

Polymorphic Services for Secure and Context-dependent VHEs

Carsten Link and Norbert Luttenberger

Inst. f. Informatik u. Prakt. Mathematik
Christian-Albrechts-Universität Kiel
{cli,nl}@informatik.uni-kiel.de

Abstract. Network services can be classified as public services or private services. Public services are offered to the anonymous public in the public part of the Internet, whereas private services are provided exclusively to selected user groups. Offering private services outside a secure intranet environment may constitute a security risk to the service provider. In this paper, we propose to increase security by using dynamically instantiated proxies that base their authorization decision for granting access to services on context information supplied by different sources. We call these proxies "polymorphic proxies". From the users' point of view, the resulting global behavior is that of a "polymorphic service": Though well-defined and stable service interfaces are used, the service itself may show a situation-dependent behavior. We present the concept of *Virtual Service Networks*, in which polymorphic services are embedded, giving the service provider distinct control over service usage.

1 Introduction

In today's IT environments, users rely heavily on the accessibility of a large number of network services. These include various kinds of communications services (e-mail, WWW, etc.); database and business application services (with different front-ends); and other services (e.g. telnet, ftp). It is the wish of users to have these services at their fingertips *anytime, anywhere, anyhow*.

With regard to service providers, network services can be classified as either public services or private services. Public services are offered to the anonymous public in the public part of the Internet, whereas private services are provided by organizations, companies etc, exclusively to selected user groups inside their intranets. Private services may in many cases constitute an important part of the *Virtual Home Environment* (VHE) for certain types of users. For instance, "nomadic" users away from their home location appreciate the integration of private services into their VHE, as do virtual enterprise staff when working in different intranet environments. It is the aim of this paper to outline a concept for resolving the conflict between user demands, on the one hand, and provider security constraints on the other. We propose the two following measures:

- Private services should generally be invisible until users are authenticated. Invisible services are not exposed to attacks.

- When—after authentication—a user is going to be authorized to use a private service, the user's actual situation should be taken into account. "Situation" includes, for instance, place, time, network attachment, and so on. For example: it may not be acceptable to grant access to an ftp-server's confidential section when the requesting user is in a public Internet cafe, but the same access request is granted when the user is in the intranet of a cooperating company.

The second point implies a strict separation between authentication and authorization. This strict separation is founded on a novel concept for providing network services which we call *polymorphic services*. Polymorphic services are embedded in *Virtual Service Networks* (VSNs), a term denoting an infrastructure for secure and context-dependent VHEs.

The paper is organized as follows: In the following section, the general concept behind polymorphic services is explained. Then, these services are presented in more detail, including some implementation aspects. In Section 4, the definition and structure of VSNs is elaborated. In Section 5, approaches with a similar background are analyzed. The paper closes with an conclusions.

2 Concept

Today, it is common practice to install firewall systems at the border between public Internet and private intranet to control incoming and outgoing traffic. Controlling is done at the packet/transport layer (packet filters) or/and at the services layer (application level gateways). To restrict the access to private services, it is preferable to install traffic control at the services layer for the following reasons:

- In distributed systems, subjects get access to (information) objects solely via network services. Therefore the services layer is the "natural" point of control.
- A general definition of what "secure" means cannot be given. Instead, security must be defined specifically for each service and its individually related objects.
- Services layer traffic control can be more easily aligned with an authorization procedure that considers the requesting subject's actual situation. This enables the implementation of the least-privilege-principle as promoted already in 1975 by Saltzer and Schroeder [6].

"Classical" application level gateways are statically installed and configured proxy servers that do not consider the requesting subject's actual situation. In this paper, we propose to use dynamically instantiated proxies instead, that (partly) base their authorization decision on context information as supplied by different sources. We call these proxies "polymorphic proxies". From the users' point of view, the resulting global behavior is that of a "polymorphic service": Though well-defined and stable service interfaces are used, the service itself may show a context-dependent behavior.

Before offering such a service, the service provider must devise a service-specific policy defining the "who" and "how" for access requests to private services. If, for instance, the communications channel to the requesting subject is exposed to eavesdropping and the information to be conveyed via this channel is confidential, an appropriate

policy would not allow access to this information, unless the client has cryptographic capabilities.

A set of polymorphic services as seen by the user is bundled into a so-called Virtual Service Network (VSN). A VSN is constituted by a number of cooperating VSN gateways each of which is located at the border between the public Internet and a private intranet. To the user, a VSN gateway is the host for a certain subset of polymorphic services. A VSN does not show its service offerings until the requesting user has been authenticated.

Figure 1 shows a VSN in which a user accesses polymorphic services as provided by different intranets. The Venetian masks symbolize polymorphism: Though to the user it always appears to be the same service, behind the masks different instances are hidden.

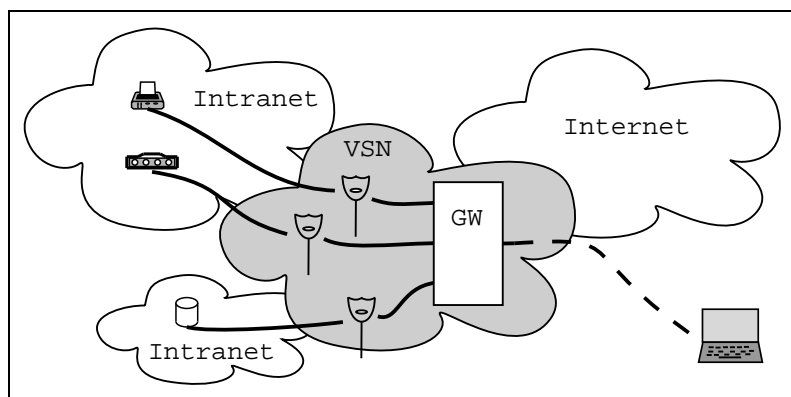


Fig. 1. A Virtual Service Network. The box marked with "GW" stands for a group of VSN gateways constituting the VSN.

The user's situation is determined by different kinds of context information. Our prototype currently uses the users topological location, time, and properties of the client node's connection to the VSN. This context information is determined by the VSN gateway to which the user is connected. The gathered information can be extended to include the client node's abilities and, for example, the general system state determined by an Intrusion Detection System. This would allow only basic access, in case of a vulnerable system state.

3 Polymorphic Services

As noted in Section 2, network services are accessible as polymorphic services. These are composed by inserting a polymorphic proxy between the client and the back-end server. The following sections explain polymorphic proxies.

3.1 Polymorphic Proxies as Capabilities

Due to their properties polymorphic services can be seen as capabilities. A capability is an unforgeable ticket containing a reference to an object and access rights associated with it. If an object is accessed using a capability, it is only necessary to verify if the requested access conforms to the access rights of the capability. No further authorization is needed. Any subject (i.e. person or program) presenting a capability is authorized to access the referenced object within the limits of the access rights contained in the capability. Capabilities allow for separation of authorization and access, which allows decentralized configuration and efficient access.

In addition to the reference to an object and associated access rights *Active Capabilities* [5] contain executable code. I.e. Active Capabilities are mobile programs or objects of an object-oriented programming language, and form a capability with a built-in reference monitor. This aggregation yields several advantages:

- The reference monitor is freely programmable. Arbitrary restrictions can be expressed.
- Servers which do not implement security mechanisms can be secured. Provided that there is no other way to access a server than using a capability, it is even possible to secure servers which are not under control of the authority issuing the capabilities.
- Moving the reference monitor to the client reduces the overall load of the server. Access requests which do not conform the access rights of the capability are blocked by the reference monitor, which minimizes network traffic.

Polymorphic proxies are a generalization of active capabilities. They are realized as Java objects, which can be serialized and transmitted to a remote execution environment. Polymorphic proxies act as monitors and enforce the context-dependent security objectives of the service provider.

3.2 Polymorphic Proxy Embedding

Within a VSN (see section 4.1) services are rendered by mobile Java objects. Using a Web-Browser, the user can activate polymorphic Services on the Portal of the VSN. A polymorphic proxy is then dynamically injected into the execution environment of the VSN server. As part of the activation process, a proxy interacts with the execution environment and registers itself as an IP, TCP or UDP service. An electronic mail service proxy would register itself as a TCP service utilizing the ports 25 and 110 to supply POP and SMTP to the user. The interface of the execution environment, which can be used by the proxy, has the following methods:

```
interface ExecutionEnvironment {
    XMLDocument description();
    GUIHandle    userInterface();
    void         registerIPService( PolymorphicProxy, Protocol);
    void         registerUDPService(PolymorphicProxy, Port);
    void         registerTCPService(PolymorphicProxy, Port);
    void         sendData(Address, PDU) throws IOException; }
```

The execution environment provides methods allowing the proxy registration, communication, and user interaction. Polymorphic services can be implemented in a very flexible manner, since proxies can register themselves at network or transport layer:

- Autonomous service: The proxy contains all data and functions necessary to perform the required service.
- Personalized service: The appearance and the available functionality is tailored to the needs of the user. Individual preferences and certain user characteristics (e.g. age, spoken language, visual acuity) can be considered.
- Protocol conversion: The proxy uses different protocols to communicate to the client and the server to which it provides access.
- Redundant service: The proxy uses more than one back-end server to improve availability or performance.
- Decoupling service: The proxy acts in place of a weak-connected client, which may temporarily not be connected to the network.
- Forwarding service: The proxy securely forwards IP, UDP or TCP data.
- Simulation service: For simulation purposes the proxy can modify, generate, duplicate, log, or drop data.

This list shows that polymorphic proxies are a powerful means to implement security-related measures. Section 4.1 describes how polymorphic proxies are supplied by the provider and activated depending on context information.

3.3 Sample Policy

A policy defines rules which determine which users can access which services, and under what circumstances. A policy can be as follows:

```
pattern officeHours
    time after 08:00 and time before 18:00
group employees@komsys members (
    cli@komsys, flr@komsys, mab@komsys )
authcrypt fileserver://komsys
    for employees@komsys
    restrict officeHours
deny *://*
```

To allow a context-dependent authorization decision, the rules take time and the users' current location into account. The above example allows members of the group "employees" encrypted access to a file server after successful authentication of the user. The access is restricted to office hours. Time and location are examples of context information that determine a decision.

4 Virtual Service Networks

4.1 VSN Server

A VSN is provided by a VSN server which runs on a VSN gateway. A VSN server consist of a portal, a policy data base, a polymorphic proxy pool, and an execution environment (see figure 2).

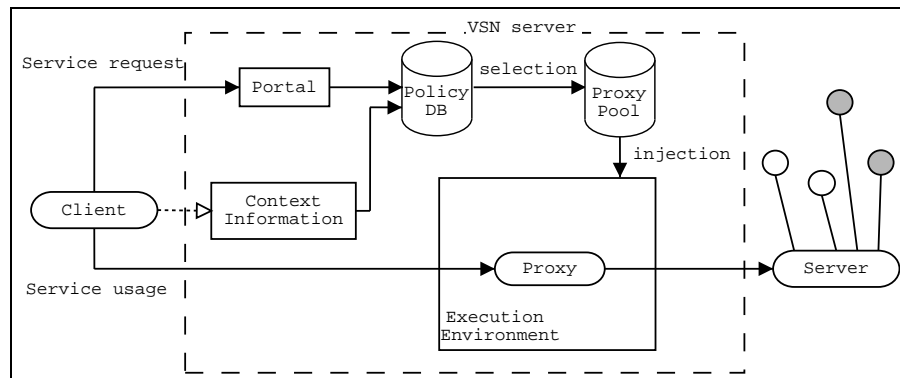


Fig. 2. The VSN server allowing controlled access to private services. Using polymorphic proxies for service access only a part of the objects accessible through the server is visible (marked gray).

If a user requests a service to be started, the VSN server evaluates rules stored in the policy database, in conjunction with the gathered context information, to select an appropriate proxy from the proxy pool. The selected proxy is then injected into the execution environment. Once embedded there, the proxy interacts with the execution environment to establish itself as a service. Polymorphic proxies implement the following interface:

```
interface PolymorphicProxy {
    PolymorphicProxy(Name, Portal);
    void attachTo(ExecutionEnvironment);
    void detach();
    void start(String Username, String VSNName);
    void stop();
    void suspend();
    void resume();
    XMLDocument description();
    void handleData(PDU);
    PDU nextPDU(); }

```

During injection the execution environment invokes the `attachTo(this)` method to allow the proxy to initialize and register itself. The execution environment uses the methods `handleData()` and `nextPDU()` to allow the embedded proxy to communicate with its client. These methods are generic, since it is possible for a proxy to supply services at different layers (e.g. network or transport layer).

The VSN server provides a web server for every user connected to the VSN. Using a standard Java-enabled browser, users can access their individual graphical web interface to the services available to them. The web interface allows for viewing of system messages, and basic configuration of services. If a service requires additional authentication it can use the browser's mechanisms to ask the user for the necessary information.

4.2 VSN "Dial-In"

A VSN is provided by at least one VSN gateway to which users can connect. Co-operating VSN gateways use the Jini [4] technology for internal communication and distribution of information and computation. Before being able to use services within a VSN, the user needs to connect to a VSN gateway using a PPP protocol. The supported PPP protocols PPTP and L2TP have the following advantages:

- Common operating systems include support for these protocols.
- The node is transparently connected to the VSN at the network layer.
- Authentication and encryption are supported.
- PPTP and L2TP use UDP for transport, which allows easy firewall traversal (See the proposed meet-in-the-middle concept in [3]).

The VSN gateway comprises a PPTP or L2TP server, permitting the client nodes to connect to a VSN. IP traffic is not routed by the kernel, but instead is forwarded to the VSN server.

5 Other Approaches

The following section describes two other approaches. Both projects are related to our proposal, because they build virtual networks upon an existing network infrastructure.

Delgrossi and Ferrari propose virtual networks which they call *supranets* [2]. Using a toolkit, supranets are created, deployed, managed and deleted by their users. A supranet constitutes a secure environment for its users. To virtualize the network, an additional abstraction layer is inserted into the protocol stack above the network layer. A supranet uses its own name space, which is translated into IP addresses by the inserted layer. The system is transparent to existing TCP/IP applications, because the users are connected at network layer. To fully exploit the supranets functionality, it is necessary to modify or redevelop the applications.

In contrast to VSNs, the implemented security mechanisms of supranets are restricted to 1) access control to the supranet and 2) cryptographically secured communication. Since supranets build a transparent virtual network no context-dependent policy-driven authorization is available.

A similar approach is pursued by a research group at Sun Microsystems. Caronni et al. develop *supernets* or *Virtual Enterprise Networks* [1] by introducing an intermediate layer on top of IP. A supernet aggregates network nodes, placing them under a single administrative domain, and giving them a network address space. Nodes within a supernet are connected by *channels* acting as an encrypted shared medium. A node is identified by an IP address. In supernets even a process can be a node. Before a node becomes visible in a supernet, it must successfully register at the admission control server.

Although this approach is similar to our proposal, there are some differences. After passing the admission control, any other node within the supernet is accessible. No further access control is conducted. Furthermore any participating host needs to have a modified IP stack. Supernets have no concept like polymorphic services.

6 Conclusions

This paper motivates and proposes a novel approach to enhancing the service providers' control over service access and usage, aimed at more secure services. A prototype is under construction and will be shown at the CeBit 2002 in Hanover. There is some further research to be done. We are investigating how to make classical network services polymorphic in an administration-friendly manner. The cooperation of multiple VSN gateways is being investigated as are other sources of context information to be considered for authorization decisions.

References

1. G. Caronni, S. Kumar, Ch. Schuba, and G. Scott. Supernetworking: The next generation of secure enterprise networking. In *ACSAC: Annual Computer Security Applications Conference*, 2000.
2. Luca Delgrossi and Domenico Ferrari. Internet-based secure virtual networks. In *SYBEN*, pages 318–326, 1998.
3. Atsushi Kara. Secure remote access from office to home. *IEEE Communications*, 39(10):68–72, October 2001.
4. Sun Microsystems. <http://www.sun.com/jini>.
5. Tin Quian. Cherubim agent based dynamic security architecture. Technical report, University of Illinois at Urbana-Champaign, June 1998.
6. Jerome H. Saltzer and Michael D. Schroeder. The protection of information in computer systems. *Proceedings of the IEEE*, 63(9):1278–1308, September 1975.