An Integrated View of Security Analysis and Performance Evaluation: Trading QoS with Covert Channel Bandwidth

Alessandro Aldini and Marco Bernardo

Università di Urbino "Carlo Bo"
Istituto di Scienze e Tecnologie dell'Informazione
Piazza della Repubblica 13, 61029 Urbino, Italy
{aldini, bernardo}@sti.uniurb.it

Abstract. Security analysis and performance evaluation are two fundamental activities in the system design process, which are usually carried out separately. Unfortunately, a purely qualitative analysis of the security requirements is not sufficient in the case of real systems, as they suffer from unavoidable information leaks that need to be quantified. In this paper we propose an integrated and tool-supported methodology encompassing both activities, thus providing insights about how to trade the quality of service delivered by a system with the bandwidth of its covert channels. The methodology is illustrated by assessing the effectiveness and the efficiency of the securing strategy implemented in the NRL Pump, a trusted device proposed to secure the replication of information from a low-security level enclave to a high-security level enclave.

1 Introduction

Multilevel secure computing enforces data access control in systems where sensitive information is classified into access levels, and users are assigned clearances, such that users can only access information classified at or below their clearances. Such a controlled sharing of information is made harder by two aspects. On the one hand, the recent trends to open and distributed computing increase the vulnerability of network systems to attacks and of confidential data to information leaks [1]. On the other hand, the securing strategies used to improve the degree of system security must minimize each unavoidable information leakage without jeopardizing the quality of service (QoS) perceived by the users that are allowed to access data. Therefore, trading QoS with information leaks is of paramount importance in the system design process. This activity involves both security analysis and performance evaluation, two tasks that – unfortunately – are usually carried out separately.

In order to achieve a reasonable balance between QoS and security, in this paper we advocate the adoption of an integrated view of security and performance

[1] It is well known from real cases that it is not possible to eliminate in practice every unwanted covert channel [MK94,R+01,AG02,ABG03].

analyses. This is accomplished by proposing a tool-supported methodology that combines noninterference-based security analysis and performance evaluation on the same formal system description. Given a system, the integrated methodology we propose works as follows. The first step consists of providing a functional description of the system, on which a security check is applied in order to reveal all the potential nondeterministic covert channels from high-security level to low-security level. Such an analysis is based on a noninterference approach to information flow theory [GM82] and is essentially carried out through equivalence checking [FG95]. The unwanted covert channels that are captured by the security analysis are removed, if possible, by adequately changing the functional behavior of the system. Secondly, if some information leakage is revealed that cannot be removed, the bandwidth of each unavoidable covert channel is quantitatively estimated. This is carried out by enriching the functional description of the system with information about the temporal delays and the frequencies of the system activities. The second description considered in the methodology thus relies on a stochastic model that can be analyzed through standard numerical techniques or simulation [Ste94,Lav83]. The output of this performance analysis is given by the value of some relevant efficiency measures of the system together with the bandwidth of its covert channels, expressed as the amount of information leaked per unit of time. Such performance figures are then used as a feedback to properly tune the system configuration parameters in a way that lowers the covert channel bandwidth under a tolerable threshold without jeopardizing the QoS delivered by the system.

Although the proposed methodology is independent of the specific description language and companion tool – provided that the basic ingredients needed by the methodology itself are supplied – in this paper the application of the methodology is illustrated using the Æmilia description language [BDC02] and a suitably extended version of the related tool TwoTowers [AB04] that encompasses security analysis.

We exemplify the application of our methodology by means of a real case study: the Network NRL Pump [KMM98]. This is a trusted device used in multiple single-level security architectures to offer replication of information from low-security level systems (Low, for short) to high-security level systems (High, for short) with high-assurance security guarantees. Data replication is needed in this framework to minimize multilevel secure accesses to shared resources from processes at different security levels. Although at first sight illegal information leaks seem to be absent in a message passing from Low to High, some subtle behaviors must be paid attention to in order to prevent unauthorized users from obtaining access to confidential information. In fact, in order to offer reliable communications, an acknowledgement (ack) is usually required for each message that is successfully sent. The transmission of an ack from High to Low is more than enough to set up a covert communication channel if the timing of the ack is under the control of the High system. The NRL Pump, which basically acts as a delaying buffer between High and Low, makes it negligible such a timing covert channel (see, e.g., the security analysis conducted in [L$^+$04]) with a mi-

nor impact on the QoS. However, some information can still be sent from High to Low through the NRL Pump. This is due to the feedback forwarded by the pump to notify Low that a connection is up/down. In fact, High can manipulate the notification procedure to set up a 1-bit covert channel. To mitigate the effect of such an unavoidable covert channel, the NRL Pump architecture is designed in such a way that a minimum delay is enforced between connection setup and connection closing/abort and between the connection reestablishment and the auditing of any connection that behaves in a suspicious way. Therefore, the question is no longer whether the NRL Pump is secure, but how much data per unit of time can be leaked by exploiting the backward information flow. By applying our methodology, we formally verify that such an information leakage is the unique functional covert channel suffered by the NRL Pump. Then we provide useful information about the relation between the bandwidth of such a covert channel and the NRL Pump configuration parameters. In particular, we emphasize the impact of the NRL Pump securing strategy on the QoS delivered by the system, expressed as the number of connection requests that are served per unit of time.

The rest of the paper is organized as follows. In Sect. 2 we describe the case study, i.e. the NRL Pump, which will be used throughout the paper to illustrate our methodology. Sect. 3 introduces the Æmilia specification language together with a sketch of the formal description of the NRL Pump. In Sect. 4 we apply our methodology to the analysis of the NRL Pump model. Some concluding remarks are reported in Sect. 5.

2 An Overview of the NRL Pump

The NRL Pump is configured as a single hardware device that interfaces a high-security level LAN with a low-security level LAN. In essence, the pump places a buffer between Low and High, pumps data from Low to High, and probabilistically modulates the timing of the ack from High to Low on the basis of the average transmission delay from High to the pump. The low-level and high-level enclaves communicate with the pump through special interfacing software called wrappers, which implement the pump protocol (see Fig. 1). Each wrapper is made of an application-dependent part, which supports the set of functionalities that satisfy application-specific requirements, and a pump-dependent part, which is a library of routines that implement the pump protocol. Each message that is received and forwarded by the wrappers includes 7 bytes of header field, containing information about the data length, some extra header, and the type of message (data or control).

The pump can be considered as a network router. For security reasons, each process that uses the pump must register its address with the pump administrator, which is responsible for maintaining a configuration file that contains a connection table with registration information. The pump provides both recov-

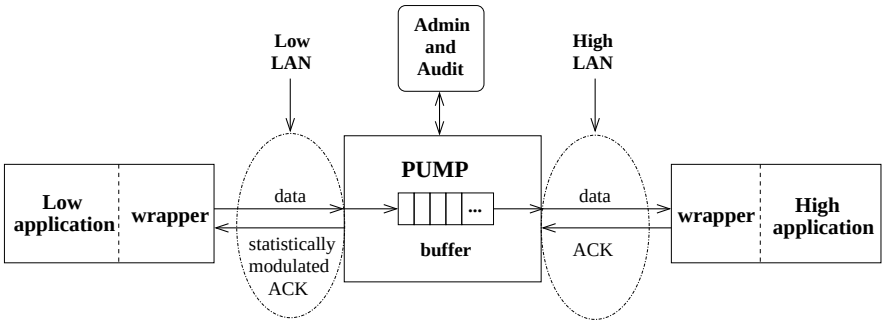

Fig. 1. Network NRL Pump architecture

erable and non-recoverable services[2]. Here, we concentrate on non-recoverable applications, like, e.g., FTP.

In brief, the procedure to establish a connection between Low and High through the pump is as follows. Initially, Low sends a connection request message to the main thread (MT) of the pump, which identifies the sending process and the address of the final destination. If both addresses are valid (i.e., they have been previously registered in the configuration file managed by the pump administrator), MT sends back a connection valid message, otherwise it sends a connection reject message. In the first case, the connection is managed by a trusted low thread (TLT) and a trusted high thread (THT), which are created during the connection setup phase to interact with Low and High, respectively. Registered High processes are always ready to accept a connection from the pump through the same handshake mechanism seen above. Once the new connection is established, the pump sends a connection grant message to both systems with initialization parameters for the communication. During the connection, TLT receives data messages from Low, then stores them in the connection buffer. Moreover, it sends back the acks (which are special data messages with zero data length) in the same order it receives the related data messages, by introducing an additional stochastic delay computed on the basis of the average rate at which THT consumes messages. On the other hand, THT delivers to High any data message contained in the connection buffer. The pump protocol also requires High to send back to THT the ack messages related to the received data messages. If High violates this protocol, THT aborts the connection. In such a case, as soon as TLT detects that THT is dead, it immediately sends all the remaining acks and a connection exit message to Low. Another special data message is connection close, which is sent at the end of a normal connection from Low to the pump.

In general, the pump is a reliable, secure, one-way communication device from Low to High, which minimizes the amount of (covert) communication in

[2] Recoverability safely assumes that any sent message will be delivered to the high system, even if connection failures occur.

the opposite direction. During the connection, only THT directly communicates with High and only TLT directly communicates with Low. Moreover, TLT and THT directly interact only through the connection buffer. However, even if the pump minimizes any timing covert channel from High to Low [L$^+$04], it cannot avoid some data leak in that direction. This is because the pump notifies Low when a connection is down. Such a feedback is more than enough to set up a 1-bit covert channel from High to Low. In the following, we formally verify the existence of such a covert channel and we measure its bandwidth and its relation with QoS in terms of number of requests that are served per unit of time.

3 Architecting Systems with Æmilia/TwoTowers

In order to illustrate an application of our integrated methodology, we need a formal specification language through which it is possible to produce precise system descriptions whose security and performance properties can both be verified by some automated tool. For this purpose here we employ Æmilia [BDC02], an architectural description language that is recalled in this section by presenting a sketch of the NRL Pump specification, and the companion software tool TwoTowers 4.0 [AB04], which has recently been extended to deal with both security analysis and performance evaluation.

3.1 Formal Modeling with Æmilia

Æmilia is an architectural description language based on the stochastic process algebra EMPA$_{\rm gr}$ [BB03]. A description in Æmilia represents an architectural type [BCD02], which is a family of systems sharing certain constraints on the component observable behavior as well as on the architectural topology. As shown in Table 1, the description of an architectural type starts with its name and its formal parameters, which can represent constants as well as exponential rates, priorities, and weights for EMPA$_{\rm gr}$ actions. Each architectural type is defined through its architectural element types (AETs) and its architectural topology. An AET, whose description starts with its name and its formal parameters, is defined through its behavior, specified as a list of sequential EMPA$_{\rm gr}$ defining equations, and its input and output interactions, specified as a set of EMPA$_{\rm gr}$ action names occurring in the behavior that act as interfaces for the AET. The architectural topology is specified through the declaration of a set of architectural element instances (AEIs) representing the system components, a set of architectural interactions given by some interactions of the AEIs that act as interfaces for the whole architectural type, and a set of architectural attachments among the interactions of the AEIs that make the AEIs communicate with each other. Every attachment must go from an output interaction of an AEI to an input interaction of another AEI. Given that every interaction is declared to be a uni-interaction, an and-interaction, or an or-interaction, the only legal attachments are those between two uni-interactions, an and-interaction and a uni-interaction, and an or-interaction and a uni-interaction. An and-interaction

and an or-interaction can be attached to several uni-interactions. In the case of execution of an and-interaction (resp. an or-interaction), it synchronizes with all (resp. only one of) the uni-interactions attached to it. The whole behavior of an Æmilia description is given by a family of EMPA_{gr} defining equations obtained by composing in parallel the behaviors of the declared AEIs according to the specified attachments. From the overall behavior, integrated, functional and performance semantic models can automatically be derived in the form of labeled transition systems, which can undergo equivalence verification, symbolic model checking, security analysis, reward Markov chain solution, and discrete event simulation.

ARCHI_TYPE	⟨name and formal parameters⟩
ARCHI_ELEM_TYPES	⟨architectural element types: behaviors and interactions⟩
ARCHI_TOPOLOGY	
ARCHI_ELEM_INSTANCES	⟨architectural element instances⟩
ARCHI_INTERACTIONS	⟨architectural interactions⟩
ARCHI_ATTACHMENTS	⟨architectural attachments⟩
END	

Table 1. Structure of an Æmilia description

We illustrate Æmilia by presenting a sketch of the formal specification of the NRL Pump. Due to lack of space, we do not show the full Æmilia specification of the NRL Pump, which can be retrieved from:

`http://www.sti.uniurb.it/bernardo/twotowers/`

The description starts with the name of the architectural type and its formal parameters with their initial values:

```
ARCHI_TYPE NRL_Pump_Type(const int buffer_capacity := n,
                         const rate conn_gen_rate := γ,
                         const rate conn_init_rate := η,
                         const rate data_trans_rate := δ,
                         const rate ack_trans_rate := κ,
                         const rate ack_delay_rate := θ,
                         const rate timeout_rate := μ,
                         const weight valid_prob := p)
```

The formal parameters represent the connection buffer capacity, the rates modeling some exponentially distributed delays, and the probability that a connection request is valid, respectively. In particular, `conn_gen_rate` is the Low connection request generation rate, `conn_init_rate` is the High connection initialization rate, `data_trans_rate` (resp. `ack_trans_rate`) is the data (resp. ack) message transmission rate, `ack_delay_rate` is the inverse of the stochastic delay added

by the pump to the transmission of the acks to Low, and timeout_rate is the inverse of the maximum amount of time that the pump waits for an expected ack.

The Æmilia specification of the NRL Pump then proceeds with the definition of the AETs. Below we report only the definition of the main thread type:

```
ARCHI_ELEM_TYPES
 ELEM_TYPE MT_Type(const rate data_trans_rate, const weight valid_prob)
    BEHAVIOR MT_Beh(void; void) =
       <receive_conn_request, _>.
          choice {
             <conn_is_valid, inf(1, valid_prob)>.<wakeup_tht, inf>.
                <send_conn_valid, data_trans_rate>.MT_Beh(),
             <conn_not_valid, inf(1, 1 − valid_prob)>.
                <send_conn_reject, data_trans_rate>.MT_Beh()
          }
    INPUT_INTERACTIONS UNI receive_conn_request
    OUTPUT_INTERACTIONS UNI wakeup_tht;
                            send_conn_valid;
                            send_conn_reject
          ⋮
```

The behavior of the main thread type is described through a single defining equation, which is built out of actions, action prefixes, choices, and behavior invocations, with every action being formed by an action name and an action rate expressing the inverse of the average duration of the action. The main thread monitors the port of the pump to which Low sends connection request messages, which is expressed through a passive action (rate _), then reacts to the reception of a connection request by verifying the validity of the received request. In order not to have to introduce a definition of the pump administrator, the reaction is abstractedly modeled by means of a choice between two immediate actions (rate inf) and their associated weights based on valid_prob, which expresses the probability of receiving a valid request. More precisely, in response to a request, either the main thread activates the trusted high thread and sends back a connection valid message with probability valid_prob, or it sends back a connection reject message with probability $1 - \texttt{valid\_prob}$. While the communication with the components outside the pump is assumed to take an exponentially distributed time, characterized by data_trans_rate, the communication delay within the pump is assumed to be negligible, hence the time to wake up the trusted high thread is approximated through an infinite rate. The definition of the main thread type is concluded with the declaration of some of the action names occurring in its behavior as being input or output interactions, which act as the interfaces of the main thread with the other components of the system.

Finally, the Æmilia specification of the NRL Pump contains the description of the system topology, in accordance with Fig. 1. Besides the declaration of all

the instances of the AETs, below we show only the attachments involving the interactions of the main thread instance:

```
ARCHI_TOPOLOGY
   ARCHI_ELEM_INSTANCES
      LW : LW_Type(conn_gen_rate, data_trans_rate);
      MT : MT_Type(data_trans_rate, valid_prob);
      THT : THT_Type(conn_init_rate, timeout_rate);
      TLT : TLT_Type(data_trans_rate, ack_trans_rate, ack_delay_rate);
      B : Buffer_Type(buffer_capacity);
      HC : High_Channel_Type(data_trans_rate, ack_trans_rate);
      HW : HW_Type()
   ARCHI_INTERACTIONS
   ARCHI_ATTACHMENTS
      FROM LW.send_low_conn_request TO MT.receive_conn_request;
      FROM MT.wakeup_tht TO HW.receive_high_wakeup;
      FROM MT.send_conn_valid TO LW.receive_conn_valid;
      FROM MT.send_conn_reject TO LW.receive_conn_reject;
            ⋮
END
```

4 Integrating Security and Performance Analyses

In this section we illustrate the use of our methodology by assessing the existence, the bandwidth, and the relation with QoS of an unavoidable covert channel in the NRL Pump. For the sake of simplicity, since the amount of data sent from Low to High does not alter the kind of communications between Low and High through the NRL Pump, we considered a system configuration where Low tries to establish a connection during which a single message is sent to High. After the transmission of the message either the connection is correctly closed or it is aborted by the pump. As a consequence, we assume that the pump buffer has capacity $n = 1$. The results we obtained are summarized as follows:

- The noninterference-based security analysis reveals the existence of a covert channel caused by a connect/disconnect strategy. Diagnostic information is also provided to detect the functional behavior of the NRL Pump that is responsible for the information leakage.
- Two metrics that are strictly related to the connect/disconnect strategy are evaluated. The result of this analysis is an estimation of the covert channel bandwidth and its relation with the NRL Pump configuration parameters.

In the following, we describe the noninterference property we checked and we formally show that the success/failure of a connection can be coded into a 1-bit value. Then, we specify some important assumptions that we made about the network scenario and the temporal behavior of the pump. Based on such assumptions and on the nature of the covert channel, we measured the information leakage through some suitable metrics.

4.1 Noninterference Analysis

The application of formal methods to the analysis of security properties (see, e.g., [Mea03] and the references therein) is a well-established approach accepted by the security community. In particular, we employ a technique based on the idea of nondeterministic noninterference [GM82]. Basically, supposing that low-security level users observe public operations only and high-security level users perform confidential operations only, an interference from High to Low occurs if what High can do is reflected in what Low can observe. The security check we apply verifies whether the Low view of the system behavior in the absence of High interferences is the same as that observed when High interacts with the system (see the Strong Nondeterministic Noninterference property of [FG95]). Formally, we divide actions into high-level and low-level actions, denoted *High* and *Low*, respectively, depending on the nature of the activities they represent. Then, from the functional model of a system P we derive two models that express the Low views specified above. On the one hand, the view of P without High operations, denoted $P \backslash High$, is obtained by preventing P from executing high-level actions. On the other hand, the low-level view of P with High interactions, denoted $P/High$, is obtained by turning all the high-level actions into invisible actions, since Low is not expected to observe them. Finally, the models we obtain are compared through equivalence checking. To this aim, the notion of equivalence relation we consider is the weak bisimulation equivalence [Mil89], which captures the ability of two processes to simulate each other behaviors up to invisible actions. If the equivalence check is satisfied, then Low cannot infer the behavior of High by observing the public view of the system, that means the system does not leak information from High to Low. The security analyzer of TwoTowers 4.0 allows the software architect to describe in an auxiliary specification file which actions belong to *High* and which belong to *Low*. All the other actions are simply disregarded by turning them into invisible actions.

As far as the NRL Pump is concerned, the low-level view of the system is represented by the communication interface between the Low wrapper and the pump, which interact through low-level actions. Analogously, all the actions modeling communications between the High wrapper and the pump are high-level actions. All the actions modeling communications among the internal components of the pump (like, e.g., the synchronizations between MT and THT, or between TLT and the buffer) are internal activities of the NRL Pump, which cannot be seen by an external observer. Therefore, as far as the security check is concerned, it is reasonable to assume that they are invisible.

Then, the security analyzer of TwoTowers 4.0 derives the models to be compared from the functional model of the Æmilia specification of the NRL Pump, which we call `NRL_Pump_Type`$\backslash$*High* and `NRL_Pump_Type`$/$*High*, and performs the weak bisimulation equivalence check. The obtained result is that they cannot be weakly bisimulation equivalent. The distinguishing modal logic formula returned by TwoTowers 4.0 intuitively shows what follows: `NRL_Pump_Type`$\backslash$*High* aborts all the connections (each connection terminates with the occurrence of the low-level action modeling the transmission of a connection exit message),

while NRL_Pump_Type/*High* is able to close connections between Low and High (a connection may terminate with the occurrence of the low-level action modeling the transmission of a connection close message). The related covert channel is caused by the notification feedback from the pump to Low. Indeed, if we prevent Low from observing the result of each connection (by hiding the low-level actions modeling the connection close/exit message) we obtain that the system turns out to be secure. That means the covert channel described above is the unique nondeterministic information leakage that occurs in the NRL Pump.

4.2 Performance Analysis

By following the second step of our methodology, the bandwidth of an unavoidable covert channel revealed by the security analysis is measured by evaluating some relevant efficiency measures. For this purpose, action durations are taken into consideration. In particular, in the case of the Æmilia specification of the NRL Pump, in Sect. 3.1 we have shown that some delays, such as timeouts and transmission times, are modeled as stochastic random variables governed by exponential distributions. Thus, the stochastic model we obtain is a continuous-time Markov chain. To derive performance measures of interest, such Markov chain can be analyzed by the performance evaluator of TwoTowers 4.0 through standard numerical techniques. To this aim, the user describes in an auxiliary specification file the rewards to be attached to specific actions of the Æmilia description. These rewards are then exploited to compute reward-based metrics, such as throughput and utilization measures.

As far as the NRL Pump is concerned, the security check revealed that the unique information leakage from High to Low is given by the occurrence of a connection exit event (in case High is absent) with respect to the occurrence of either a connection exit event or a connection close event (in case High is present). Hence, the number of connections that can be closed/aborted because of the behavior of High represents an estimate of how many bits High can pass to Low in a certain period. Formally, such an estimate is obtained by measuring the throughput of the low-level actions modeling the transmission of the connection close and the connection exit messages that are observed by Low.

Before showing the analysis results, we explain some assumptions about the timing of the actions occurring in the Æmilia specification of the NRL Pump. All the delays are exponentially distributed with a certain rate expressed in sec^{-1}. The data (resp. ack) transmission rate and the round-trip propagation rate experienced during the connection setup phase between the pump and High are δ (resp. κ) and η. We assume that the pump uses two 64 Kbps full-duplex lines and the (mean) length of data (resp. ack) messages is 512 (resp. 49) bits, so that $\delta = 125$ (resp. $\kappa = 1306.12$) and $\eta = 62.5$. The connection request generation rate γ varies in the range $[1, 1000]$, i.e. from one request per sec to one request per ms. The rate of the stochastic delay added by the pump before sending the ack to Low is θ. We assume such a delay to be equal to the

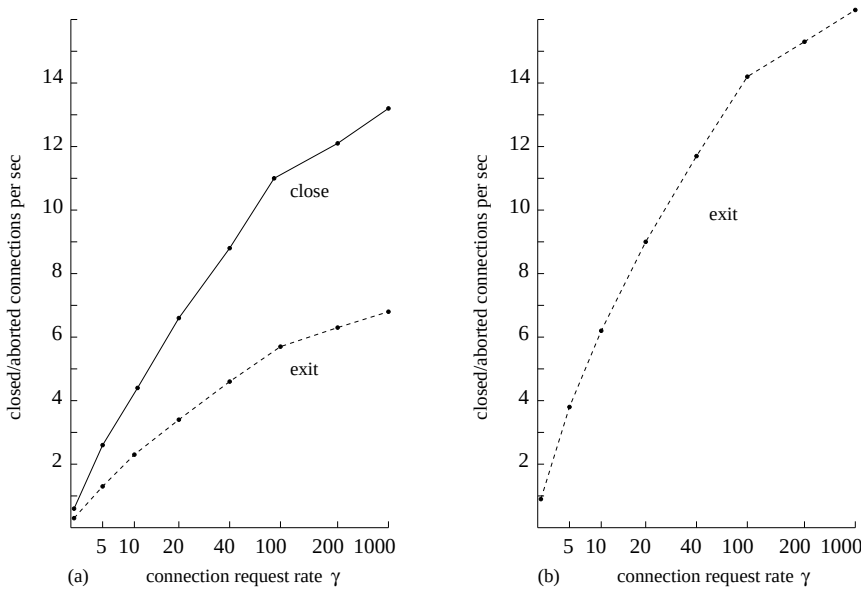

Fig. 2. Throughput of closed/aborted connections with and without High

transmission time of three ack messages [3], so that $\theta = 435.37$. The timeout delay used by the pump when waiting for the ack from High varies from 200 to 10 ms. Therefore, the corresponding rate, denoted μ, varies in the range $[5, 100]$. Finally, for each connection request we abstract from the configuration file look-up and we assume that each incoming request is valid with probability $p = 0.99$.

Fig. 2 reports the number of connection close/exit messages observed per sec in the case $\mu = 57.04$, corresponding to double the average time needed to send a data message and to receive the related ack (i.e., about 17 ms). Fig. 2(a) refers to the scenario in which High correctly executes the protocol. Therefore, most connections are normally closed, while aborted connections can occur because of the expiration of the timeout set by the pump. Fig. 2(b) refers to the scenario in which High is absent, i.e. all the connections abort. For both scenarios, we have that as the connection request rate γ increases, the number of closed/aborted connections increases as well. Note that abortions occur in both figures independently of the behavior of High. As a consequence, a connection exit message cannot reveal the presence/absence of High. Instead, Low deduces the presence of High if a connection is correctly closed, which is an event that occurs in Fig. 2(a) only. In particular, from Fig. 2(a) we derive that High succeeds in leaking its presence to Low up to 13 times per sec. Finally, note that the difference between the curve of Fig. 2(b) and the corresponding curve of Fig. 2(a) shows that the number of aborted connections observed per sec is appreciably altered by the

[3] This is long enough to hide the fluctuations of the transmission delays of the ack messages propagating from High to the pump.

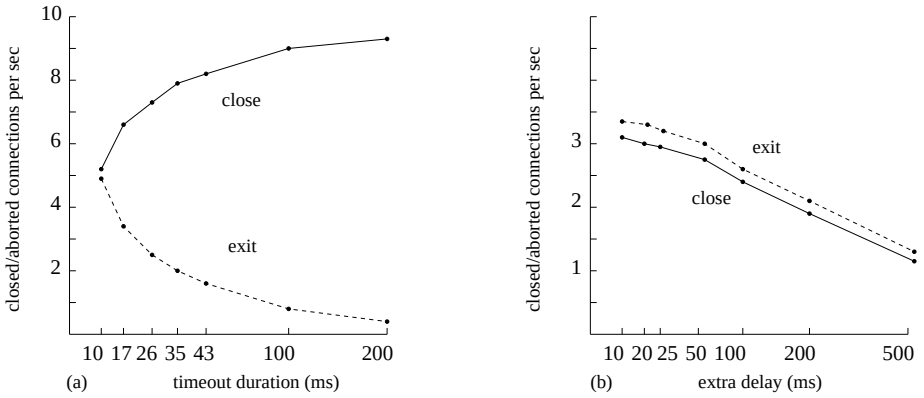

Fig. 3. Throughput of closed/aborted connections for different scenarios

absence of High. That means Low can deduce the presence of High by simply measuring the number of connection exit messages received per sec.

The number of connections that abort because of the timeout expiration can be limited by increasing the timeout duration. In Fig. 3(a) we show the tradeoff between the timeout duration and the pump throughput in terms of number of connections served per sec. In particular, we consider a scenario where both Low and High correctly execute the protocol, $\gamma = 20$ (corresponding to a connection request every 50 ms), and the timeout duration varies in the interval $[10, 200]$ ms (i.e., μ varies from 100 to 5). The curves show that as the timeout duration increases, the number of connection exit messages tends to zero, while the number of connection close messages rises up to 9 per sec. The most interesting result is that in the limiting scenario where the timeout expires after 200 ms, it is very likely that an ack sent by High arrives before the expiration of the timeout. In such a case, since an aborted connection does not occur because of the timeout expiration, High may exploit the connection exit message to leak a bit to Low. Indeed, the timeout is long enough to assure that its expiration is caused by a misbehavior of High. In other words, each connection really leaks a bit from High to Low (e.g., 0 if it succeeds and 1 if it fails). In order to measure the bandwidth of such a 1-bit covert channel, we consider a limiting scenario where $\gamma = 200$ (corresponding to a connection request every 5 ms), $\mu = 5$ (i.e., the timeout duration is 200 ms), and High alternatively completes and blocks (with equal probabilities) the connections in order to express a sequence of bits to be sent to Low. In this case, we obtain that about 3.14 connections are closed and 3.42 connections are aborted per sec, that means about 6 bits per sec are leaked from High to Low.

In general, the pump designer can quantitatively assess the relation between the amount of bits leaked from High to Low and the value of each configuration parameter that influences the QoS delivered by the pump. In particular, we have seen that covert channel bandwidth and pump throughput (in terms of number

of connections served per sec) are directly proportional. A strategy to reduce the covert channel bandwidth consists of enforcing a minimum delay to elapse between subsequent connection establishments. Consider, e.g., the addition of an extra delay, exponentially distributed with rate λ, after the abortion of a connection and before its reestablishment. Then, let us evaluate the effect of this extra delay by considering the same limiting scenario as above (i.e., $\gamma = 200$, $\mu = 5$, and High tries to alternatively abort and complete a sequence of connections). The formal verification of the NRL Pump in such a scenario produces the results depicted in Fig. 3(b), which are obtained by varying the extra delay from 500 to 10 ms (i.e., $\lambda \in [2, 100]$). Note that as the artificial delay increases, the total number of closed/aborted connections per sec decreases from the upper bound of about 6 per sec to a lower bound of about 2 per sec. As a consequence, since each connection leaks a bit, we have that the covert channel bandwidth decreases, in spite of a reduction of the QoS in terms of requests served per sec. In practice, a tradeoff exists between the robustness against the 1-bit covert channel and the QoS delivered by the NRL Pump. To reduce the unfavorable impact of the proposed strategy on the QoS, which could be unacceptably burdensome, the pump should carefully activate the extra delay, e.g. only in the case of frequent abortions, which are an evidence of the misbehavior of High.

5 Conclusion

In this paper we have presented an integrated methodology – implemented through the Æmilia/TwoTowers technology – that combines noninterference analysis and performance evaluation in order to trade QoS with covert channel bandwidth.

On the one hand, the need for both qualitative and quantitative security assessment stems from the fact that real systems like the NRL Pump suffer from unavoidable information leaks, which have to be quantified in order to estimate the degree of system security. On the other hand, performance evaluation allows for a quantitative estimation of the efficiency (and the impact on the QoS) of the securing strategies implemented to reduce the covert channel bandwidth.

In general, the application of such a methodology can represent an effective support to validating the security guarantees of real systems while preserving the expected QoS. For instance, audio/video applications based on real-time channels require both critical QoS constraints and privacy guarantees. Such applications often offer customized security (choice of the authentication and privacy methods, tolerance to replay attacks, use of caching and prefetching strategies) to achieve a customized tradeoff between security and performance, which can be formally analyzed and supported by the use of our methodology.

As a future work, it would be interesting to extend the methodology to consider not only nondeterministic covert channels, but also interferences caused, e.g., by possible probabilistic aspects of the system behavior [ABG03].

References

[AB04] A. Aldini and M. Bernardo. TwoTowers 4.0: Towards the Integration of Security Analysis and Performance Evaluation. *1st Int. Conf. on Quantitative Evaluation of Systems* (QEST'04), IEEE CS Press, to appear, 2004.

[ABG03] A. Aldini, M. Bravetti, and R. Gorrieri. A Process-algebraic Approach for the Analysis of Probabilistic Noninterference. *Journal of Computer Security* 12(2), 2004.

[AG02] A. Aldini and R. Gorrieri. Security Analysis of a Probabilistic Nonrepudiation Protocol. *2nd Int. Work. on Process Algebra and Performance Modelling, Probabilistic Methods in Verification* (PAPM-ProbMiV'02), Springer LNCS 2399:17–36, 2002.

[BB03] M. Bernardo and M. Bravetti, Performance Measure Sensitive Congruences for Markovian Process Algebras. *Theoretical Computer Science* 290:117-160, 2003.

[BCD02] M. Bernardo, P. Ciancarini, and L. Donatiello. Architecting Families of Software Systems with Process Algebras. *ACM Trans. on Software Engineering and Methodology* 11:386-426, 2002.

[BDC02] M. Bernardo, L. Donatiello, and P. Ciancarini. Stochastic Process Algebra: From an Algebraic Formalism to an Architectural Description Language. *Performance Evaluation of Complex Systems: Techniques and Tools*, LNCS 2459:236-260, 2002.

[CLS00] W.R. Cleaveland, T. Li, and S. Sims. *The Concurrency Workbench of the New Century - Version 1.2 - User's Manual.* `www.cs.sunysb.edu/~cwb/`, 2000.

[FG95] R. Focardi and R. Gorrieri. A Classification of Security Properties. *Journal of Computer Security* 3:5-33, 1995.

[GM82] J.A. Goguen, J. Meseguer. Security Policy and Security Models. *Symposium on Security and Privacy* (SSP'82), IEEE CS Press, pp. 11-20, 1982.

[KMM98] M.H. Kang, A.P. Moore, and I.S. Moskowitz. Design and Assurance Strategy for the NRL Pump. *NRL Memo* 5540-97-7991, Naval Research Laboratory, Washington, D.C., 1997, appeared in *IEEE Computer Magazine* 31:56–64, 1998.

[L$^+$04] R. Lanotte, A. Maggiolo-Schettini, S. Tini, A. Troina, and E. Tronci. Automatic Analysis of the NRL Pump. To appear in ENTCS, *Selected Papers from MEFISTO project "Formal Methods for Security"*, 2004.

[Lav83] S.S. Lavenberg editor. *Computer Performance Modeling Handbook.* Academic Press, 1983.

[Mea03] C. Meadows. What Makes a Cryptographic Protocol Secure? The Evolution of Requirements Specification in Formal Cryptographic Protocol Analysis. *12th Europ. Symp. on Programming Languages and Systems* (ESOP'03), Springer LNCS 2618:10–21, 2003.

[Mil89] R. Milner. *Communication and Concurrency.* Prentice Hall, 1989.

[MK94] I.S. Moskowitz and M.H. Kang. Covert Channels – Here to Stay? *9th Conf. on Computer Assurance* (Compass'94), National Institute of Standards and Technology, pp. 235–244, 1994.

[R$^+$01] P.Y.A. Ryan, J. McLean, J. Millen, and V. Gligor. Non-interference: Who Needs It? *14th Computer Security Foundations Workshop* (CSFW'01), pp. 237–238, IEEE CS Press, 2001.

[Ste94] W.J. Stewart. *Introduction to the Numerical Solution of Markov Chains.* Princeton University Press, 1994.