Covert Channels Detection using Process Mining

Amir Jalaly Bidgoly, Behrouz Tork Ladani, Ahmad Baraani Dastjerdi
Department of Computer Engineering,
Faculty of Engineering, University of Isfahan,
Isfahan, Iran

Abstract—**Covert channel is a communication channel between two processes that is neither designed nor intend for communication. Since it is very hard to detect covert channels, they are an important security challenge for current systems and may be used for illegal data transfer by malwares. There exist two main approaches for finding covert channels. First identifying potential covert channels and second detecting currently used covert channels. In this paper, two methods are proposed for covert channel detection based on process mining. Whereas most of existing methods try to identify potential covert channels, the proposed methods detect currently using ones. In both methods, first the process model is extracted using process mining and then covert channel patterns are searched in the model. In the first proposed method, both high and low processes should be available whereas in the second one only high process is required for detection. The efficiency and accuracy of the proposed methods is demonstrated by a simple but clear case study.**

Index Terms—**Covert channel, Process mining, Process model, Temporal logic.**

I. Introduction

Covert channel is a type of information leak from a higher classification process to a lower one that the communication channel used by the processes is neither designed nor intended to transfer information[1]. This characteristic of covert channels makes them very hard to detect, therefore they are often hidden from the security and design viewpoint. Covert channels are an important challenge for security of current systems since it enables malicious process to transfer top-secret information to a lower class process.

Nowadays, in almost security documents, there exist standards to overcome covert channels or reduce their risk [2][3][4]. Five base approaches exist for this purpose: Identification, Detection, Analysis, Prevention and Migration. Many methods are proposed in the literatures based on these approaches and we have reviewed some of famous ones in the related works section. However these methods suffer from two common weaknesses: First, They are not an automated and need an expert supervisor. Second, however the methods have shown their effectiveness in theory but in practice they are not so useful, because they consider a very limit or abstract system which is very far from the reality.

In this paper two novel methods using process mining are proposed for covert channel detection. *Process Mining* [6][7] is a tool for extracting information from event logs that usually presented as a process model. Process mining enables us to have a quit real model from running process, therefore in practice the performance of the proposed methods is greatly enhanced. On the other hand, unlike other existing methods which are more intended to identify potential channels, the proposed methods is looking for a way to detect currently using covert channels.

The reminder of the paper is organized as follow: in next section related works have been reviewed. Motivation and contribution of the proposed method is presented in section three. In section four, process mining is reviewed in summary. The proposed methods are described in section five, and a case study is presented in section six. Finally the paper ends with conclusion and future works in section seven.

II. Related Works

As mentioned before, there are five approaches in the field of covert channels: *Identification*, *Detection*, *Analysis*, *Prevention* and *Migration*. In covert channel identification, all potential covert channels will be identified. Identified cases may not currently used by any malware, but they are capable to be used as a communication media between malwares later. In identification, the goal is to analyze of system design and structure to eliminate and avoid possible covert channel attacks as much as possible. *Shared Resource Matrix* [7][8] and *Covert Flow Tree* [9] are two famous methods of this class.

Covert channel detection as the second approach is not to find all potential channels but just the currently using one. In most systems, identification will find many potential covert channels that are hard or even impossible to be eliminated. So in practice, detection is a better approach then identification, but, in literatures the former methods gains less interest than the later and there are not many methods for covert channels detection. [10][11][12] are some examples of these type of methods.

The third approach, covert channel analysis, helps us in

different way. In this approach, the capacity of channel will be analyzed to determine how dangerous it be if used by malwares. For example a 1 bit per second covert channel is not a big security challenge but a 1000 bits per second one could make important security problems. Most of security documents say that eliminating of all covert channels is not possible [13]. For example, DoD documents described for securing a system there must exist no more then 1 bit per second covert channel. Covert channel analysis is a tool for this mean. [14] is an example method of these approach.

Another approach in this field is covert channel prevention. This approach is trying to separate and distinct processes of a certain security class from other processes. This means a higher process could neither communicate nor be observed by lower process. *Virtual Machines* and *Sand Box* [15] are two important methods in this approach. However prevention methods are usually impractical since eliminating all communication media and the shared resource is impossible and hinders normal behavior of process.

The last approach is *Migration*. Migration methods are used when there is a covert channel that cannot be eliminated. Here the goal is to reduce the risk of channel (i.e. decrease channel capacity). Pump [16] is the famous tool for migration however these methods usually reduce system performance due to the changes in the allocation of the shared resources.

In next section weakness points of above methods are discussed and the motivation of the proposed methods is presented.

III. Motivation

Identification and detection of covert channels are distinguished between other mentioned approaches, due to the fact that for analysis, migration and prevention of covert channel, first we need to identify or detect them. This shows the importance of identification and detection. On the other point of view, in many cases, identification of a covert channel is not very helpful, because in any system that needs shared resource or communication between processes, various types of covert channels could be identified. In practice, eliminating all covert channels is possible only when two conditions could be met [13]: First, the process cannot be observed or monitored by any other process of lower class and second there is no communication media or shared resource between processes. The above assumptions are often not applicable. So identification is not an effective approach for covert channels.

Detection approach, unlike identification, is to find covert channels which are currently used by malwares. This approach has several advantages compared to identification. Detecting currently using covert channels not only enables us to eliminate them but also help to detect/remove malwares from the system.

Despite to the advantages of detection approach, it needs to analysis running processes. To analysis running processes, their specifications and documentations are required which are often not available or accessible. Even in the cases where a specification or document is provided, if the process uses a covert channel, the specification is reorganized in a way that makes it difficult or impossible to discover the covert channel. Who give a document that say "I am a covert information thief"?. It can be said that in the most cases either there is no documentation or if documentation is provided, it is different from reality (i.e. have been manipulated to mislead). Lake of specifications for running processes is a big challenge for detection approach.

Considering above, in this paper a novel method is presented for detecting of covert channels based on process mining. Process mining provides a real (or at least close to reality) model of the process. Using this method, the process are not be black box anymore, thus a detection method can be applied. In next section a brief description of process mining is given before describing the proposed methods.

IV. Process Mining

Process mining (PM) [5][6] is a tool to discover information about a process from its event logs. It can be used when no specification is available about a process or there are doubts about correctness of a given specification. There are three different classes in PM. First class involves the algorithms for extracting the process model from collected process logs [17]. The model is usually shown by a standard formalism like Petri nets. In addition process mining may be used for extracting other types of knowledge like social networks.

The second class in PM is used when the question is about the correctness of a given process model [18]. In this class, PM helps to check the matching of a provided model with recorded event logs. In addition, it helps predict or suggest next step in the running of the process. Intrusion Detection Systems (IDS) that work based on anomaly activity detection are examples of this class.

In the third class of PM, whole or a special part of the given model is improved considering event logs. This class is to enhance a property (e.g. the fitness) of the process model.

Many algorithms and useful tools are proposed for PM. Alpha [18], Genetic miner [19] are some famous algorithms and ProM [21] is one of the best tool in this regards.

V. Proposed Methods

In this section, our proposed methods are described. We have two methods for detecting covert channels based on PM called simple and advanced methods. In both, we need to collect process logs and extract the model. Since covert channels are often created via shared resources and system functions; events that are recorded for each process are events that are related to system function calls and events related to an inner state or process depend transactions are discarded. These two methods are presented in continue.

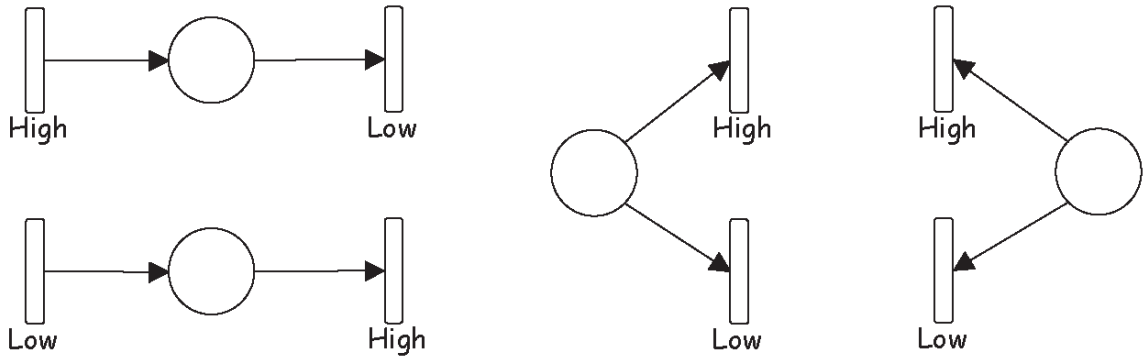

Fig. 1. Covert channel patterns in simple the method

A. Simple Method

Suppose there are two processes from different security classes in the system. The process from higher class called *High* involves access to sensitive information and the other one called *Low* does not permit to access them. A mandatory access control system checks security policies including these two processes should have no contact with each other. As mentioned before, a covert channel may allow them to have a hidden communication channel while they are explicitly prohibited from. Here the goal is to find out if there is any covert channel between *High* and *Low* or not.

If a relationship between the processes be found, it can be concluded that there is a covert channel between them because they should have no contact with each other due to the access control system. Simple method detects covert channel based on this principal. For this purpose, events in addition to the performer of each event (i.e. *High* or *Low*) are recorded and then the process model is extracted using process mining. If there is a relationship between the processes in the obtained model, the existence of covert channel can be derived.

The term "relationship between processes" should be described. Here, it is assumed that process model is presented using Petri nets. To identify relationship between processes, some patterns can be defined in Petri nets. These patterns are shown in figure (1) and include following ones:

- A token generated by *High* process enables a transaction in *low* process or vice versa.
- Events of two different processes have a shared input place.
- Events of two different processes have a shared output place.

After presenting process model by Petri net, it's required to search for above patterns. If one of the patterns is found, existence of covert channel between processes can be concluded. In section six, a case study is presented for this method. Although this method is effective and simple, but it also has some weakness points. In continue we have reviewed these weakness points and then the advanced method is introduced.

B. Advanced Method

In simple method, two conditions should be met before applying the method: First, both *High* and *Low* process or at least their recorded event logs must be available. Second, the processes should be independent. This means no relationship between them is allowed except when a covert channel exits. In some types of covert channel, above assumption are not true. For example, if an unreachable shared resource is used for creating covert channel (e.g. network resources); its event logs cannot be recorded. When *Low* process is outside of system boundary or it is a virus hidden in system files, again its event logs cannot be collected. So the extraction of the process model that includes both *High* and *Low* process is not always possible, thus the first condition is not always true. Also about the second condition, the processes may allow to have a limited and controlled connection. For example *Low* process injects itself as a virus to a trusted process. Here, the model contains some relationship stands for contact or communication between them whereas it is not clear that it is a legal or illegal data transfer. Therefore in many cases mentioned conditions are violated.

Considering the mentioned problems, advanced method is introduced. In this method is assumed that only *High* process is available thus the process model just includes and be extracted with its logs. Then this model is searched to find out if there is a covert channel or not.

In each covert channel there exist at least two signals to send 0 and 1 respectively. If these signals are created using a shared resource, a special pattern can be found in the model. This pattern is shown in figure (2). As it can be seen, three steps are required to send a signal (zero or one). First *High* process makes a change in a shared resource. We call this shared resource "high signal resource" (i.e. var_i in fig. (2)). High signal resource must be observable by *low* process so it will notice the change. After it find a change, *low* process makes a change in a so-called "low signal resource" (i.e. var_j in fig. (2)) that means it received *high* signal. Low and high signal resources can be same. After *high* process receives *low* signal via low signal resource, it goes to the third step and prepare for sending next bit. Since the process model is

obtained of event logs of system function call, it is needed to specify the shared resources in each event. These resources are the hidden system variables which are changed in system function calls. In the case study, file existence flag is the shared resource used for creating covert channel.

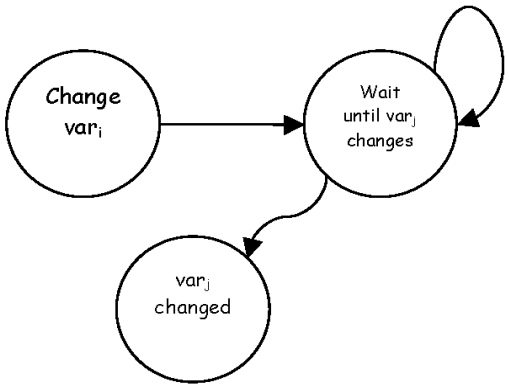

Fig. 2. Covert channel pattern in the advanced method.

In large and complex models, the given pattern may not be discovered simply. In some case given pattern is separated between irrelevant events that make it difficult to find. Hence to complete this pattern, temporal logic can be used. This not only led to a more accurate diagnosis but also makes detection process automated without need to an expert supervisor. The LTL pattern formula is shown below:

$$\exists F[change(\mathrm{var}_i, x) \quad X(read(\mathrm{var}_j, \neg y) U read(\mathrm{var}_j, y))] \tag{1}$$

In above equation, var_i and var_j refer to high and low signal resources respectively. x and y are the changed state made by the processes.

VI. Case Study

In this section a simple case study is presented to clarify the efficiency of the proposed methods. The case study involves a covert channel simulation implemented by the programming language (C#). In addition, *Prom* [21] and *PromImport* [22] is used for the process mining jobs and *PDETool* [23] helps to verify the model against the given patterns.

The simulation program includes two threads as *low* and *high* processes in addition to a third one that has no role in covert channel and just makes logs noisy. The shared resource is a file management system. Four files exist in this system: *A*, *B*, *C*, and *D*. To send a signal, first *high* process creates file *A*. *Low* process monitors the file system to discover this change. If *low* detect *A* is created, it removes it from the file system. When the file is deleted, *High* notice that *low* has received the signal. This means *high* send 0 to *low*. Sending 1 is exactly in the same way but file *B* is used instead of *A*. *C* and *D* are to make logs noisy and each one of three threads may use them randomly.

After running threads, event logs are collected. These events include *CreateFile*, *DeleteFile*, and *CheckFile* for creating a file, deleting and check the existence of a file respectively. For each event, the performer of event (i.e. *High*, *low*, and *noise*) in addition to corresponding file name (i.e. *A*, *B*, *C*, and *D*) are also recorded. These data are converted to MXML format (using PromImport) as the input data for Prom. Then the process model is extracted using two different algorithms; Alpha [17] and ILP [24]. These models are shown in figure (3). The transaction of *high* process is colored in red and the transaction of *low* process is colored in blue. Applying the simple method, some relationship can be found in the models. These relationships are marked in red zone that means a covert channel is detected using the method.

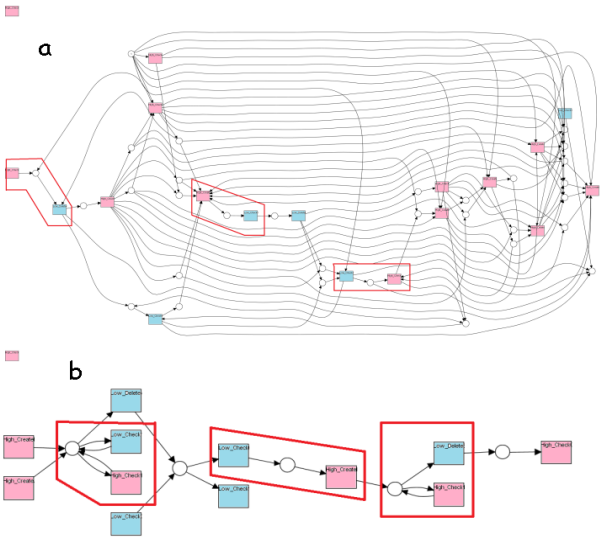

Fig. 3. The process model of case study of the simple detection method **a)** Alpha algorithm **b)** ILP algorithm

For testing the advanced method, another instance of simulation program is executed and the event logs of just *high* process are collected. As before the process model is extracted using Prom and PromImport which is shown in figure (4). Again the models are searched for given patterns. The model obtained by Alpha algorithm does not contain the pattern however the model obtained by ILP has corresponding pattern that is marked in red zone. It seems that Alpha algorithm is a primitive algorithm, thus it has some weakness that make the extracted model incomplete, whereas ILP as the more advanced algorithm discovers pattern simply. This clears the importance of used algorithm and fitness of extracted model. The models are also checked for the given LTL formula using PDETool and results confirm the existence of covert channel again.

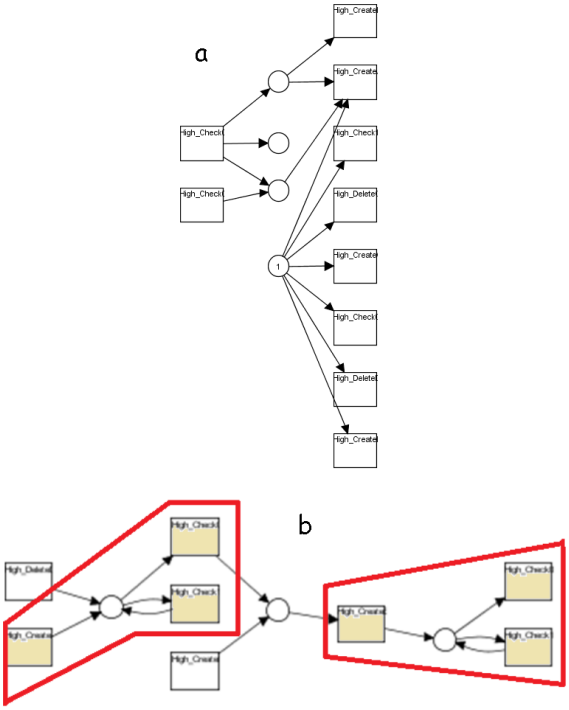

Fig. 4. The process models of the advanced detection methods **a)** Alpha algorithm **b)** ILP algorithm

For having a complete case study, a third instance of simulation is executed with no covert channel and threads using files randomly and completely independent. The goal is to check the proposed method whereas no covert channel exists. The extracted process model is shown in figure (5). As seen in this model there is neither simple nor advanced method patterns. This is a good example of efficiency and accuracy of the proposed methods.

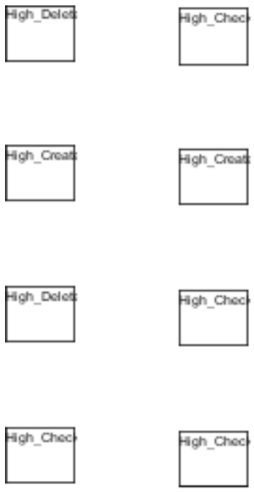

Fig. 5. The process model when no covert channel exists

VII. CONCLUSION AND FUTURE WORKS

In this paper, two methods are proposed for detecting covert channels which are based on process mining. In the proposed methods, first the event logs of processes are collected and then the process model is extracted using process mining and collected logs. In the simple detection method is assumed that both high and low processes (that intends to use covert channel) are available and should have no contact with each others. In the advanced detection method only high process and its event logs is required to detect covert channel. This method is also presented with LTL logic.

The proposed methods cannot detect time covert channels which use a sequence of events for data transfer. This could be considered in the future development. On the other hands, some other behaviors different from covert channel behaviors may create patterns discussed in this paper. In future, we should also identify and distinguished these types of behaviors.

REFERENCES

[1] B. W. Lampson, "A Note on the Confinement Problem," Communications of the ACM, 16:10, pp. 613-615, October 1973.

[2] J. McHugh. Handbook for the Computer Security Certication of Trusted Systems, chapter Covert Channel Analysis, pages 4-78. US Navy, 1995.

[3] NCSC-TG-030, Covert Channel Analysis of Trusted Systems (Light Pink Book) from the United States Department of Defense (DoD) Rainbow Series publications.

[4] 5200.28-STD, Trusted Computer System Evaluation Criteria (Orange Book) from the DoD Rainbow Series publications.

[5] W. M. P. van der Aalst, and A. Weijters (2004). "Process mining: a research agenda." Computers in Industry **53**(3): 231-244.

[6] Process Mining: Discovery, Conformance and Enhancement of Business Processes, Springer Verlag, Berlin (ISBN 978-3-642-19344-6).

[7] R. Kemmerer, "A Practical Approach to Identifying Storage and Timing Channels," Proceedings of the 1982 IEEE Symposium on Security and Privacy, pp. 66–73 (Apr. 1982).

[8] R. Kemmerer, "Shared Resource Matrix Methodology: An Approach to Identifying Storage and Timing Channels," ACM Transactions on Computer Systems, 1 (3), pp. 256–277 (Aug. 1983).

[9] P. Porras and R. Kemmerer, "Covert Flow Trees: A Technique for Identifying and Analyzing Covert Storage Channels," Proceedings of the 1991 IEEE Symposium on Security and Privacy, pp. 36–51 (May 1991).

[10] R. Accorsi, and C. Wonnemann (2011). Strong non-leak guarantees for workflow models, ACM.

[11] V. Berk, A. Giani, G. Cybenko, "Covert channel detection using process query systems." In: Proc. of FLOCON 2005. (Sep. 2005).

[12] V. Berk, A. Giani, G. Cybenko, "Detection of covert channel encoding in network packet delays." Technical Report TR2005-536, Department of Computer Science, Dartmouth College, Hanover, NH., USA (Aug. 2005).

[13] N. Proctor and P. Neumann. Architectural implications of covert channels. In National Computer Security Conference, pages 28 43, 1992.

[14] J. Millen, "Covert Channel Capacity," Proceedings of the 1993 IEEE Symposium on Research in Security and Privacy, pp. 60–65 (May 1993).

[15] I. Goldberg, D. Wagner, R. Thomas, and E. Brewer, "A Secure Environment for Untrusted Helper Applications: Confining the Wily Hacker," Proceedings of the 6th USENIX Security Symposium, pp. 1–13 (July 1996).

[16] M. Kang and I. Moskowitz, "A Pump for Rapid, Reliable, Secure Communication," Proceedings of the 1st ACM Conference on Computer and Communication Security, pp. 119–129 (Nov. 1993).

[17] W. van der, Aalst, A. Weijters, and L. Maruster, "Workflow Mining: Discovering Process Models from Event Logs." IEEE Transactions on Knowledge and Data Engineering, 16 (9), 1128-1142, 2004.

[18] W. van der, Aalst, H. Beer, and B. van Dongen, "Process Mining and Verification of Properties: An Approach based on Temporal Logic." In R. Meersman & Z. T. et al. (Eds.), On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE: OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2005 (Vol. 3760, pp. 130-147). Springer-Verlag, Berlin.

[19] W. M. P. van der Aalst, , A. K. A. de Medeiros, et al. (2005). "Genetic process mining." Applications and Theory of Petri Nets 2005: 48-69, 2005.

[20] B. F. van Dongen, A. K. A. de Medeiros, et al. "The ProM framework: A new era in process mining tool support." Applications and Theory of Petri Nets 2005: 444-454, 2005.

[21] C. Günther, and W. van der Aalst "A generic import framework for process event logs," Springer, 2006.

[22] A. Khalili, A. Jalaly Bidgoly, M. Abdolahi Azgomi, "PDETool: A multi-formalism modeling tool for discrete-event systems based on SDES description." Applications and Theory of Petri Nets: 343-352, 2005.

[23] J. M. Werf, B. F. Dongen, et al., "Process Discovery Using Integer Linear Programming," Proceedings of the 29th international conference on Applications and Theory of Petri Nets. Xi'an, China, Springer-Verlag: 368-387, 2008.