History Based Distributed Filtering – A Tagging Approach to Network-Level Access Control

Reiner Sailer, IBM T. J. Watson Research Center, sailer@watson.ibm.com
Matthias Kabatnik, University of Stuttgart, kabatnik@ind.uni-stuttgart.de

Abstract

This contribution discusses a network-level access control technique that applies the non-discretionary access control model to individual data packets that are exchanged between hosts or subnets. The proposed technique examines incoming data's integrity properties to prevent applications within a node or subnetwork from so called subversive channels. It checks outgoing data's secrecy requirements before transmission. Security labels are used to identify data packets as members of different categories and security levels. Additional tags store context information to validate the trustworthiness of a packet's content. Labels and tags of a data packet reflect events that may be relevant to access control throughout its life. As opposed to stateful filtering, which is based on the history of a flow of packets, our approach works on the history of an individual packet. Any state information is part of the packet rather than stored in all the nodes inspecting the packet; i.e. nodes do not need to create and maintain state information.

1 Introduction

Emerging services in the Internet and in telephony networks are increasingly vulnerable to attackers that send invalid or unauthorized data to service control processes. Furthermore, firewall and application configurations are very complex and change often. Large companies must assume that users willingly release sensitive data via network services. Hence, enforcing access control[1] independently of users and applications is indispensable.

Basically, we distinguish *secrecy, integrity, and availability* requirements. We protect secrecy and integrity by applying network-level access control to data packets. The history based approach aims at packets carrying their own history and exploits this history for access control.

One could argue that the above problems are easily solved by applying strict *discretionary access control* at the application level. Discretionary access control schemes were first formalized by Lampson in 1971 [15] and refined later. They state access rights for each subject (accessing entity) concerning any object that can be accessed. In the discretionary scheme, the owner of an object usually can change the access control information. Therefore, access control is on the discretion of the user. A misbehaving process can subvert discretionary access control schemes by changing access rights of an object on behalf of an authorized user ([18]; these so called Trojan horse attacks were identified first by D. J. Edwards). Thus, this scheme is not effective against dishonest users or Trojan horses on the host where objects are stored or processed (e.g. data bases).

In the Internet, millions of hosts are registered. Emerging Internet services are offered to mobile users using remote access points to connect to the company's servers. In this environment, discretionary access control on user or application level seems not to be the solution – even if end-to-end protection (e.g. Transport Layer Security [19]) is applied – as one cannot be sure what happens on the host of an authorized user with possibly sensitive information.

We propose to complement existing access control mechanisms at the user level by *non-discretionary (mandatory) access control mechanisms at the network level*. Mandatory access control can be implemented by so called lattices. Lattices consist of classes that form a partial order. Lattice models are described in [7] and have served since then as the basic access control mechanism in military environments. Lipner [27] and Denning [8] have shown that for lattice models, it is possible to solve the confinement problem, i.e. the leakage of information from higher to lower secrecy classes. In this model, objects are assigned classes and subjects are assigned clearance. A subject cannot change the class of an object, i.e. cannot grant access to objects to other subjects with lower clearance [12]. Lattice models evolved from military classification schemes to enforce security policies. As subjects cannot change an object's classifications, lattice models can be designed to resist Trojan horse attacks concerning information leakage.

1.1 The problem

The exemplary network configuration depicted in Fig. 1 shows a company network that has three major branch office networks A, C, and D. These include further subnet-

1 See Lampson [15], Karger [12], Sandhu [22] for a more thorough discussion of access control principles.

works (e.g. B). The branch office networks are connected directly with each other via internal links and indirectly via the insecure Internet. Remote users access the company's network via remote access servers (RAS). Routers are located at those points where a link meets a subnetwork.

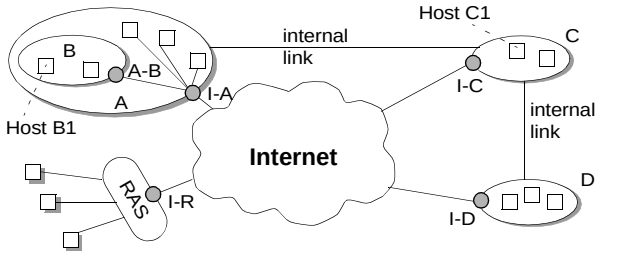

Figure 1: Exemplary scenario

Packets from company networks C and D are allowed to pass into network A. Hosts in subnetwork B need better protection. Therefore, only internal traffic from network A hosts or IPSec [17] protected traffic from network C hosts via internal links are allowed to enter B.

Conventional filtering demands respective filters to be deployed at subnet A's entrance. Therefore, any changes in the security policy of subnet B will lead to changes in subnet A's filtering rules. Additional filters could be applied at the subnetwork boundary of B; but at that point it is not clear, whether traffic from a particular network C host has been protected between A and the host.

History based filtering allows network operators to enforce, e.g., following policies at subnet B's edge-routers:

- A-internal traffic in/out is allowed
- any company internal traffic (A/C) in/out is allowed, if protected by IPSec on A-C or via internal links only
- RAS-internal traffic protected by IPSec is allowed in/out A,C,D but is disallowed in/out B

Conventional nested filtering functions lack information that has been available at former (outer) filter stages, e.g.:

- original destination addresses if network address translation (NAT) is used; used only within the NAT domain
- authenticity of packet contents: e.g., has the packet been sent over an end-to-end IPSec tunnel with authentication?
- has the packet ever left the trusted network?
- if internal links are over-loaded: has the packet been IPSec protected when transmitted via the Internet?

1.2 Advantages of the history based approach

Distributed filtering means that access control is implemented for the local domain only. Nested subdomains and remote domains will have independent access control to enable different security levels. It follows, that:

- Filtering for large subnets is less dependent on requirements of nested subnetworks (i.e. changes less often).
- Filtering is fast for large subnets, which usually do not need highest protection.
- Packet history enables filtering closer to the subject (subnet, host) as we preserve context information.

Filtering takes longer for packets that are subject to stricter inspection, e.g. applying cryptographic operations. Thus, effective and time consuming decisions and filtering are pushed close to subjects and applied only if necessary.

Access control decisions of filters are usually based not only on the data packet "under inspection" but also on additional context information (e.g. incoming or outgoing link). This additional information is locally available and is called *implicit* information. Today, implicit information is lost after the packet leaves a filter. Succeeding filters do probably have other environmental context information.

We propose to add *implicit* information as *explicit* tags to a data packet if it is passed through an inspection point. Then, cascaded incoming and outgoing access mechanisms in successive filters build hierarchical protection domains.

As opposed to existing filtering approaches, later filtering stages can exploit local context information as well as the history of packets. Consequently, security sensitive decisions can be based on more detailed information; this leads to a finer granularity of network-level access control.

1.3 Organization

Next, we give a short overview of work related to our approach. Following that, we describe the network-level access control model. Guards for sending and receiving data packets over an access control device and lattices defined in that section implement the non-discretionary access control model. Traffic restrictions, arising due to worst case assumptions when controlling the flow of data packets, are addressed in the following section. Finally, we sketch implementation aspects and give a detailed example that illustrates how network-level access control works.

2 Related work

Filters generally decide whether a data packet that enters a module is passed through to an output link or not. Examples are filters for IP or signalling data packets or the suppression of certain in-band signals in telephony networks. Decisions for discarding data are based on criteria that are expressed by filtering rules.

In *static filters* [25], filtering rules are static, i.e. are changed by management access only. *Dynamic filters* (stateful filters [26]) apply different rules according to their internal state or change the existing rule set dynamically. The internal state depends on packets that have been processed by the filter. Unlike stateful filtering –which uses the history of related packets in a single node– our approach

uses the history of a single packet gathered throughout preceding processing stages. Both approaches are orthogonal.

Our approach aims at hierarchical filtering as opposed to the fully distributed approach of Bellovin [4], who proposes filtering in all hosts. In the very extreme, our approach can be applied *also* in the host; but then this machine needs to be trusted. End system protection might be used for very sensitive servers, e.g. e-commerce servers. Our approach deals with two kinds of history information:

- *Context information* is stored in tags that are *added* to packets (e.g. destination/origination address before network address translation, incoming link, etc.) for later use.
- *Classification information* is derived from context information (policy) and existing classification information and stored in security labels [9] added to packets. A packet can be assigned to one security class and category only.

The change of a packet's security class or category is called re-labelling [21]. FIPS PUB 188 [6] defines security labels on application and network level. Housley [3] describes a Security Label Framework for the Internet. Both contributions assume that sender and receiver agree upon a set of common labels and authentication methods to decide about the authenticity of data packets. We do not presume this; we determine the authenticity of a packet's contents by history information about the environment to which data packets have been exposed.

Our approach differs significantly from existing approaches by not assuming any kind of explicit application-level authentication of access control information. Mainly because authentication per packet seems to be too expensive (time, complexity). Consequently, we have to combine trust in certain hosts and knowledge about applied protection means on network level to infer the authenticity of a data packet's contents. This introduces the need for trust management which is discussed in detail by Blaze et. al. [1]. Trust information concerns static relations between networks, nodes, network paths, and existing protection means like logical protection (e.g. IPSec, link encryption) and physical protection (e.g. link protection, physical access control for rooms and buildings).

3 Network-level access control

Basically, access control restricts access to objects in order to preserve their classification. We rather take the opposite approach as we do not have control over the access to a packet throughout its life. We reflect the possibility that processes might have (had) access to the packet by adjusting the integrity and secrecy labels of the object. This worst-case approach usually over-classifies data packets.

In the following, we introduce a lattice model for network-level access control. In this model, explicit classification information reflects a packet's security level. The security level of a data packet is adapted to processing events that occur to this packet.

Sending a data packet implies possible information disclosure problems. *Receiving* a data packet implies possible subversion problems by processing tampered data. We control sending and receiving of data packets independently from other protection mechanisms (user access control etc.) on a host or node. Our approach enforces that data packets are exchanged between hosts only if both hosts have the appropriate security clearance according to the local security policy. Fig. 2 illustrates the rational.

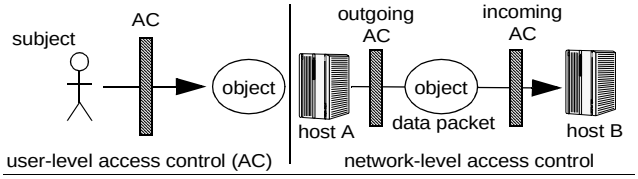

Figure 2: Mandatory access control on network level

The left side of Fig. 2 shows a user-level access control monitor controlling each reference to an object. This reference monitor decides about the grant of access and enforces access control by use of access control decision functions and access control enforcement functions.

On the right side, the network-level approach depicts objects that are rather exchanged than accessed. Objects are data packets that are exchanged between hosts A and B. The major protection goals are implemented as follows:

- An *access control mechanism for outgoing data* checks whether a data packet is allowed to be sent out to the network or not. Decisions are mainly based on secrecy policies, i.e. does the receiver have the necessary clearance to process the data packet and does the path to the receiver have the necessary secrecy clearance. Secrecy lattices are meant to prevent unauthorized disclosure channels.
- At the receiving side, an *access control mechanism for incoming data* decides whether the integrity class of incoming packets is high enough for the assumed receiver. The integrity classification shall prevent *sabotage channels* [10], i.e. protect the receiver of data packets from processing forged or tampered contents.

3.1 Network-level lattice model

General aspects of secrecy and integrity lattices are described in Karger et. al. [13]. They applied them for Smart Cards and mobile devices. We will focus on deploying lattices for network-level access control. We define following exemplary classes, clearances, and categories:

We introduce the categories *internal*, *company-internal*, *ras-internal*, and *external* to differentiate between data packets that have only been processed inside the local network, inside the company wide network, by RAS users having protected access to the company wide network, and

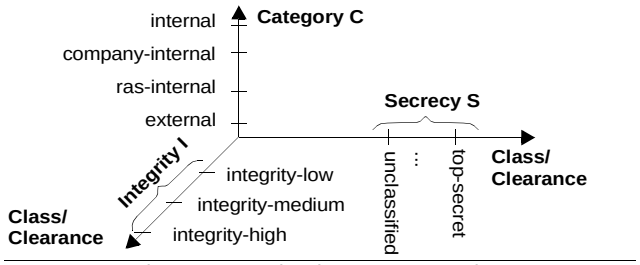

Figure 3: Security classes & categories

outside the domain. This interpretation leads to the following totally ordered set of categories:

C: external<ras-internal<company-internal<internal (1)

The category of a packet can change on its way through the network. This is reflected by re-labelling a packet as member of a new class category.[2]

Additionally, integrity and secrecy classes are introduced to further distinguish between data packets. Relations for secrecy and integrity classes are given as follows:

S: unclassified < classified < secret < top-secret (2)

I: integrity-low < integrity-medium < integrity-high (3)

We derive the *secrecy lattice* from Denning [8]. To adjust the secrecy classification of data packets travelling through the network, we adapt Weissman's high-watermark approach [11] to the network-level: The secrecy classification of a packet is determined by the node or path with the highest secrecy clearing that has been passed by this packet. Weissman introduced the high-watermark approach for determining a file's secrecy classification in the ADEPT-50 time-sharing multi-user environment.

For *integrity lattices* we refer to Biba [14] and Karger [13]. We apply the low-watermark approach to integrity classes: The integrity class of a packet is determined by the node/path with the lowest integrity clearing that has been passed by this packet. Let I and S denote a packet's integrity and secrecy class and $IC^{node/path}$ and $SC^{node/path}$ denote integrity and secrecy clearances of the node/path involved in an event (receive, transmit, send), then

$I := min(IC^{node/path}, I)$ (4)

$I^0 := IC^{node}$ (5)

$S := max(SC^{node/path}, S)$ (6)

$S^0 := SC^{node}$ (7)

I and S denote the cumulative history of the security class of a data packet. (4) implements the down-grading of the integrity class to the lowest integrity clearance of a node or path that has been visited by this packet. Equation (6) upgrades a packet's secrecy class to the highest secrecy clearance of a node that has been visited by this packet.

Packets are initially put into the class that corresponds to the sender's clearance, cf. equations (5) and (7). This scheme tends to over-classify data packets. Therefore, we will introduce "sanitizers" later in this contribution.

The *category* of a packet is updated when the packet enters a node and checked when it leaves a node. As equation (1) indicates a totally ordered set of categories, we can down-grade the category of a packet according to the category of the incoming path.

This said, we will subsequently define access control equations (decision functions) for sending, receiving, and forwarding data packets in routers or hosts. Let SC^{node}, IC^{node}, $SC^{nexthop}$, IC^{pin}, and SC^{pout} be the secrecy and integrity clearance of the local node, the next hop, and the incoming and outgoing path, respectively. Let C^{node}, C^{pin}, and C^{pout} be the category of the node, the incoming path, and the outgoing path. If S, I, and C describe the current security classes and category of a data packet, then the access control decision functions are defined as follows:

send guard (cf. outgoing AC in Fig. 2):
$(S <= min(SC^{nexthop}, SC^{pout}))$ AND $Policy(node,out,C,C^{pout})$
re-labelling: $I := min(I, IC^{node})$ (8)

receive guard (cf. incoming AC in Fig. 2):
$(IC^{node} <= min(I, IC^{pin}))$ AND $Policy(node,in,C,C^{in})$
re-labelling: $S := max(S,SC^{node})$; $C := min(C,C^{pin})$ (9)

forward guard: send guard AND receive guard
re-labelling: I, S, C cf. (8),(9) (10)

(8) describes that a data packet is sent to the next hop if and only if the next hop and the path to the next hop have appropriate secrecy clearance. When *sending* a data packet, its integrity level is adjusted according to (4). IC^{path} is not included in (8), as packets will not pass the incoming guard of the receiving node, if IC^{path} is larger than the receiving node's IC. Before sending data packets, the category is checked against the local policy, too. The predicate $Policy(node,out,C,C^{pout})$ says, whether packets may leave the protected domain, or not.

(9) postulates that incoming data packets are accepted, if and only if their integrity class is not lower than the node's clearance. *Receiving* a data packet can change a packet's secrecy class. Note that the secrecy level is not adapted to the path's secrecy clearance. We do not expect the path to add any secret information to the data packet. The category of a data packet is adapted to the category of the incoming link. An "internal" marked packet might be re-labelled company-internal or external, depending on the category of the incoming link.

Regarding secrecy, (8) and (9) implement *send-up* and *receive-down*[3] rules. Regarding integrity, (8) and (9) can be interpreted as *send-down* and *receive-up*. (10) describes forwarding as a combination of receiving and sending. This equation may be used within trusted routers that implement incoming and outgoing access control at a single point.

2 Companies might not grant access to top-secret data to external hosts no matter the clearance of the host. E.g. the environment of the host might not be trusted (shoulder-surfing) or electronic emanation leaks data.

3 This means, that we can send data to a higher secrecy clearance holder and we can receive data from a lower secrecy clearance holder.

If the respective guard is not true, then the packet can be discarded, logged, or –in case of sending– rerouted via sanitizers that check whether the packet is over-classified. Sanitizers are described later in more detail.

Regarding an information flow model [8], we can conclude that our model satisfies the reflexivity, transitivity, and anti-symmetry rules. A node can send data to itself; the above guards and equations allow this for any host (reflexivity). If node A can send packets to node B and node B can send packets to node C, then node A can send packets to node C, too, e.g. through B (transitivity). If node A can send packets directly to node B and node B can send packets directly to node A, then A and B have the same secrecy and integrity clearance (anti-symmetry). Hosts with different clearance cannot communicate directly.

3.2 Access control information and policies

Usually, classification and category information of objects is maintained in a secure environment. A subject requesting access to an object must deliver certified clearance information. There are two major restrictions for implementing network-level access control in an efficient way:

• Access control on the network level shall be independent of the management of routers, hosts, user accounts etc. For this reason, we do not differentiate between users as we cannot verify their identity. We could distinguish between different applications using port numbers of packets that are sent or received. But then, we would assume a multi-level secure host that implements the separation of applications with different clearance. As our approach is meant to complement user oriented access control, a single integrity and secrecy clearing per host and path seems acceptable.

• The classification and category of data packets is not stored in a secure environment but included in the data packets' protocol data (header). The access control decision functions (guards) decide firstly whether to trust the access control information of a data packet or not. If they do not trust the information delivered with the packet, they re-label it with a default classification and category.

To address these problems, the security policy comes into play. Security policies include the clearance of hosts and communication paths in the local domain, which are needed for access control decisions (e.g. send guard). The security policy also includes rules for validating a packet's content based on tags that are included in an IP packet.

Context tags provide history information of a packet which helps to decide about the trustworthiness of the packet and its contents. Tags can denote the incoming link, the category, the use of protected IPSec tunnels (VPN) or link encryption. Based on these tags, policy rules are applied and an authenticity level for the packet is yielded. We distinguish between the following authenticity levels:

$$\textit{non-authentic} < \textit{ambiguous} < \textit{authentic} \qquad (11)$$

Policy rules to validate incoming data may look as follows:

• We trust the contents of IPsec-tunneled packets as much as the IPSec peer at the other end of the tunnel.

• The secrecy clearing of a shared medium is the lowest of the clearings of any connected node and the medium itself.

• *Ambiguous* data packets are marked and forwarded only to destinations that apply a next level of access control themselves, not to unprotected internal nodes.

Policy information is used when checking send and receive guards and re-labelling incoming or outgoing packets according to (8), (9) and (10). Distrusted packets are re-labelled according to a default policy, e.g. "unclassified, integrity-low, external".

3.3 The confinement problem

The confinement problem deals with misbehaving applications or users that might short-circuit access control mechanisms. In our case, this results either in unauthorized information leakage (with regard to the actual security policy) or processing of forged and tampered data. We assume access control devices being installed at domain boundaries and protected by a secure run-time environment. The totally ordered secrecy and integrity lattices and the sending and receiving guards ensure protection against these attacks at domain-boundaries.

However, the data flow within a subdomain is not controlled, i.e. no protection is given there by network-level access control. Hence, Trojan horses in hierarchical domains can be countered by additional end-to-end or point-to-point security functions between subdomain boundaries. In doing so, a Trojan horse in an intermediate domain will just see encrypted and authenticated data packets. The "polluted" subdomain is bridged from the perspective of the access control devices.

As a result, our model restricts Trojan horse problems to a single protection domain. Within that domain, a covert channel or subversion channel cannot be countered; but the data cannot leave the domain (through the network) as long as surrounding access control mechanisms are working. Even if misbehaving applications tag top-secret data as unclassified, the hosts' or subdomain's top-secret clearance will prevent any application data from being sent through access control devices to paths or hosts with lower than top-secret clearance.

4 Thwarting over-classification

Applying the above access control mechanisms leads to traffic that tends to be over-classified. We up-grade, e.g., the secrecy class of data packets as soon as they enter nodes with higher clearance. This is due to the worst case

assumption, that packets leaving this node might include data classified as high as the node. It prevents nodes from sending any data to nodes with lower clearance (lower secrecy level or higher integrity level). We propose two mechanisms to counter this effect:

- *Multi-level secure forwarders* avoid over-classification and can bridge insecure paths (using IKE/IPSec).
- *Sanitizers* reduce over-classification by carefully down-grading secrecy or up-grading integrity classes of packets.

Both mechanisms move the actually allowed traffic towards the theoretical limit, given by the access control lattices and the data packets' real sensitivity.

Multi-level secure forwarders. Multi-level secure forwarders work on the network level. Assume the scenario depicted by Fig. 4: A single router connects four hosts (or subnets), of which two –A, C and respective links to the router– are cleared for secret data, whereas the other two (B and D and their paths) are cleared for top-secret data. The router must have a secrecy clearance of top-secret to be able to forward packets from host B to host D. But then, it cannot forward any packets from host A to host C. The reason for this is, that the secrecy class of any packet sent by A will be upgraded to the router's top-secret level. Consequently, such packets can not be delivered to host C.

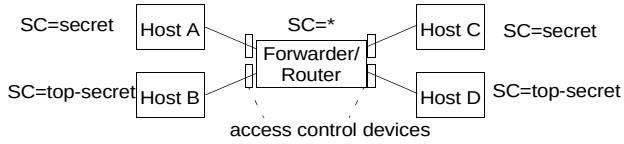

Figure 4: Multi-level secure forwarder

If the router itself, i.e. the routing functions including buffer and queue management, is not trusted then this is a good restriction. In this case, packets from host A to host D could accidentally be delivered to host C. This could result in leaking top-secret information.

This router must be trusted not to mix packets from hosts B or D with packets from hosts A or C and work properly. Hence, the router will not re-label the packet with its own secrecy level but only check the receive and send guards given in Sec. 3.1. In doing so, hosts A and C can communicate over router R as well as hosts B and D without compromising the mandatory access control scheme. Equation (10) might be called *distrusted forwarding* rule. A trusted forwarder incorporating access control devices can skip their own clearance when re-labelling data packets.

Multi-level secure forwarders enable sharing of routing infrastructure and are useful within subnets to connect hosts with different clearance.

Multi-level secure IPSec forwarders. We can build more powerful forwarders if we protect data packets between sender/receiver and forwarder. In this case, the forwarder can receive packets of any security class (as he is trusted) and the sender can use IKE/IPSec [17] to bridge intermediate insecure network paths. Then, the forwarder checks whether the packets classification is compatible with the clearance of the paths, hosts, and routers to the next access control device or to the destination. If the guards do not hold, then the forwarder can protect the secrecy level of the packets by encrypting them, e.g. using IPSec. This is useful if the other end of the IPSec tunnel has appropriate clearance to receive (unwrap) the original IP packets, i.e. the send guard holds for the IPSec peer.

However, IPsec-tunnels can only be trusted if they start or end at the access control devices –or in trusted neighboured devices– as otherwise the peer-authentication can not be trusted. Authentication is essential as claimed IKE identities are used for checking clearances in the send and receive guards. Trojan horses on different sides of access control devices can send IPSec packets with forged origination and destination addresses to build covert channels. Nevertheless, even bridging insecure paths by standard IKE/IPSec tunnels in *tunnel mode* opens covert timing channels of limited bandwidth using traffic analysis. It probably opens covert data channels, using IPSec header parameters (e.g. using Security Parameter Indices), too.

Similarly, the integrity level of packets can be preserved by using IKE/IPSec authenticated tunnels to send them over network paths with lower integrity clearance. Secrecy and integrity classes can be used when configuring IKE/IPSec to chose proper IPSec policies. In doing so, encapsulation strength and cost is adjusted to security needs.

Up- and down-grading sanitizers. *Sanitization* mechanisms are very critical as they work against the lattices. Sanitization functions are discussed by Stork [10]. They take the current security level (S^c,I^c), the new security level (S^n,I^n), and the contents of a data packet as input and rewrite the packet in a way that the new data packet can be released with the new security level without compromising the security policy.

Sanitized *down-grading* refers to the construction of an information path from a given data packet to a data packet with a lower *secrecy level*. Sanitized *up-grading* refers the construction of an information path from a given data packet to a data packet with a higher *integrity level*. Up-grading secrecy or down-grading integrity need not to be sanitized as they comply with the lattice model.

Sanitized down-grading of secrecy levels: To down-grade the secrecy level S^c of a data packet, the sanitization function must inspect the data packet and discard any information whose secrecy level is higher than the secrecy level S^n of the new packet. Decisions about the secrecy level of application data can be supported by company-wide application level security labels [6]. Protecting

WWW-Servers or browsers, this sanitization could, for example, filter cookies, user information, or XML-Documents [28] with high secrecy labels. Using user level information to determine secrecy levels of packets, we assume that the sending host is trusted and marks classified data. This kind of down-grading can be allowed within integrity-high environments; in other environments, automatic down-grading of secrecy should be forbidden.

Sanitized up-grading of integrity levels: Sanitization functions inspect data packets and decide whether the integrity level of the packets can be raised or not. This is a difficult job and must be done regarding a specific application environment. For WWW-Servers, it could filter CGI script parameters – or any other data that might compromise a WWW-Server – before up-grading the integrity level. Up-grading integrity levels is done after receiving packets. If an up-grade to the current integrity clearance of the network or host is not possible, the packet must not be passed but discarded or logged. Sanitizing integrity might lead to stateful filtering as well as intrusion detection systems [20] inspecting correlated packets. On application level (de-fragmentation and re-assembly assumed), meta-virus scanners can be included into sanitizing.

Re-introducing the confinement-problem. *Sanitizers* are a sensitive part of the access control architecture as we must assume Trojan horses or malicious users within all hosts that are not trusted by default. It follows, that sanitizers must maintain the *-property [7]:

- Sanitizers are likely to be fairly complex but must be correct at the same time in order to make sure that they do not allow information paths that are forbidden in the lattice model (disclosure or subversion channels).
- The sanitization function must evaluate threats to which the data packet has been (->integrity) or will be (->secrecy) exposed. E.g., if decisions are based on security labels, it is presumed that these labels have not been removed or altered in an unauthorized way by previous nodes.

If these assumptions cannot be guaranteed, the *-property might be violated and data might be leaked or forged subversive data might be processed. How to ensure there is no covert channel left between machines? Rushby, Randall [12], and Karger [23] describe many problems with regard to covert channels between hosts with different clearance. Today, we must assume that down-grading of secrecy-classes introduces (limited) covert channels.

5 Implementation

This section describes the implementation of the network-level access control schema. The security relevant history of a packet is inferred from labels and tags, which have been added by former processing stages of this packet, and from environmental conditions (context information). Firstly, *screening modules* are used to validate context information. Then, *tagging modules* add tags or update labels. Finally, *filtering modules* implement access control decision (filtering rules) and enforcement (discard/pass) functions. For efficiency reasons, tagging modules must be simple. As a simple designs promotes the overall security; simple designs are preferred for the other modules, too.

5.1 Basic modules

Screening modules examine incoming packets and check them against explicit properties, e.g. IP addresses, port numbers, type of service-bytes (e.g. IPSec), or authentication data that must be validated. Screening modules examine incoming packets to assign them to different groups that can be handled independently by succeeding tagging or filtering functions. If packets share homogeneous implicit properties (e.g. arrive over an authorized IPSec channel), screening can assure that this property is stored in the packets by following tagging modules for future use.

Fig. 5 shows screening as the first processing step applied to incoming packets. Implicit properties are denoted by braces, tags by brackets. On the left hand side data packets with a distinguishing property (e.g. message authentication code) enter the screening module. After verifying the property, the screening module chooses a branch according to the result of the verification (properties {P1}, {P2}). Thus, packets are sorted into different groups – embodied by logical or physical output links. Screening modules do neither alter nor discard packets.

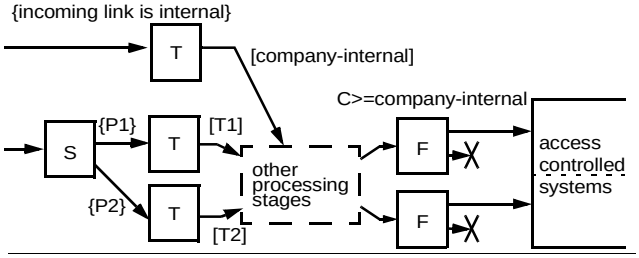

Figure 5: Stages in network-level access control

Implicit properties of each outgoing link of a screening module can be tagged to data packets. These tags (e.g. [T1] and [T2] in Fig. 5) can be used as input for filtering in later processing steps.

Tagging modules (T-boxes in Fig. 5) add tags to every packet passing by on a transmission path. Thus, an implicit property that is correlated with the transmission path is transformed into a tag and stored in the packet. Fig. 5 shows a tagging module that transforms the implicit property "internal incoming link" into explicit information by storing it as a "comp.-internal" category tag in the packet.

Filter modules (*tag filters*) realize access control functionality. Each of these filters operates on a single tag. A

filtering rule is a predicate that yields true for all packets that pass through the filter. All other packets are sieved out by the filter and can either be discarded, logged, or transmitted to a further processing stage (e.g. another filter). These packets satisfy the complementary filtering condition. Since lattices have been introduced, filters can be designed as threshold filters. Fig. 5 gives an example (upper F-box) of the filtering of incoming packets which are at least company-internal (cf. equation (1)). More complex filter rules can be implemented by combining several basic filters. Then, combinations are represented as boolean expressions (AND, OR, etc.). The modules introduced above generally occur in the following order: screening, tagging, and finally (distributed) filtering [2].

5.2 Node architecture

Since tagging is used to support remote filtering, the stages in Fig. 5 can be distributed over the network. Screening and tagging modules are usually located at nodes or routers that have access to implicit context information. Such a node can be an access router to a company network which screens incoming packets according to the type of access.

An access router, e.g., determines a packet's category and tags the category to the packet (internal, external, etc.). Additionally, it labels the packets with initial secrecy and integrity labels according to the *default policy*. The corresponding filter modules that exploit the category tags are distributed and can be implemented in any trusted environment within the subnetwork.
Usually, consumers of tags are local filter modules following the tagging module or remote filter modules guarding internal subdomains or an exit of the current subnetwork.

Fig. 6 depicts the access control architecture for incoming packets. All incoming packets are subject to an unconditional filtering (F^R) on remotely set tags. In this example all external or ras-internal packets are discarded before fine-grained access control is applied. This counters external denial of service attacks on the screening module.

The screening module examines the properties of incoming data and distinguishes non-authentic (negative authentication), authentic (successful authentication regarding given security policy), and ambiguous (authenticity can not be determined) packets. In the filtering stage the first filter F^a discards non-authentic packets. Filter F^b passes packets that are authentic and gives the remaining ambiguous packets to the third filter which in turn passes them if they are tagged *internal*. Packets which are passed towards a cross are discarded. We can handle discarded packets by logging them or by sending an ICMP error message back to the source to inform the sender about the problem (unless the discarded message was ICMP itself to avoid avalanche effects). In [5], such an ICMP message is composed with type "Destination Unreachable" and error code "Communication Administatively Prohibited".

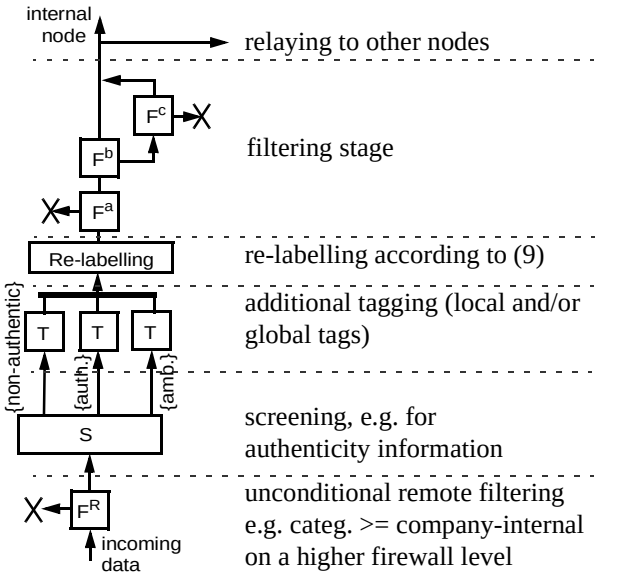

Figure 6: Node architecture for incoming data

The architecture for outgoing access control is similar. In a first step all outgoing packets are unconditionally tagged with the node's secrecy and integrity clearance and the category of the packet. Then screening is performed on destination addresses and the secrecy clearances of the selected outgoing path and the next hop are determined. This information is tagged to the packet. Then, the packet's security label is updated according to equation (8) and local tags –used to store sensitive information which shall not leave the current subnetwork– are removed. Finally, a filtering stage implements the send guard according to equation (8).

5.3 Example

The following example is based on Fig. 1. We assume that nodes and links have the properties shown in Table 1 and that routers are multi-level secure and trusted (cf. Sec. 4).

System component	SC	IC
sending host C1	secret	high
links in subnet C	secret	medium
router C - Internet	trusted environment	trusted environment
Internet links	unclassified	low
router Internet - A	trusted environment	trusted environment
links in Subnet A	classified	medium
router A - B	trusted environment	trusted environment
links in subnet B	secret	high
receiving Host B1	secret	high

Table 1: Clearance of system components

In Table 2, the transmission path of a packet between hosts C1 and B1 is shown including events and re-label-

ling. The values of the labels S, I, and C denote the state of the data packets after the event in the same row. Rows with the event *transmission* visualize the view of the receiving side on the incoming link. Framed fields indicate secure IPSec forwarding. IPSec is applied to bridge insecure domains for which the guards in equations (8) and (9) would not hold otherwise. Access control functions are located within a secure module as depicted in Fig. 4.

	Event	S	I	C
1	send at C1	secret	high	internal
2	generate MAC	secret	high	internal
	transmission in C	secret	medium	internal
3	check MAC at I-C	secret	**high**	internal
4	encrypt at I-C	**unclassified**	high	internal
5	generate MAC at I-A	unclassified	high	internal
	transmission in I	unclassified	low	external
6	decrypt at router I-A	secret	low	**co.-internal**
7	check MAC at I-A	secret	**high**	co.-internal
8	encrypt at router I-A	**classified**	high	co.-internal
9	generate MAC at I-A	classified	high	co.-internal
	transmission in A	classified	medium	co.-internal
10	decrypt at router A-B	secret	medium	co.-internal
11	check MAC at A-B	secret	**high**	co.-internal
	transmission in B	secret	high	co.-internal
12	receive at host B1	secret	high	co.-internal

Table 2: Packet processing

Explanation for Table 2: 1) Node C1 generates a packet and the security module sets S and I according to the values assigned to the host (see (5) and (7)). 2) A message authentication code (MAC) is added by a functionality encapsulated in the trusted security module (e.g. [24]). The packet is transmitted on the medium-integrity link. 3) At router C-I, the claimed I-class does not correspond to the link's I-class. The trusted router validates integrity by checking the MAC but it cannot send the packet into the internet since this would violate the send guard (SC^{pout} and C^{pout} are too low). 4) Therefore, an encrypting module reduces S of the packets. 5) A MAC is added to bridge the insecure Internet. Steps 4 and 5 can be implemented by IPSec. The encapsulated packet is sent to the Internet. Thus, it implicitly loses it's I- and C-properties. 6) Decrypting the packet. The security class is restored according to step 4. Note, that the category update from *external* to *company-internal* is actually a down-grading from *internal* in step 4. 7) Raising the integrity level by validating the MAC. 8)-11) Again, the secrecy clearance of the next subnetwork is not sufficient and encryption is applied. Note that the secrecy level shall be reduced to the highest possible level after encryption and restored after decryption. This limits emerging covert channels to the bridged domains. 12) In the subnetwork B, category, secrecy level, and integrity fit the lattice rules; no additional protection is needed.

5.4 Implementing tags and labels

Tags (or labels) can be composed in an optimized format for network protocols. This means, they must be easy to access and as compact as possible. FIPS PUB-188 [6] defines such structures with a tag type field (1 octet), a length field (1 octet), and finally the security data field. Kent [5] defines security options for the Internet Protocol (IPSO). These can be used to carry the classification and category of IP packets as well as additional history information in the additional security information option.

Referring to our classification and categorization scheme in Fig. 3, we can store the integrity level information in bits 0-3 and the secrecy level in bits 4-7 of the Classification Level octet of the Basic Security Option [5]. The classification can be stored together with the Protection Authority encoding in the Protection Authority flags. In doing so, re-labelling will affect a single IP option only.

Explicit context information – incoming link, virtual private network classification – as well as digests over the security options can be stored in the Extended Security Option. The IP header can be up to 60 Bytes (15x32 bit). The fixed header is 20 bytes. Consequently, there is enough space left to include tags in further options.

6 Conclusions

This contribution has introduced a new network-level access control schema that is based on lattices. It controls access to data packets that are exchanged between hosts. The schema addresses the confinement problem by use of mandatory access control. It is secure –i.e. grants access to data packet's contents to authorized (cleared) subjects only– if following assumptions hold:

- Users and hosts can not change access control information, access decision and enforcement functions.
- The security policy is set correctly, i.e. clearances of hosts, network nodes, and paths hold.
- Information stored in tags and security labels, included in packets for access control, complies with the packets secrecy class. Sensitive tags (e.g. network addresses) are categorized "internal" and filtered at external links.
- Protection means included in evaluating the send- and receive-guards (e.g. IKE/IPSec, link encryption) work according to their clearance assumptions.
- Sanitizers do not open covered (data or timing) channels and work reliably.
- Additionally, access control devices must satisfy *three requirements* [10],[18]: they must be (i) invoked on every access attempt; (ii) tamper proof, (iii) small and reliable.

The first requirement postulates, that any access point to a domain (i.e. network router, subnetwork router, or host)

must be guarded by access control devices. This implicitly shows, that we do not protect against Trojan horses within a domain. Logging and audit trails might profit from the additional history within captured packets to trace back to misbehaving entities. Our approach does protect packet flows only. Other information flows –e.g. electromagnetic emanation, social engineering, 'dumpster' channels– must be addressed, too. The second requirement can be satisfied by using small, certified security environments like the IBM 4758 cryptographic coprocessor [24] with high assurance certification. The third requirement is supported, as our approach leads to less complicated access control functions. Firstly, because it operates on network level with less complicated data structures. Secondly, because we do not use user-level certificates and authentication.

The re-labelling equations include up-grading of secrecy and down-grading of integrity levels only. Consequently, the confinement-problem is solved: Data will not be released to components with lower secrecy or higher integrity clearance. Policy data that is used by access control decision functions is static and cannot be changed by applications on the hosts. Thus, information leakage by provoking access control decisions (reject/pass), from which observing entities can derive information – e.g. shown for file access by Lampson [16] –, is restricted.

One of the established principles for network layer protocols in the Internet is, that network nodes do not keep state information per packet flow. Our approach provides a natural broadening of this principle to network level access control. Few state information is contained in the packet itself and the network nodes do not keep information per packet. Instead the nodes use environmental policy information to decide about the trustworthiness of a packet's contents, surrounding nodes and communication paths.

Acknowledgements
The authors would like to thank Paul Karger for fruitful discussions and interesting primary references and hints.

7 References

[1] M. Blaze, J. Feigenbaum, J. Lacy: Distributed Trust Management. IEEE Conference on Security and Privacy, CA. May 1996.
[2] M. Kabatnik, R. Sailer: Modelling of Secure Interconnection. Communication Fraud Control Association, Spring International Conference in Ismaning/Germany, May 1999.
[3] R. Housley: Security Labels Framework for the Internet. May 1993. RFC 1457 (Informational).
[4] S. Bellovin: Distributed Firewalls. login, Nov 1999.
[5] S. Kent: Security Options for the Internet Protocol. Nov 1991. RFC 1108.
[6] FIPS PUB 188: Standard Security Label for Information Transfer. Federal Information Processing Standards, Sept. 1994.
[7] D. E. Bell, L. J. LaPadula: Computer Security Model Unified Exposition and Multics Interpretation, The MITRE Corp., ESD-TR-75-306, HQ Electronic Systems Division, Ma., June 1975.
[8] D. E. Denning: A Lattice Model of Secure Information Flow. Communications of the ACM, Vol. 19, No. 5, May 1976.
[9] International Organization for Standardization, Information processing systems – Security Architecture, ISO 7498-2, 1988.
[10] D. F. Stork: Downgrading in a Secure Multilevel Computer System: The Formulary Concept. The MITRE Corp., ESD-TR-75-62, HQ Electronic Systems Division, Ma., 1975.
[11] C. Weissman: Security Controls in the ADEPT-50 Time Sharing System. AFIPS Conference Proc., Vol. 35, N. J., 1969.
[12] P. A. Karger: Non-Discretionary Access Control For Decentralized Computing Systems. MIT/LCS-TR-179, Massachusetts Institute Of Technology: Cambridge, MA. May 1977. URL: http://ncstrl.mit.edu:80/Dienst/UI/2.0/Describe/ncstrl.mit_lcs%2fMIT%2fLCS%2fTR-179
[13] P. A. Karger, V. R. Austel, D. C. Toll: Using a Mandatory Secrecy and Integrity Policy on Smart Cards and Mobile Devices. EUROSMART Security Conference, June 2000.
[14] K. J. Biba: Integrity Considerations for Secure Computer Systems. ESD-TR-76-732, HQ Electronic Systems Division, Hanscom AFB, Ma., April 1977.
[15] B. W. Lampson: Protection. Proc. 5th Princeton Conf. on Information Sciences and Systems, Princeton Univ., March 1971.
[16] B. W. Lampson: A Note on the Confinement Problem. Communications of the ACM, Vol 16, No 10, October 1973.
[17] RFC 2401: Security Architecture for the Internet Protocol; RFC 2409: The Internet Key Exchange (IKE). November 1998.
[18] J. Anderson: Computer Security Technology Planning Study, ESD-TR-73-51, Vol I+II, HQ Electronic Systems Division, Hanscom AFB, Ma., Oct. 1972.
[19] T. Dierks, C. Allen: The TLS Protocol Version 1.0. RFC 2246. January 1999.
[20] H. Debar, M. Dacier, A. Wespi: Towards a Taxonomy of Intrusion-Detection Systems. Computer Networks 31, 1999.
[21] S. N. Foley, L. Gong, X. Qian: A Security Model of Dynamic Labelling Providing a Tiered Approach to Verification. In Proceedings of the Symposium on Security and Privacy, CA, 1996.
[22] R. S. Sandhu: Lattice-Based Access Control Models. IEEE Computer, Vol. 26, No. 11, November 1993, pages 9-19.
[23] J. Rushby, B. Randell: A Distributed Secure System. IEEE Computer, July, 1983, pages 55-67.
[24] S. Smith, S. Weingart: Building a High-Performance, Programmable Secure Coprocessor. IBM Research Report RC21102(94393). New York, February 1998.
[25] S. McCanne, V. Jacobson: The BSD Packet Filter: A New Architecture for User-level Packet Capture. Proc. of the 1993 Winter USENIX Conference, January 1993.
[26] D. R. Engler, M. F. Kaashoek: DPF: Fast, Flexible Message Demultiplexing using Dynamic Code Generation. Proc. of the ACM SIGCOMM'96. 1996.
[27] S. B. Lipner: A comment on the confinement problem. Operating Systems Review, Vol 9, No 5, 1975. pages192-196.
[28] XML-Signature Syntax and Processing: W3C Working Draft 11-July-2000. http://www.w3.org/TR/xmldsig-core/