods and to some protected parts of the file system.

We give in the following some details of the ABSANI architecture because it is an MA-based PN system specifically developed with the goal of providing a flexible, open and interoperable security framework. ABSANI is Java-based, and its developers are skeptical about the introduction of dedicated programming languages for the PN area [2] [3] [13].

ABSANI completely isolates the execution of injected agents into abstraction localities called places to prevent any interference among executing mobile agents. In particular, several isolated places can be concurrently present on the same ABSANI node. A resource manager component acts as a mediator in the interactions between agents and node resources. The resource manager can also provide the basic mechanism for auditing: it can collect the data generated by network activities to identify the users responsible for security breakouts. In addition, it provides control and management operations to change the overall system behavior, e.g., the modification of local security policies is only allowed from a dedicated place responsible for node management.

Agent authentication is based on credentials that permit the association of agents with responsible principals and the control of agent actions according to the local security policies. Credentials can vary to include standard X.509 and SPKI certificates, the hash of packet contents, the list of its signers and their signatures, etc. The granted permissions result from the intersection of two policies, at the place and at the node level. Policies can be administered via application-layer management tools; abstractions such as groups and roles for principals can further enhance policy manageability [3].

3.3 Comparing the Approaches: Security at Which Layer?

The above discussion has shown that various frameworks can address security at different levels of abstraction, by exploiting features typical of either the network layer or the application one. The awareness of the advantages deriving from the security approaches at different layers favors a more knowledgeable choice in the trade-off between security requirements and expected performance. The correct choice impacts crucially on the acceptance of deployed PN protocols and can widen the range of application areas of PN secure services.

Other areas have already faced a similar debate about security provision at different layers. The request for Web secure services has motivated the introduction of application-layer secure protocols, such as Secure HTTP and SSL. Their extensive usage has stressed the need for more efficient solutions that can be provided by working at the network layer as in the IPsec proposal. However, the discussion about at which layer security should be provided is still open [6].

The main advantage of network-layer solutions for PN is efficiency in exchanging authenticated and encrypted packets. The encapsulation of security information within packet headers permits to perform security checks at the network layer by saving packet security processing to upper layer protocols. However, some security issues can only be dealt with at higher levels of abstraction, e.g., the management of authentication and authorization services. For instance, SANE can achieve the performance typical of network-layer solutions, after the security associations have been established between all nodes in the active packet path; but this preliminary negotiation phase works at the application layer.

The application-layer approach simplifies the support to system durability because solutions at this level can provide flexibility, extensibility and dynamicity to the management of security services. For example, a policy/role management service demands solutions to simplify administrator operations of adding and changing policies/roles. The embedding of this functionality directly in network-layer packets could imply continuous extension of PN protocols and formats, to accommodate evolving requirements and facilities, and this would clash with the need of keeping the packet size to the minimum. In addition, application-layer solutions simplify the implementation of an open and interoperable security architecture, capable of integrating with diffused standard security frameworks that exploit state-of-the-art technologies, as shown in [3].

The above considerations motivate the design of security frameworks that integrate the two layers, by taking advantage of both the efficiency deriving from embedded protocols at the network layer and the expressive capacity stemming from solutions and tools at the application layer. PN administrators and users can exploit the frameworks to find their specific balance between performance and flexibility, depending on particular service requirements and the level of trust of the target environment of operations.

4 The Programmable Network Component

We have developed a framework for the fast prototyping and deployment of protocols and services that is based on a Programmable Network Component (PNC) to be installed in the nodes of the network infrastructure. The PNC supports active protocols and services expressed in terms of mobile agents that employ the migration and communication services of the SOMA programming environment [7]. The PNC is built on top of the JVM to exploit the Java inherent support for dynamic class-loading, platform independence and security.

Mobile agents are used to distribute the behavior of active nodes out-of-band and to support the dynamic extension of active node functions [2]. In addition, MAs permit the easy installation of service- and user-specific protocols that can be injected dynamically into the network. Our PNC provides a secure environment for agent-based active protocol execution, with a wide range of security solutions at different layers. The main guideline is to combine the efficiency of basic security features implemented at the network layer together with the flexibility and extensibility of more advanced security tools and infrastructures provided at the application one.

The PNC is designed to support differentiated protocols that can coexist in the same node without reciprocal interference. For this purpose, the PNC provides isolated environments for agent execution called *places* (see Figure 3). A component called dispatcher is present in any PNC node to forward incoming packets to the agent responsible of their handling depending on the specific security and management policies of the PNC node. The PNC support ensures a protected binding between loaded agents and local node resources. The binding is implemented via a proxy-based mechanism where each node resource is encapsulated and available via a proxy object. Agents refer initially only to these proxies with no possibility to access resources directly. In particular, any resource proxy exports a `Resource` interface with the `getEnvironment()` method that agents have to call to access the managed resources. The proxy accepts requests for its resources and determines whether to allow the agent access on the basis of the node security policy. For instance, returned references can depend on the role dynamically associated with the agent principal. To improve efficiency, agents are forced to pass via the proxy only once at first retrieval of resource references, whereas afterwards they can maintain these references locally. Any PNC node takes advantage of a set of basic security services that include:

- the *secure transport* service that provides integrity and secrecy for the transport of agents between PNC nodes. At agent arrival at any PNC node, security checks are performed to ascertain if integrity and secrecy have been preserved during agent transport;
- the *authentication* service that accepts/discards agents on the basis of their corresponding user identities and roles. Cryptographic operations are performed to

verify the X.509 identity and role certificates, possibly locally to the PNC. If the verification succeeds, agents can be dispatched to the correct place, otherwise forwarded to a severely restricted default environment that support anonymous agent execution;

- the *safe checking* service that exploits the Java class verifier to ensure agent class file conformance to the JVM specification. Static checks avoid stack over/underflow, and dynamic controls are provided to grant correctness of symbolic references. Agents not satisfying the safety property are discarded;
- the authorization service that extends the Java security architecture to permit the utilization of a role-based access control model. Security policies rule the access of agents to all local PNC resources, both shared and private ones, that are available in the execution place. Authorization checks are performed by resource proxies when the `getEnvironment()` method is called. The access control policies define the set of permitted references for the requesting agents.

It is worth noting that some security checks, such as the integrity, secrecy, and authentication ones, can be implemented at the network-layer to improve efficiency. However, even these security services require to integrate with application-layer solutions in order to be exploited in large scale networks.

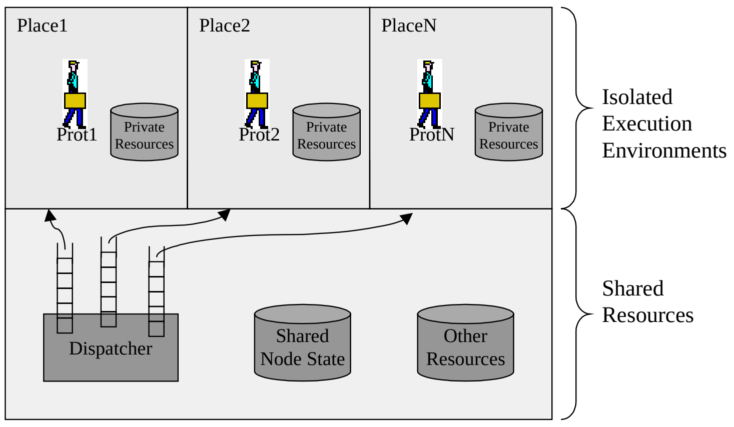

Fig. 3. PNC isolated environments for agent execution

4.1 Network-layer Solutions

We have designed the PNC security architecture to achieve the needed degree of extensibility to permit the addition of new security features without modifying or recompiling existing security components. To this purpose, the PNC framework includes several modules that provide similar security services, but with different properties in terms of flexibility and performance. This permits to configure and install the most proper solution depending on application-specific requirements.

The modularity of the approach applies to the implementation of the authentication and the secure transport services, which are provided by either the ANEP module or the IPsec one (see Figure 4). The ANEP-compliant active packets exploit the `TypeID` and `Option` fields to indicate respectively the identifier of the involved MA-based protocol and the authenticator data, in the same way as in SANE. By now, there are no hardware implementation of ANEP-compliant routers and the possible performance improvements cannot apply to real service protocols. We are also completing the implementation of the alternative IPsec module that adopts the IPsec network-layer protocol to provide secure transport and authentication services. We are currently working on the IPsec module implementation on a dedicated IPSec-compliant hardware component, the TimeStep VPN Gateway [17].

Both the ANEP module and the IPsec one can be configured to use standard public key cryptography mechanisms and X.509 certificates that can be distributed, revoked and suspended by an external application-layer PKI [9]. The integration of both modules with a PKI can further simplify the modularity and interchangeability of the implementations.

4.2 Application-layer Solutions

Advanced application-layer security services are implemented on top of the basic security services to improve the manageability, scalability, flexibility, and dynamicity of the basic security services (see Figure 4).

The certificate management service is used to enhance the manageability and scalability of the secure transport and authentication services by supporting keys/certificates distribution, revocation and suspension. The service is offered by the Entrust PKI [18] that permits to provide transparent and automatic key management in application-specific components written in different programming languages, e.g. Java. The certificate service is implemented to realize a local cache of most recently used X.509 certificates and certificate revocation lists at any PNC node to improve the efficiency of integrity, secrecy and authentication checks. When security operations require certificates that are not present in the local cache, the needed certificates are requested to the Entrust PKI together with their corresponding revocation/suspension status. It is worth noting that in a realistic scenario different PNC administrators may wish to adopt different PKI solutions depending on their peculiar management and security policies. For this reason, we are also examining the interoperability issues that stem from the integration of our PNC with different and heterogeneous PKIs. In addition, all the basic security services can benefit from the *policy/role management* service. This service increases the usability of access control policies when dealing with a large-scale PNC network infrastructure that provides services to a potentially large number of users. The service adopts the Ponder policy language [19] to model the actions that agents are permitted/forbidden to perform on the PNC node.

In addition, it provides the required support to map Ponder policy specifications into platform-dependent policies that can be interpreted and enforced at run-time in the system. In particular, the service includes a policy/role graphical user interface for the specification, editing, and administration of policies/roles and a policy repository, local to the PNC node, for the storage and retrieval of policy/role information. The policy/role management service is designed to support dynamic roles/policies modifications with no need to suspend PNC operations. Administrators can modify the security policies of the managed resources and the changes are propagated automatically to involved PNC nodes, and consequently to the resource proxies.

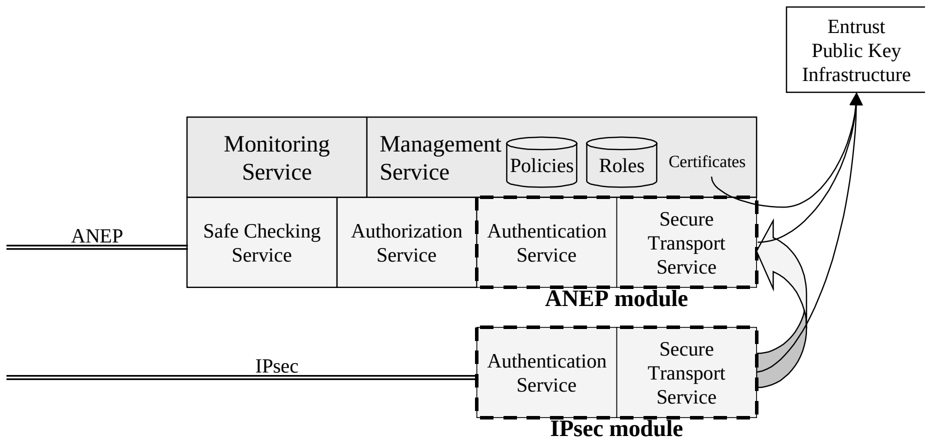

Fig. 4. The PNC architecture of security services

The PNC node also provides an *on-line monitoring* service that permits system managers to control and prevent any agent excess in resource consumption, by making available the usage of PNC local resources. The monitoring service can be configured to visualize the utilization of the local processor, the quantity of used memory and the generated network traffic, both for any Java thread and any other process outside the Java Virtual Machine. To reduce the overhead effect of on-line monitoring on PNC performance, our monitoring service can be dynamically tuned to observe only a subset of executing threads, possibly with different observation frequencies. For instance, to face denial-of-service attacks, we collect the CPU utilization percentage only for the agent threads responsible of active packet execution; when one thread exceeds a threshold, the PNC alerts the system administrator and begins to collect and visualize all possible monitoring information about the specified thread, with a possibly increased frequency.

The collected monitoring information is obtained in two different ways. On the one hand, we exploit platform-dependent functionality (Solaris/Linux `/proc` directory, Microsoft WindowsNT system registries), integrated in the PNC via the Java Native Interface [20]. On the other hand, to permit fine-grained monitoring visibility of all Java threads, we use the novel Java Virtual Machine Profiler Interface. The JVMPI is proposed by Sun within the latest version of the Java platform, to notify Java applications of several kinds of events that can take place in the virtual machine [21]. The result is a common monitoring API that abstracts from the PNC hosting platform (Solaris, WindowsNT and Linux are currently supported) and that is mapped transparently to the correct platform-dependent dynamic libraries at run-time.

5 Final Remarks

In the global environment proposed by the Internet, many application areas have experienced an exponential growth in the number of interested developers and users. There are several services and protocols that could add new impulse to this scenario, but their deployment is currently limited due to the long and difficult standardization process. The application of PN technologies to the Internet infrastructure could boost even more its importance, because PN could accelerate the deployment of new service-specific protocols that can be installed at run-time.

However, the PN potential has not been exploited completely because of the lack of general agreement on comprehensive and accepted security frameworks. Only the definition of general security standards, or, at least, of more precise security recommendations, can produce the momentum needed to grant durability to the PN design efforts.

The paper has considered how several PN proposals have faced the issues connected with security. The paper does not give a complete classification but should help security service designers in better understanding the properties offered at different layers. The aim is to drive the design of a PN security framework offering a wide range of solutions and tools to compose the contrasting requirements of flexibility and efficiency.

As a final consideration, PN emphasize programmability for network components but also call for programmability of the security framework itself, to fully adapt to different environments, to diverse user expectations, and to various requirements in performance.

Acknowledgments

Investigation supported by the Italian “Consiglio Nazionale delle Ricerche” in the framework of the Project “Global Applications in the Internet Area: Models and Programming Environments” and by the University of Bologna (Funds for Selected Research Topics: “An integrated Infrastructure to support Secure Services”).

References

1. Aaron, R., Skillen, R. (eds.): Special Section on Electronic Commerce. IEEE Communications Magazine **37**(9), 1999
2. Psounis, K.: Active Networks: Applications, Security, Safety, and Architectures. IEEE Communications Surveys. http://www.comsoc.org/pubs/surveys (1999)

3. Covaci, S. (ed.): Proc. 1st Int. Working Conference on Active Networks (IWAN'99). Lecture Notes on Computer Science, Vol. 1653. Springer-Verlag, Berlin Heidelberg New York (1999)
4. Amir, E., McCanne, S., Katz, R.: An Active Service Framework and its Application to Real-time Multimedia Transcoding. Computer Communication Review **28**(4), 1998
5. Alexander, D.S., et al.: Active Network Encapsulation Protocol (ANEP). RFC draft (1997)
6. Oppliger, R.: Security at the Internet Layer. IEEE Computer Magazine **31**(9), 1998
7. Bellavista, P., Corradi, A., Stefanelli, C.: Protection and Interoperability for Mobile Agents: A Secure and Open Programming Environment. IEICE Transactions on Communications, IEICE/IEEE Special Issue on Autonomous Decentralized Systems **E83-B**(5), 2000
8. Schneier, B.: Cryptographic Design Vulnerabilities. IEEE Computer Magazine **31**(9), 1998
9. Ford, W., Baum, M.S.: Secure Electronic Commerce - Building the Infrastructure for Digital Signatures and Encryption. Prentice Hall (1997)
10. Blaze, M., et al.: The Role of Trust Management in Distributed Systems Security. Secure Internet Programming: Issues in Distributed and Mobile Object Systems. Lecture Notes on Computer Science. Springer-Verlag, Berlin Heidelberg New York (1999)
11. Wang, P.Y., Yemini, Y., Florissi, D., Zinky, J.: A Distributed Resource Controller for QoS Applications. IEEE/IFIP Network Operations and Management Symposium (2000)
12. Alexander, D.S., et al.: A Secure Active Network Environment Architecture: Realization in SwitchWare. IEEE Network Magazine **12**(3), 1998
13. Schwartz, B., et al.: Smart Packets for Active Networks. 2nd IEEE Conference on Open Architectures and Network Programming (1999)
14. Gong, L.: Inside Java 2 Platform Security: Architecture, API Design, and Implementation. Addison-Wesley (1999)
15. Raz, D., Shavitt, Y.: Active Networks for Efficient Distributed Network Management. IEEE Communications Magazine **38**(3), 2000
16. Putzolu, D., Bakshi, S., Yadav, S., Yavatkar, R.: The Phoenix Framework: A Practical Architecture for Programmable Networks. IEEE Communications Magazine **38**(3), 2000
17. TimeStep - http://www.timestep.com/HTML/ProdConGate.htm
18. Entrust, Entrust Technologies Inc. - http://developer.entrust.com/
19. Damianou, N., et al.: Ponder: A Language for specifying Security and Management Policies for Distributed Systems, V 2.0. Imperial College Research Report DoC, 2000
20. Gordon, R.: Essential Java Native Interface. Prentice Hall (1998)
21. Sun Microsystems - Java Virtual Machine Profiler Interface (JVMPI), http://java.sun.com/ products/jdk/1.3/docs/guide/jvmpi/jvmpi.html