Use of Cross Domain Guards for CoNSIS network management

Philipp Steinmetz
Cyber Defense
Fraunhofer FKIE
Wachtberg, Germany
philipp.steinmetz@fkie.fraunhofer.de

Abstract— **This paper discusses filtering of messages sent from a classified to an unclassified network using a cross domain guard. We discuss how we can use such a guard within the network architecture designed in the CoNSIS (Coalition Networks for Secure Information Sharing) project for use in future coalition operations. A guard design is presented which enforces that only XML messages conforming to a specific format may pass the guard. It also limits the message rate based on message size and the resulting possible covert channel. We can use this guard design for low data rate applications which have to communicate across networks of different classification. We also discuss a proxy device located in the unclassified network to reduce the required amount of communication between classified and unclassified network.**

Keywords-Information Security; Computer networks

I. INTRODUCTION

Protecting confidential information while at the same time reaping the benefits of networked systems is an important goal. Traditionally military computer networks containing sensitive data have been protected by physically separating them from other systems. This complicates or prevents many important applications for which data has to pass from a classified to an unclassified system. Cross Domain Guards have been developed to allow a controlled exchange of information between systems of different classifications while filtering confidential information. The focus of this paper is on looking into the intended behavior of guards. While the actual implementation of guards is not the focus of the paper, we keep in mind that composing them from small building blocks which interact in a simple fashion is helpful for secure implementation.

II. THE CONSIS PROJECT

The CoNSIS (Coalition Networks for Secure Information Sharing) project is a joint effort of France, Germany, Norway and USA. The focus is on designing network architectures and protocols for future coalition operations. The work is distributed among five tasks. The author participates in Task 3 which is responsible for security.

The overall architecture [3] contains many elements of the Protected Core Networking (PCN) concept, but it is not identical. In CoNSIS several Colored Enclaves (CEs) are each connected to a Transport Network (TN) (Figure 1). The TN consists of several Transport Network Segments (TNSes). The CEs may contain unencrypted classified data. They are assumed to be physically protected from unauthorized access. Each one is run by a nation participating in the coalition.

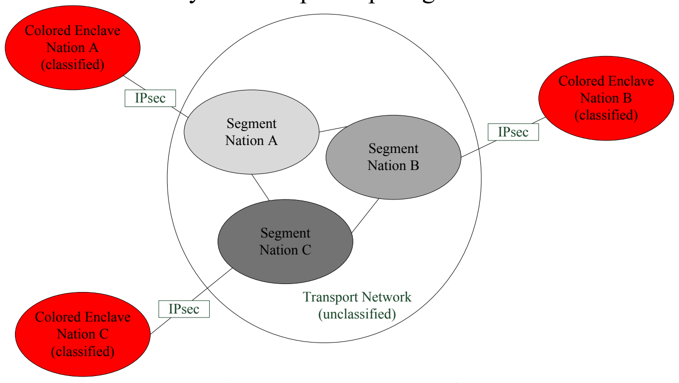

Colored Enclave Nation A (classified)
IPsec
Segment Nation A
Segment Nation B
IPsec
Colored Enclave Nation B (classified)
Segment Nation C
Transport Network (unclassified)
IPsec
Colored Enclave Nation C (classified)

Figure 1: A CoNSIS network

The TNSes are either run by a nation or by the coalition. The TN they form is an unclassified network with focus on availability without confidentiality protection. This means that classified data transmitted from one CE to another has to be encrypted before it reaches the TN. This is achieved by placing IPsec devices between each CE and the TN which encrypt all traffic leaving a CE and decrypt the incoming traffic. Message confidentiality depends on correct installation of the IPsec devices and protecting the CEs from unauthorized physical access.

III. MULTILEVEL SECURITY AND CROSS DOMAIN GUARDS

Protecting classified information from unauthorized disclosure is among the most important goals in information processing in military applications. Strict separation of devices handling information of different degrees of confidentiality is often used to achieve this. For example, a user employs both a Secret and an Unclassified workstation not connected to each other for handling data of each classification.

Such complete separation also prevents desirable flows of information between the systems. Full replication of hardware also means higher weight and greater power consumption, which can be problematic for mobile units.

Multilevel Security deals with handling data of several classifications on the same device according to some set of rules. One well-known rule set is Bell-LaPadula (BLP) [2]. Each information object has a specific classification and each user has a clearance for access to data up to a specific maximum classification. BLP enforces that no data can be transmitted to a user with insufficient clearance. This is achieved by two rules. The first rule enforces that a user may not read data without having sufficient clearance. The second one prevents users from writing data to objects with a lower classification than their own clearance. This prevents data leaks by malicious software executed by a user with a high clearance.

This strict rule set does not provide mechanisms for releasing or downgrading data which is no longer considered confidential or had its confidential parts removed. Often, some downgrading mechanism has to be implemented and exempt from the BLP rules for practical reasons. In [1] Rushby introduces the concept of a separation kernel. Such a separation kernel restricts the interaction of processes on a machine to specifically allowed communication. It allows a system to behave like a distributed system with specified connections but runs on a single piece of hardware. The motivation for this is using a separation kernel for providing reliable separation of processes and using specialized code to enforce policy by message filtering and verifying the correct behavior of each individually.

There are several applications such as safety-critical real-time systems which are required to behave deterministically without being influenced by other processes. Rushby explicitly names filtering data which has to bypass an encryption device as an application.

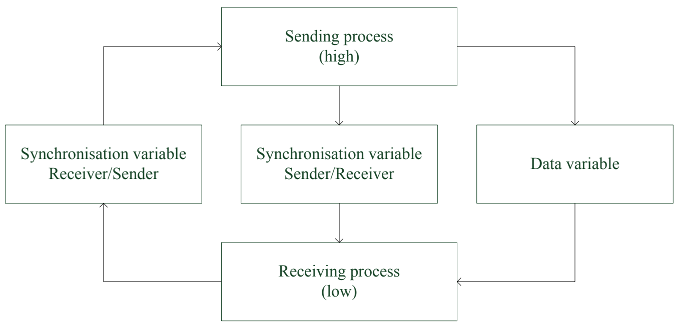
We design a downgrading mechanism based on a separation kernel. One partition contains the classified data (red), one contains the unclassified data (black) and a third contains the downgrading mechanism filtering the data (Cross Domain Guard). The separation kernel enforces that no data flows directly from red to black but has to go through the guard first. This means that only the separation kernel and the guard have to be trusted. Weaknesses in other code cannot be exploited to circumvent the guard.

IV. Steganography and Covert Channels

The main task of a Cross Domain Guard is to enforce a policy on the traffic flowing through it. It has to prevent the unwanted release of classified information. The obvious part of this task is to prevent accidental or malicious transmission of classified information which is transmitted as application data and properly marked or otherwise recognizable as classified. A guard identifies the data by searching it for “dirty words” such as “secret”, validation against an XML schema, which describes the format of messages intended to pass the guard, fails or some other mechanism inspecting the message payload flags the message as classified.

Apart from this more subtle ways of data transmission have to be taken into account. Steganography is the art of hiding information inside other information in order to conceal the existence of the hidden message altogether. An overview of relevant definitions can be found in [5]. A well-known example is replacing the least significant bit of color information of pixels in an image file with the embedded message. A human observer is unlikely to notice the difference, but evading detection through statistical analysis will require more advanced techniques. Anderson explains several mechanisms in [6].

Covert channels are a related topic. They are used to transmit data from an object with a high classification (High) to one with a low classification (Low). In [6] a covert channel is defined as a mechanism not intended for communication which can be abused to communicate information from High to Low. In [7] the components of a covert channel, different examples and countermeasures are explained. A covert channel consists of a data variable and two synchronization variables, one sender-receiver (s-r) and one receiver-sender (r-s) synchronization variable. The first two variables are properties of the system which can be set by High and read by Low. The last one can be set by Low and read by High (Figure 2). High sets the data variable to a state representing the information to be transmitted. In the simplest case one of two states representing either 1 or 0 is set. High then uses the s-r variable to indicate that data can be received. Low reads the data variable and uses the r-s variable to inform High that it has received data. This process is repeated until all data has been transmitted.


Sending process (high)
Synchronisation variable Receiver/Sender
Synchronisation variable Sender/Receiver
Data variable
Receiving process (low)
Figure 2: Covert channel components (see [7])

When a common time reference is used for instead of the synchronization variables, the channel is called a timing channel otherwise it is called a storage channel. Properties of shared resources can be used as variables. A simple example is a hard disk shared by High and Low with access control mechanisms in place to prevent Low from reading files owned by High. High can allocate almost all remaining disk space and then allocate the rest to represent a 1 or deallocate some space to represent a 0. Low can now try to allocate space and determine whether it fails or not. They can then repeat this synchronized by the system clock.

Both steganography and covert channels can be used by malicious software in a classified network to send classified data to an unclassified network through a guard. Guard design has to take limiting covert channels to acceptable values into account. Acceptable values depend on the environment a guard is used in. As noted in [7], the risk of espionage by sending classified satellite images via a low data rate covert channel without being detected is low due to the large file sizes, while an encryption key vulnerable to transmission by covert channel is a serious problem unless the covert channel bandwidth is almost nonexistent.

V. A GUARD FOR MANAGEMENT DATA

The CoNSIS architecture is designed to prevent unencrypted classified data from leaking to the TN by encrypting all data which leaves a CE and only accepting data originating from other CEs into a CE. Since the TN is a means to transport data and the users working on classified data operate inside the CEs, the fact that messages cannot be exchanged between a device in the TN and one in a CE does not pose a problem to regular applications.

If we preclude all exchange of unencrypted data between TN and CE, we limit our options regarding network management. The management has to happen inside the TN. If instead we allow management data to be exchanged between TN and CE, users inside a CE can receive status information on the transport network and manage transport network segments if they are authorized to. This provides the users inside the CEs with the ability to adapt their transmission behavior to the available resources and manage the transport network according to their priorities. While devices connected to the TN could be physically located in reach of a CE user, this would mean manual control by the user and hardware replication.

Passing messages between CE and TN means that these messages bypass the IPsec device and pass a filter to remove unwanted messages.

Messages from CE to TN

- must have legitimate management message syntax,
- must not contain classified information and
- must not allow transmission of classified information through covert channels.

Messages from TN to CE

- must not introduce malicious code.

One has to balance the degree to which these goals are accomplished and the limitations enforced on legitimate traffic. This paper focuses on filtering messages from CE to TN using a guard.

VI. STRUCTURE OF A GUARD

The guard is designed as a sequence of filters running on a separation kernel. A message from CE to TN has to pass all filters before being released to the transport network. Each filter is installed on a partition of its own to minimize the size of each piece of critical code. We assume that XML is used for the management messages.

The first filter validates the XML messages against a schema of legitimate messages. The second filter enforces additional constraints to limit the possible transmission of classified data through a sequence of messages of valid format. The last filter minimizes covert channels in packet headers. Figure 3 shows the guard components. We now discuss the properties of these components.

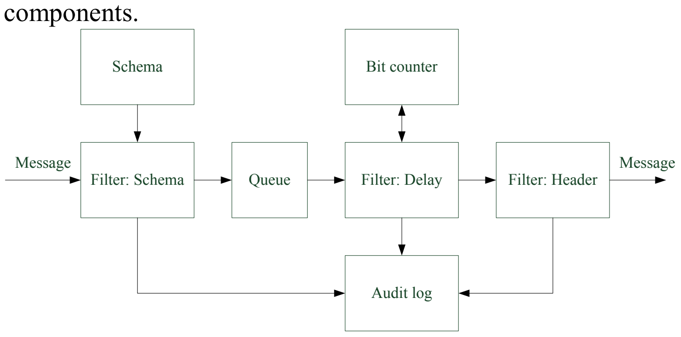


Figure 3: Guard components

The intended behavior of the first filter is specified by a schema file which is determined by the syntax of the legitimate messages. The last filter needs to overwrite packet header fields usable for covert channels. Reference [4] contains an overview of TCP and IP header fields usable for covert channels. The last filter is also responsible for limiting timing channels by forwarding incoming messages at regular intervals. The next chapter discusses the second filter.

VII. A FILTER FOR DELAYING MESSAGES

The second filter forwards and in some cases delays messages in an effort to minimize potential misuse of messages of legitimate format containing classified information hidden with steganographic mechanisms. While enforcing this security requirement, legitimate traffic needs to be delayed as little as possible. A simple version of such a filter limits the message rate by queuing them and forwarding them at fixed intervals. When the legitimate message rate is set, the expected rate in regular operation and the acceptable covert channel capacity have to be taken into account.

Depending on the application more complex requirements can be enforced by the filter such as setting individual message rates for each message type depending on their expected rate. If different message types have varying size, the acceptable message rate can be replaced with an acceptable payload bit rate. As an example, we can assume that there are two message types, message type A has no parameters and message type B has a 10 bit parameter. Sending a type A message transmits 1 bit, sending a type B message transmits 11 bits. If we assume a malicious sender in the enclave, this is the maximum amount of classified information that can be encoded in the messages themselves. We then define a bit counter which is increased according to the acceptable bit rate and decreased according to the covert channel capacity (CCcap) when a message is sent. If the counter would be reduced to less than zero, the message is delayed (Figure 4). We set a maximum value for the counter to prevent a burst of malicious messages following a long period of regular operation. This setting has to take the expected bursts in legitimate traffic into account. The bit counter and the filter queue are checked at regular intervals. If the bit counter value is sufficient for the first message in the queue, the message is forwarded and the bit counter is adjusted.

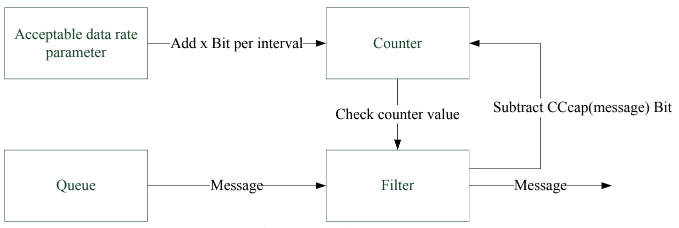


Figure 4: Bit counter

The advantage of applying a guard containing such a filter is the fact that we do not need to make assumptions about the validity of messages. We can just assume that each and every message may have been sent by an attacker using every bit to covertly send messages. Then we enforce a maximum data rate on this covert channel using the mechanisms above. No knowledge of steganographic mechanisms that may have been applied is necessary. This only works if the message rate in normal operation is low. If the message rate is high, we have the choice between two unacceptable scenarios. We either set a low acceptable covert channel data rate, which will dramatically slow down legitimate traffic or we set a high acceptable covert channel data rate and thereby give up on covert channel mitigation.

Depending on data available to the guard additional filter rules can be enforced. If, for example, there is a known set of routers management messages are sent to, we can keep a list of legitimate IP addresses and block messages to other destinations. If the relevant data is static and provided by a trusted mechanism, e.g. a protected configuration interface of the guard, it can be considered a configurable part of the first filter, the XML schema filter.

VIII. ERROR HANDLING, AUDIT AND OTHER MECHANISMS

We can install an anomaly detection mechanism to detect unusual sequences of messages. A sequence of messages switching a setting in a router back and forth or similar occurrences may be suspicious. In such cases an alarm can be raised. In order to prevent additional guard complexity we suggest that such a device is not integrated in the guard itself, but the guard and the anomaly detection mechanism are installed in sequence. Figure 5 shows the guard and the anomaly detection mechanism.

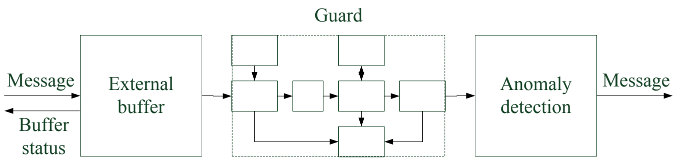


Figure 5: Guard and additional mechanisms

We assume that error messages and other logging data are sent from the components generating them to an audit component within the guard via unidirectional links (Figure 3). Only authorized administrators may access the component through a physically protected interface. This simplifies development by minimizing the information flow.

Unidirectional flow of messages through the filter means that we cannot explicitly notify a sender, if an internal buffer is full. In order to prevent legitimate messages from being silently discarded, we can prepend an additional buffer to the filter. It is not part of the trusted guard device. This external buffer forwards messages to the filter at the same rate as the guard does. Unlike the buffer inside the guard it can notify senders when it is full. Figure 5 shows the position of the external buffer.

IX. INTERACTION WITH CRYPTOGRAPHIC PROTECTION MECHANISMS

In our example, network management, we assume that the management messages are not particularly confidential and may be sent in the clear. If we use a guard for filtering of encrypted messages, we assume that the guard has access to the decryption key.

If messages are signed to prove their authenticity to the intended recipient in the transport network, we have to prevent subliminal channels - data hidden in the signature. This can be achieved through choice of signature algorithm. While for example DSA (Digital Signature Algorithm) allows the signer to choose a parameter influencing the signature, RSA signatures are deterministic which prevents a subliminal channel [6].

The text above discusses signatures used by applications to ensure integrity and authenticity. There are concepts in which a signature is applied to a message by a trusted device to label it as releasable. Then a guard releases the message if it has been signed by an authorized entity. These concepts are out of scope of this paper. The focus of this paper is on determining whether to release messages or not based on their content. A signature by the sender in the enclave is not considered sufficient for message release to the transport network in our scenario.

X. A MANAGEMENT PROXY

Minimizing the amount of legitimate messages passing the guard increases the difficulty of covertly passing classified information through them. If there are typical patterns in the messages that should pass, it can be helpful to install a proxy device in the transport network which expands messages to sets of messages. If, for example, a message has to be sent to all routers controlled by the administrator in the enclave, a single message can be sent to the proxy which instructs it to generate all these messages instead of generating all messages in the enclave.

Depending on the application different “strategy” messages to the proxy and its reaction when receiving them can be defined before deploying the system. The reaction may be more complex than just forwarding the message to multiple recipients. The goal is to identify the information that needs to be transmitted to the transport network and send the least amount of bits necessary to represent this information. This way we can set the guard to a low allowed bit rate while maintaining functionality.

In a scenario without flow of information from the TN to the CE installing such a proxy basically allows us to compress the messages from the CE to the TN. If, without a proxy, messages are also sent from the TN to the CE, it can be possible to reduce the number of messages passing through the guard in both directions. This is the case when the proxy can take care of message exchanges without further information from within the CE. If, for example, several devices report their status and receive an acknowledgement (ACK) in return (Figure 6), we can use a proxy. Instead of each status message passing the guard to the CE and each acknowledgement passing it to the TN, we can do each exchange between device and proxy within the TN and send one aggregated status message to the CE (Figure 7). In both figures "Guard" represents the whole set of mechanisms shown in Figure 5.

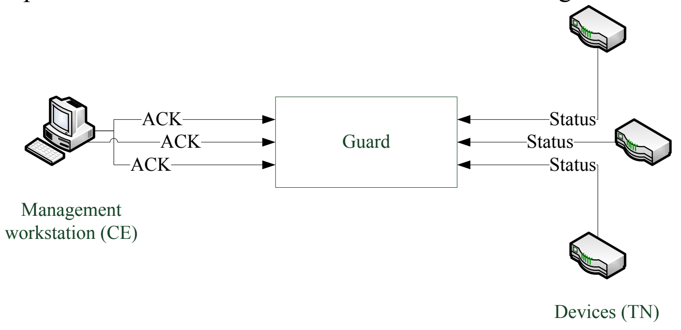


Figure 6: Information flow through a guard

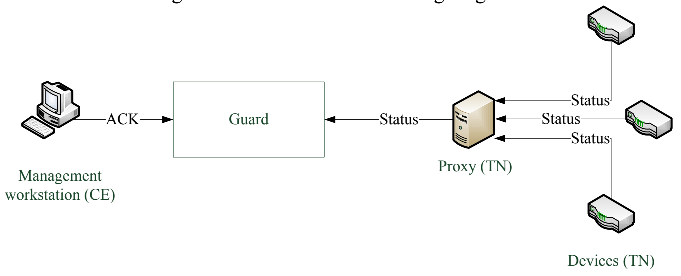


Figure 7: Information flow through a guard with proxy

For some applications such proxies may exist for efficiency reasons anyway. In that case we just have to choose where they should be placed to minimize cross domain traffic. For other applications proxies may not be considered advantageous in general because of increased complexity, delay or other reasons. Here we can analyze whether the expected decrease in cross domain traffic is worth introducing a proxy or not.

XI. RELATED WORK

Several products for cross domain filtering exist. One application is controlling the settings of an integrated radio and crypto device which encrypts all user data before transmitting it. If a computer containing classified data is connected to the device, commands sent from the computer to the radio component, e.g. setting the frequency, have to bypass the encryption component. Filter products are available, which make sure that only specific commands may bypass it.

There are also similar guard products for filtering IP network traffic crossing a boundary between two security domains. They use filters specific to protocol and application. Usually they offer customers to define filters according to their needs without mentioning details in promotional material.

We are not aware of a common format to specify the desired behavior of a guard beyond definition of the message format for example by an XML schema language. We intend to look into ways to represent more complex requirements such as message rate dependent on message properties.

XII. SUMMARY AND FUTURE WORK

This paper examined how messages can be passed from a classified to an unclassified network in a CoNSIS network with minimal risk of leaking classified data. The introduction was followed by an overview of the CoNSIS architecture and general information on Multilevel Security. Then we described a guard for filtering data sent from a classified to an unclassified network for low message rate scenarios. It consists of a series of filters running on a separation kernel. Then we discussed effects of use of cryptographic protection of messages. This was followed by the concept of a proxy device in the unclassified network to limit the amount of messages having to pass the guard. Finally we provided some pointers to related work.

Future work includes defining the behavior of a guard and a proxy with respect to a specific application such as a network management protocol. As mentioned in the related work section, choosing a clear representation of complex requirements regarding guard behavior is also a part of future work.

REFERENCES

[1] J. Rushby, "The Design and Verification of Secure Systems," in Eighth ACM Symposium on Operating System Principles (SOSP), Asilomar, CA, 1981.

[2] L. J. La Padula and D. E. Bell, "Secure Computer Systems: A Mathematical Model," The MITRE Corporation, Bedford, MA, USA, 1973.

[3] CoNSIS, "System and Experimentation Architectures v1.0," 2011.

[4] S. J. Murdoch and S. Lewis, "Embedding Covert Channels into TCP/IP," in Information Hiding: 7th International Workshop, volume 3727 of LNCS, Barcelona, Spain, Springer, 2005.

[5] B. Pfitzmann, "Information Hiding Terminology - Results of an Informal Plenary Meeting and Additional Proposals," in Proceedings of the First International Workshop on Information Hiding, Springer-Verlag, 1996.

[6] R. Anderson, Security Engineering: A Guide to Building Dependable Distributed Systems - 2nd ed., Wiley, 2008.

[7] V. Gligor, "A Guide to Understanding Covert Channel Analysis of Trusted Systems," National Security Agency, Ft. George G. Meade, MD, USA, 1993.