Providing Authentication & Authorization Mechanisms for Active Service Charging[1]

Marcelo Bagnulo[1], Bernardo Alarcos[2], María Calderón[3], Marifeli Sedano[4]

[1,3] Departamento de Ingeniería Telemática. Universidad Carlos III de Madrid.
Av. Universidad 30 - 28911 LEGANES (MADRID)
[2,4] Área de Ingeniería Telemática, Universidad de Alcalá de Henares
28871 Alcalá de Henares (MADRID)
{marcelo,maria}@it.uc3m.es, {bernardo,marifeli}@aut.alcala.es

Abstract. *Active network technology enables fast deployment of new network services tailored to the specific needs of end users, among others features. Nevertheless proper charging for these new added value services require suitable authentication and authorization mechanisms. In this article we describe a security architecture for SARA (Simple Active Router-Assistant) architecture, an active network platform deployed in the context of the IST-GCAP project. The proposed solution provides all the required security features, and it also grants proper scalability of the overall system, by using a distributed key-generation algorithm.*

1. Introduction

Network services provision versatility has been dramatically improved by the introduction of active networking [1]. This technology provides network nodes with dynamic programmability capabilities, enabling the provision of customized services in a per customer basis, thus allowing clients to select specific services to be used when coursing its traffic. Since these services provide an added value to the users and its provision can imply additional costs to the operator, new charging mechanisms compatibles with this new service dynamic must be deployed as well. This charging mechanisms need that proper authentication and authorization guarantees be provided by the active network security architecture. However, heavy security can preclude deployment in real scenarios because of the imposed overhead in terms of processing, bandwidth and/or latency. So, in order to achieve a deployable active network architecture, the security solution must not only provide the guarantees needed for the charging system but it must also grant the scalability of the system. In this article, we will present the authentication and authorization mechanisms needed to provide an suitable charging system for active networks based on the SARA platform which can fulfill both requirements thanks to a distributed key-generation algorithm and to architectural features of the SARA platform.

[1] This work has been funded by CICYT under project AURAS.

International workshop on Internet Charging and QoS Technologies (ICQT'2002). Zurich, Switzerland. October, 2002. Lecture Notes in Computer Science, nº 2511. pp.337-346. ISSN 0302-9743.

The remainder of this article is structured as follows: in section 2 an introduction to SARA is presented, along with a description of its active packet exchanges. In section 3, the security solution requirements are detailed, including threats assessment and scalability requirements. Next, in section 4, the security architecture is described. In section 5, related work is presented and finally, section 6 is devoted to conclusions.

2. About Active Networks

There is clear trend towards extending the set of functions that network routers support beyond the traditional forwarding service. Active network technology aims to allow intermediate routers to perform computations up to the application layer and therefore making network more intelligent. Besides, this technology supports the deployment and execution on-the-fly of new active services, without interrupting the network operation. In this way, an active network is in the position to offer dynamically customized network services to customers/users.

This dynamic network programmability can be conceived by two different approaches. Some active networks platforms follow a discrete approach. This mean, packets don't include the code to be executed in the active routers, but exist a separate mechanism for injecting programs intro an active router. Usually this download is done from a server code or other system with the responsibility of storing the code. Others follow a integrated approach, and packets (called capsules) include not only user data but the code for the forwarding of the own packet as well. This code is then executed at the routers, or switches, as the packet propagates through the network.

Potential advantages of active networking include the opening up of the network to third parties, the easy introduction of sophisticated and unanticipated network services, and significant speedups in the deployment of such services.

2.1. About SARA

SARA (Simple Active Router Assistant) [2] is an active router prototype developed in the context of the IST project GCAP [3]. It is based on the router-assistant paradigm, meaning that active code does not run directly on the router processor but on a different device, called assistant, which is directly attached to the router through a high-speed LAN. Hence, the router only has to identify and divert active packets to its assistant. Active packets are identified by the router alert option, enabling active router location transparency, since active packets need not to be addressed to the active router in order to be processed by it. After requested processing is performed by the assistant, the packets are returned to the router in order to be forwarded. The active code needed to process active packets is dynamically downloaded from Code Servers when it is not locally available in the assistant. In this way safety is checked in advance, since only registered harmless-proofed code is allowed to run on the network. Thus the presumed target scenario is one in which a central administration provides active services loaded on the fly from a choice of known applications that have been provided by the customer or network manager.

SARA is available in two platforms: One fully based on linux [2] (playing both roles: router and assistant as a development scenario) and a hybrid platform where the router used is an Ericsson-Telebit AXI462 running a kernel adapted to work with an active assistant.

2.1.1. SARA Packet Exchanges

We will next introduce the packet exchanges performed, so we can detect the authentication and authorization requirements.

2.1.1.1. Elements involved in the packet exchange

Source: User terminal that generates traffic and uses the active features of the network.

Destination: It is the terminal that *Source* addresses its traffic to.

Active network operator: provides network services and additional active services. There are two main elements in the active network:

Active Router: It is an active router (router plus assistant) capable of processing active packets. It is also able of obtaining the active code needed.

Code Server: It is the active code repository that serves the *Active routers*.

2.1.1.2. Packet exchange description

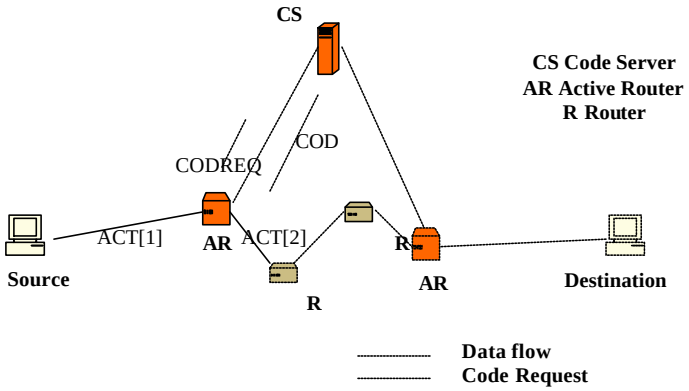

Figure 1. Services Network Architecture.

When *Source* needs special active processing for a flow of packets between itself and *Destination*, it must send packets (ACT[1] in the Figure 1), addressed to *Destination*, containing the Router Alert option and the identification of the active code that it desires to be executed. When this packet reaches the first *Active Router*, it is inspected and the identification of the active code is extracted. If the Active Code is locally available at the *Active Router*, it performs the requested process and then forwards the packet (ACT[2] in the Figure 1). If the needed active code is not locally available the

Active Router requests it to the *Code Server* (CORREQ in the Figure 1). The *Code Server* then sends the requested code to the *Active Router* ([COD] in the Figure 1), which now processes the packet and forwards it to the next hop. The same procedure is executed by all the *Active Routers* along the path, until the packet reaches *Destination*, where the packet is received. Next active packets of this flow will presumably follow the same path, so the *Active Routers* will be capable of processing them without needing to request the code from *Code Servers* again.

3. Security Architecture Requirements

In this section we will present the different requirements imposed to the security architecture. Charging system imposes the need for authentication and authorizations of , i.e. it must be possible to verify that the user that is requesting the code (*Source*) is authorized to executing it at this moment. In addition, the service request must be provided in a non repudiable fashion, since it is considered as an asset when charging is involved. However, it must be noted that non repudiation features are usually expensive, because they require the usage of public key cryptography. Besides, other security issues impose additional requirements that will be presented next. Finally we will describe other general requirements, specially emphasizing in scalability aspects that also have great impact in the final solution.

3.1. Additional Security Requirements

In order to perform an exhaustive analysis, we will retrieve the security requirements from each elements´ perspective.

From the *Active Router´s* perspective, it is relevant that the active code loaded into the routers is provided by an authorized *Code Server* and not from an unauthorized one. Besides, the code integrity must be preserved while it is transmitted from the *Code Server* to the *Active Router*.

From the *Code Server´s* perspective, it must be able to authenticate *Active Routers* that are requesting active code, since not all the code will be available to all routers. Furthermore, the security solution must provide confidential code transfer, in order to prevent unauthorized parts to inspect the delivered code.

From the *Source´s* perspective, it must be able to be certain no other user is requesting active services on its behalf, so that it is only charged for the services that it has requested. It must also be the only one capable to control its active services, meaning that no other user is capable of introduce new active packets or modify active packets sent by *Source*, interfering with the requested active services.

From the *Destination´s* perspective, there are no requirements since it does not demand active services from the network, it just receives packets sent using them. In case that *Destination* would be interest in answering this packets using also active services, it would become *Source* and *Source´s* requirements would apply. It should be noted that end-to-end security is out of the scope of this security solution.

3.2. Other General Requirements

Zero user knowledge at the *Active Routers:* In order to build a manageable solution, user management must not be performed on each and every *Active Router*. A central database containing all the users information, including access rights would be the preferred solution.

Active Router transparency: It must not be required that *Source* be aware of which *Active Routers* are in the path used by the network to transport packet towards destination. This means that active packets sent by *Source* must not be dependent of which *Active Routers* are addressed to.

4. Security Architecture

4.1. *Source* Authorization

4.1.1. Authorization Paradigm

The key feature that must be provided by the security architecture is authorization i.e. *Sources* must be authorized to execute the solicited code on *Active Routers*. There are two authorization paradigms that can be used: authorization based in access control lists or authorization based in credentials. The first paradigm is based on the existence of an access control list (locally available or in a remote location) that must be queried every time an *Active Router* receives an active packet sent by *Source*, in order to validate the *Source´*s permissions. In this case the identity of requesting part must be authenticated in order to prevent impersonation. This approach then requires that the requested device (*Active Router)* has information about *Sources* and permissions or it imposes a communication with an authorization server every time a *Source* sends an active packet. The second paradigm demands that every time *Source* sends a packets, a credential that proves the *Source´*s permissions must be presented. Then the requested device (*Active Router*) only needs to verify the credential. However, credential generation and distribution may be more than a trivial task.

The solution proposed in this paper will be designed based on the second paradigm, since we consider that it provides better scalability attributes. Note that since the access control list can not reside on every *Active Router*, because of the Zero user knowledge requirement, it must reside in a remote location, imposing a remote query every time a *Source* need to be authorized.

4.1.2. Public Key Cryptography vs. Symmetric Key Cryptography

In order to allow the intended use, a credential must contain verifiable authorization information i.e. the permissions granted to the holder of the credential. Besides, it must be possible to verify that the issuer of the credential has the authority to grant these permissions, (it will be called a valid issuer). It is also critical to

validate that the user that is presenting the credential is the same user that the credential was granted to.

In order to fulfill the above stated characteristics of a credential, public key encryption can be used. So, a credential containing the *Source´*s permission and the *Source´*s public key is signed with the private key of the valid issuer. Then, the *Active Router* must be capable of verifying the authenticity of the credential, using the valid issuer public key, and also it must be capable of verifying that the requesting user has the private key that corresponds to the public key included in the credential. Even though this mechanism provides all the required features, the usage of public key cryptography is very demanding in term of processing, specially when considering that for every active packet, two public key signatures must be verified.

In order to obtain a less demanding solution, symmetric key cryptography can be used. However, building a similar system using symmetric key would require the usage of two different symmetric keys (a first one shared by the valid issuer and the *Active Routers* and another key shared by *Source* and the *Active Routers*). This system would still demand for two signature verifications and it would present the additional problem of key distribution. So, in order to improve the scalability of the solution, we will next explore the possibility of using only one symmetric key, shared by the valid issuer, *Source* and the *Active Routers*.

The requirements imposed to this key are:

? Different keys for different *Sources*. (i.e. the key must be linked to a *Source*)
? Different keys for the same *Source* at different moments (i.e. the key must have a validity period)
? Different keys for different active codes by the same *Source* (i.e. the key must be linked to a service)

So, the key issued by the valid issuer is linked to a *Source*, a code and a validity period.

Then, if this key is used to sign (HMAC [4] signature) an active packet that request the execution of a particular active code, the active packet itself plays the role of a credential. Basically, an *Active Router* receives a signed active packet that includes the requested code identification, the *Source* and the time when the active packet was generated. Then if the *Active Router* has a valid key linked to the *Source* and the requested code, it can verify the authenticity of the active packet, without any further information. This mechanism imposes the usage of an A*uthorization Server* (the valid issuer role), that generates the keys. So, in order to execute a code in the network, *Source* must obtain the correspondent key from the *Authorization Server* in a secure way. This is not a time critical task, since it is only performed when the service is requested and it is possible to execute it in advance . However, once the service is authorized and the key is generated, the *Authorization Server* must communicate it to all *Active Routers* in the network, so they are aware of the new authorization. This is does not seems to be the most scalable solution, because of the amount of communications needed between the A*uthorization Server* and the *Active Routers*.

We will next present an improved solution that minimizes the required interaction between this elements. The basic idea is that the key can be almost autonomously generated in every *Active Router* when it is needed. In order to achieve this, we will associate a key to every Active Code that can be loaded in the *Active Routers*. The

key associated with active code Ci is called Kci. These keys are known by the *Code Server* and by the *Authorization Server*. Then, when a *Source* S requests authorization for the execution of code Ci at a moment T and for a period P, the *Authorization Server* generates the key K as the HMAC of the concatenation of Kci, S, T and P. The key K is then transmitted to the *Source*, so it can sign the active packet with it. If we analyze the characteristics of K we can see that: K is linked to an active code (Kci); K is linked to a *Source*(S); K has a validity period (T, T+P); K can not be generated by any *Source*, since they do not have Kci. However, if the *Code server* confidentially transmits the active code [COD] it also attaches Kci, then the *Active Routers* are capable of regenerating K without contacting the *Authorization Server* every time an active packet arrives or when a new *Source* requests an already downloaded code. The *Active Routers* have all the information needed to generate K, i.e. S, T, P and Ci are included in the service request and Kci is obtained when they download the code from de *Code Server*. Note that the solution is based on shared secret keys, so the security level of the solution can be defined by setting the number of parts share the keys Kci (authorized *Active Routers* for a given code Ci) and the frequency Kci are changed.

4.2. Code Downloading

Another key feature that must also be provided is a secure way to download code (and keys Kci) from the *Code Server* into the *Active Routers*. However, this is not as time critical as user authorization since it is only performed once, when the first packet arrives. The following packets will benefit from a cached copy of the code and the Kci. So, a protocol that allows a secure communication between two parts is needed. We will use TLS [5] since it provides all the needed features. However, the usage a new protocol specifically designed for this task would result in improved efficiency, since TLS is a generic protocol. Then the *Code Server* must have a digital certificate (public key cryptography is used), and a TLS session is established between the *Code Server* and the *Active Router*, before the code is downloaded. The *Active Router* does not need a digital certificate, since non repudiation features are not required. Its authentication can be performed using a user and password, transmitted through the TLS session.

4.3. Non Repudiation

Since charging is involved when the user requests a service non repudiation features must be provided. In order to assure non repudiation, public key cryptography must be used when the user request authorization to the *Authorization Server*, as it will be described in the following section. However, our solution will not use public key cryptography in the credentials because of performance issues. Summarizing, the proposed solution provides non repudiation features when the service is requested to the *Authorization Server*, but it does not provides them when active packets are processed by *Active Routers*; this is tradeoff between performance and security features that we consider acceptable for most scenarios.

4.4. The Security Solution: Step by Step

In this section we will describe the complete mechanism which is illustrated in figure 2.

First (step 1 in figure 2) , *Source* requests authorization (to the *Authorization Server*) to execute an active code Ci in the network. This request is done in a secure way, meaning that public key cryptography and digital certificates are used by both parts. So, *Source´*s request is signed with the private key of *Source* and its digital certificate is also included. This request is encrypted with the public key of the *Authorization Server*. Then the *Authorization Server* after receiving and verifying the request, it generates K as the HMAC of the *Source´*s identification (S), the key associated to the requested code (Kci), the moment of the request (T) and the validity period requested by Source (P). Then the *Authorization Server* sends a signed message containing K. The message is encrypted with the public key of *Source*. At this moment, the *Authorization Server* has all the needed information for charging, since it has the requested service, the user, authenticated in a non repudiable way, and the time that this service was requested.

Source decrypts the message and obtains K. Then (step 2 in figure 2), it generates active packets , that includes its own identification S, the moment of the request T, the validity period P and an identifier of the solicited active code Ci. This message is signed performing a HMAC of the message plus K.

When an *Active Router* receives active packets, it first verifies that the packet is not obsolete, i.e. it is within the validity period and then it verifies the solicited active code availability. In case the code (and Kci) is not locally available, it downloads it, using a secure (TLS) connection from the *Code Server* (step 3 in figure 2). Then the *Active Router* generates K, using S, T and P extracted from the packet and Kci obtained from the *Code Server* when the code was downloaded. If the HMAC signature is verified, it means that *Source* has been authorized to execute the requested code, so it processes the packet using the solicited code and forwards it to the next hop. The same procedure is repeated on every *Active Router* along the path until the packet reaches *Destination*. The following active packets of the flow will benefit from cached copies of the active code and Kci in every *Active Router*.

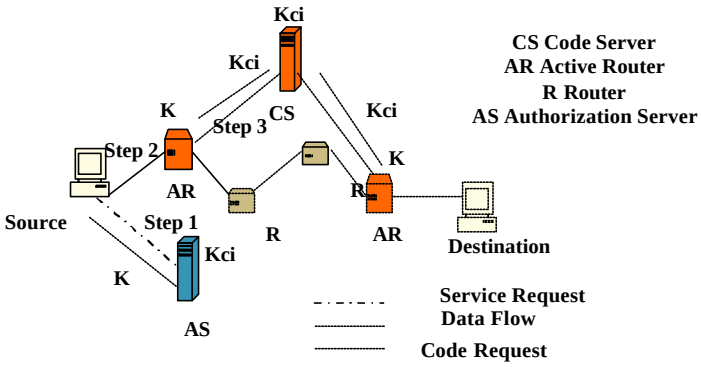

Figure 2. Key Distribution.

5. Related Work

There are many proposals for charging and accounting in a Multiservice Network [6], but few works have addressed the problem of the security mechanisms related to charging. One of these proposals has been done in the context of the CATI project [7].This IST project aims to investigate mechanisms of charging based on accounting QoS-based end-to-end communications. In CATI is proposed a security architecture for a scenario of multiple ISPs, an E-Commerce Service Provider, a customer, and a payment provider.

On the other hand, some proposals consider charging and accounting as an active application more. AIACE[8] and PEACH [9] explore the idea of using active services in order to enable charging and accounting system more scalable and flexible. In AIACE accounting active services are loaded on-the-fly and perform accounting tasks on behalf of accounting servers. PEACH is similar in many aspects to AIACE but focus more on the fast change in charging logic to reflects changes in business policies.

But to the best of our knowledge, FAIN [10] is the only active network platform previous to SARA that considers the charging for the use of active services. However, the scope of FAIN security architecture is limited mainly to protecting Active Network infrastructure from users and active code.

6. Conclusions

We have presented a security solution that provides authentication and authorization services for active networks based on the SARA platform. These features allow proper charging for active services. Futhermore, the solution performance is guaranteed by the usage of symmetric key cryptography. The scalability of the solution is assured by the authorization model, based on credentials, and the key distribution mechanism, that minimizes key exchanges by allowing key generation at every *Active Router* in an autonomous fashion. The security level of the solution is determined by the re-keying frequency i.e. how often Kci are changed. Then, we conclude that this solution enables the deployment charging mechanisms for SARA in a public testbed. Finally, it should be noted that it is possible to extend this architecture to other active network platforms as long as a central active code repository exists in the other platform.

References

1. David Wetherall, Ulana Legedza y John Guttag, “Introducing new Internet services: Why and How”, IEEE Network Magazine [1998]
2. SARA home site. http://matrix.it.uc3m.es/~sara.
3. GCAP IST project home page. http://www.laas.fr/GCAP
4. Krawczyk H., M. Bellare, R. Canetti, HMAC: Keyed-Hashing for Message Authentication, RFC 2104, April 1997.
5. Alan O. Freier, Philip Karlton, Paul C. Kocher. TLS Working Group. The SSL protocol version 3.0. (November 18, 1996).

6. M. Falkner, M. Devetsikiotis, and I. Lambadaris, An Overview of Pricing Concepts for Broadband IP Networks, IEEE Communications Review, Vol. 3, No. 2, 2000.
7. B. Stiller, T. Braun, M. Günter, B. Plattner: "The CATI Project: Charging and Accounting Technology for the Internet", 5th European Conference on Multimedia Applications, Services, and Techniques (ECMAST'99), Madrid, Spain, May 26-28, 1999, LNCS, Springer Verlag, Heidelberg, Vol. 1629, pp 281-296.
8. Franco Travostino . "Towards an Active IP Accounting Infrastructure", Proc. 3rd IEEE Conference on Open Architectures and Network Programming, OPENARCH 2000, Tel Aviv, Israel, March 2000.
9. Lee, B. & O'Mahony,D., A Programmable Approach to Resource Charging in Multi-Service Networks, Proceeding of the IEEE International Conference on Software, Telecommunications and Computer Networks (SoftCoM 2000), October 10-14, 2000, Split, Croatia.
10. FAIN project (IST-1999-10561-FAIN) Deliverable Initial Active Network and Active Node Architecture, CEC Deliverable Nr: D2, editor: Spyros Denazism May 2001