Covert Channel Detection Using Process Query Systems

Vincent Berk, Annarita Giani, George Cybenko
Thayer School of Engineering
Dartmouth College
Hanover,NH
$\{$*vberk, agiani, gvc*$\}$*@dartmouth.edu*

Abstract

In this paper we use traffic analysis to investigate a stealthy form of data exfiltration. We present an approach to detect covert channels based on a Process Query System (PQS), a new type of information retrieval technology in which queries are expressed as process descriptions.

1 Introduction

1.1 Covert Channels

A covert channel between two machines consists of sending and receiving data bypassing the usual intrusion detection techniques. In storage covert channels, data are written to a storage location by one process and then read by another process. The focus of this research is timing covert channels, in which the attacker modulates the time between the packets that are sent. Information is encoded in the inter-packet delays. Suppose an intruder gained access to machine X in our local network and uses this machine to exfiltrate information to machine A. At the receiver side, machine X receives packets with given delays. As the packets flow through the network they traverse a certain number of forwarding devices such as routers, switches, firewalls or repeaters. This equipment has an influence on the delays between packets so that the inter-packet delay at destination might not be exactly the same as at the source. In other words the section of network between sender and receiver acts as a noisy channel. Figure 1 gives a graphical description of the situation.

Suppose an apparatus on our network registers all the inter-packet delays of outgoing communications. Given the chain of delays that are seen, we want to state with a certain probability if a covert channel is being used. The goal is to analyze the inter-packet delays and see if they

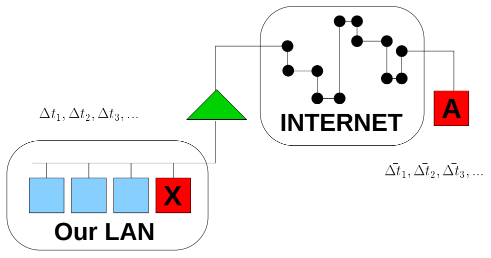

$\Delta t_1, \Delta t_2, \Delta t_3, \ldots$
INTERNET
A
X
Our LAN
$\bar{\Delta t}_1, \bar{\Delta t}_2, \bar{\Delta t}_3, \ldots$

Figure 1: An intruder was able to control machine X which is inside our local network and use it to exfiltrate data coded in inter-packet delays. Machine A is the receiver.

are too particular to be generated by a normal network communication.

The first formal definition of a covert channel was given in [7] as those used for information transmission which are neither designed nor intended to transfer information at all. Later covert channels were defined as those that use entities not normally viewed as data objects but can be manipulated maliciously to transfer information from one subject to another [5, 6]. Since covert communication is very difficult to detect, most researchers resort to investigating methods that simply minimize the amount of information that can be transmitted using a covert timing channel [4, 8].

1.2 Process Query Systems

Process Query Systems are a new paradigm in which user queries are expressed as process descriptions. This allows a PQS to solve large and complex information retrieval problems in dynamic, continually changing environments where sensor input is often unreliable. The system can take input from arbitrary sensors and then forms hypotheses regarding the observed environment, based on the process queries given by the user. Figure 2 shows a simple example of such a model. Model $\mathcal{M}_1$

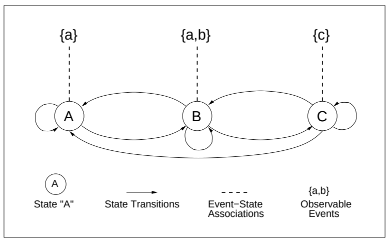


Figure 2: A Simple Process Model, $\mathcal{M}_1$

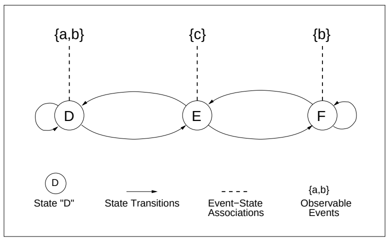


Figure 3: Another Process Model, $\mathcal{M}_2$

represents a state machine $S_1 = (Q_1, \Sigma_1, \delta_1)$, where the set of states $Q_1 = \{A, B, C\}$, the set of observable events $\Sigma_1 = \{a, b, c\}$, and the set of possible associations $\delta_1 : Q_1 \times \Sigma_1$ consists of

$$\delta_1 = \{\{A, a\}, \{B, a\}, \{B, b\}, \{C, c\}\}$$

A possible event sequence recognized by this model would be: $e_1 = a, e_2 = a, e_3 = b, e_4 = c, e_5 = b$ which we will write as $e_{1:5} = aabcb$ for convenience. Possible state sequences that match this sequence of observed events could be $AABCB$, or $ABBCB$, both of which are equally likely given $\mathcal{M}_1$. A rule-based model would need a lot of rules to identify this process, based on all the possible event sequences. Below is a set of all the rules necessary for detecting single transitions:

$$\begin{array}{ll} AA \rightarrow & \{aa\} \\ AB \rightarrow & \{aa\}, \{ab\} \\ BB \rightarrow & \{aa\}, \{ab\}, \{ba\}, \{bb\} \\ BA \rightarrow & \{aa\}, \{ba\} \\ BC \rightarrow & \{ac\}, \{bc\} \\ CC \rightarrow & \{cc\} \\ CB \rightarrow & \{ca\}, \{cb\} \\ CA \rightarrow & \{ca\} \end{array}$$

Let us introduce a second model $\mathcal{M}_2$ in Figure 3. Now consider the following sequence of events:

$$e_{1:24} = abaacabbacabacccabacabbc$$

where each observation may have been produced by instances of model $\mathcal{M}_1$, model $\mathcal{M}_2$, or be totally unrelated. A Process Query System uses multiple hypothesis, multiple model techniques to disambiguate observed events and associate them with a "best fit" description of which processes are occurring and in what state they are. In comparison, a rule-based system would get impossibly complex for the above situation.

A PQS is a very general and flexible core that can be applied to many different fields. The only things that change between different applications of a PQS are the format of the incoming observation stream(s) and the submitted model(s). Internally a PQS has four major components that are linked in the following order:

1. Incoming observations.
2. Multiple hypothesis generation.
3. Hypothesis evaluation by the models.
4. Selection, pruning, and publication.

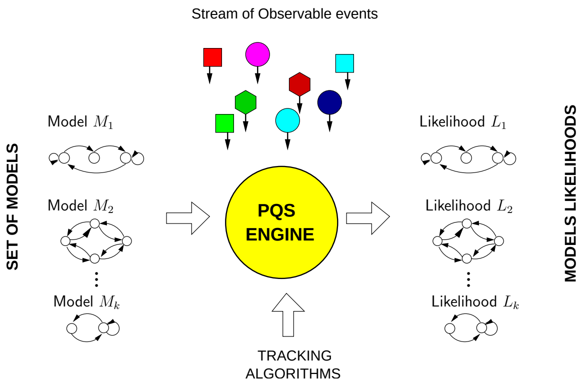


Figure 4: A set of models is given as input to the engine. Given a stream of observations and using tracking algorithms the engine returns the likelihood of each model.

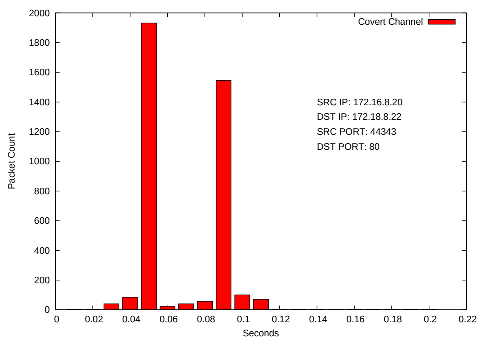
The big benefits of a PQS are its superior scalability and applicability. The application programmer simply connects the input event streams and then focuses on writing process models. Models can be constructed as state machines, formal language descriptions, Hidden Markov Models, kinematic descriptions, or a set of rules. The PQS is now ready to track processes occurring in a dynamic environment and continuously present *the best possible explanation of the observed events* to the user.

See Figure 4 for a graphical representation of the system. A detailed overview and application of a Process Query System can be found in [2, 1, 3].

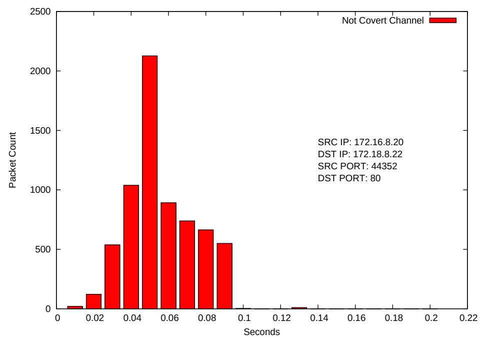
2 Detection

2.1 Covert Channel Models

We focus our attention on binary codes. Our models are based on the assumption that in the case of a covert channel, the inter-packet delays will center around two distinct values (ie. two distinct delays), see Figure 5 (a).



(a)



(b)

Figure 5: The figures show the number of packets received with a given delay. Horizontal axis shows the inter-arrival time in seconds, vertical axis shows the number of packet arrived. The parameters are the following. In case (a) the two spikes show that a covert channel communication is in place. Case (b) represents normal communication.

For a covert channel the sample mean μ (average) of the inter-packet delays will be somewhere between the two spikes. The packet-count in the histogram at that point will therefore be very low. However, looking at a normal traffic pattern the mean of the inter-packet delays will be in the center of the large spike. The packet-count at the mean will thus be very high, if not the highest. If we divide the packet-count at the mean by the maximum packet-count from the histogram, we get a measure of how likely it is that the communication is a covert channel. In particular the smaller is the ratio $\frac{C_\mu}{C_{max}}$ the higher is the probability of having a covert channel communication. We can therefore assume the following probability:

$$P_{CovChan} = 1 - \frac{C_\mu}{C_{max}}$$

where $C(\mu)$ is the packet-count at the mean and C_{max} is the maximum packet-count of the histogram. Experiments with three different types of data were conducted, and Figure 6 shows the ratio $\frac{C_\mu}{C_{max}}$ for these experiments. The data considered are of the following types.

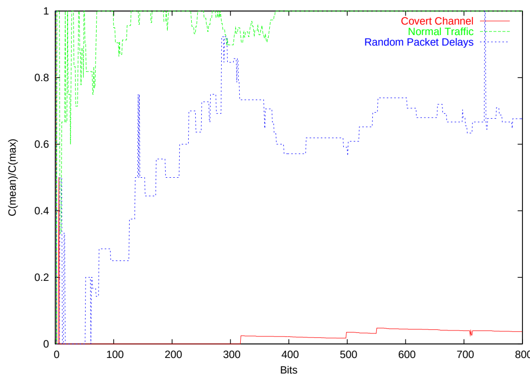


Figure 6: The ratio between the packet count at the sample mean and the maximum packet count for normal traffic, fully random delays, and for a binary covert channel. Horizontal axis shows the length of the estimated sequence in bits, vertical axis shows $\frac{C_\mu}{C_{max}}$.

Normal Data. Packets with a average delay of 0.2 seconds were transmitted. The inter-arrival times vary, but the maximum number of packets has a delay of 0.2 seconds. The sample mean μ is therefore represented by a delay very close to 0.2 and the number of packets with exactly that delay (C_μ) is very high. The ratio between this number and the maximum number of a packet for a certain delay (C_{max}) quickly grows to 1.0, and stays there as more packets are transmitted.

Random Data. Packets are sent with a fully random delay. Although this is not realistic for traffic encountered on the network, it does present a good idea of the worst case scenario. Initially, when only a few bits have been sent, the delays scatter across the range and it is unlikely that the sample mean will have a high count. That explains why until approximately the first 10 bytes the ratio $\frac{C_\mu}{C_{max}}$ remains zero. Later on, as more packets arrive the histogram starts to flatten so the ratio starts to rise. As the number of transmitted packets increases even further the ratio keeps growing, until it eventually hits 1.0 as the packet count goes to infinity.

Covert Channel Communication. Two delays are used, thus the interarrival times concentrate around those two values. The sample mean μ lies approximately in the middle between the two spikes. The count C_μ is low and therefore the ratio $\frac{C_\mu}{C_{max}}$ is approximately zero. As more and more bits are transmitted over the covert channel the spikes increase in size, however that ratio always remains very close to zero.

Our algorithm detects the sequence that most likely represents the covert communication channel analyzing the value $\frac{C_\mu}{C_{max}}$. The lower that value the higher is the probability of having a malicious communication hidden in inter-packet delays.

Figure 6 compares the three cases.

2.2 Implementation of a PQS

Our current implementation of a Process Query System is called TRAFEN (TRacking And Fusion ENgine). We implemented software applications to use as sensors that send data to the engine. TRAFEN then, given such observations, returns the most likely hypotheses.

In order to input observations in the system we built sensors. This software monitors the packet flows and returns quantities that are crucial to our models. It aggregates packets that have the same source IP, destination IP, source Port, destination Port and computes the quantity $\frac{C_\mu}{C_{max}}$. An observation looks like the following:

Src ip, Src port,
Dst ip, Dst port,
Protocol, $\frac{C_\mu}{C_{max}}$.

The TRAFEN engine evaluates each incoming observation and checks their correlation. Correlated events are assigned to the same track. Sets of tracks constitute a hypothesis. At the moment our process descriptions are mainly based on first principles. An example of a model used in our experiments is the following.

Singleton: Get observation

$$\frac{C_\mu}{c_{max}} \geq 0.9 \Rightarrow Score = 0.001$$

$$\frac{C_\mu}{c_{max}} < 0.9 \Rightarrow Score = 0$$

Track: Get observation OBS. If OBS is consistent (same characteristics with the track)

$$f = 1 - \frac{1}{lenght\ track + 1}$$

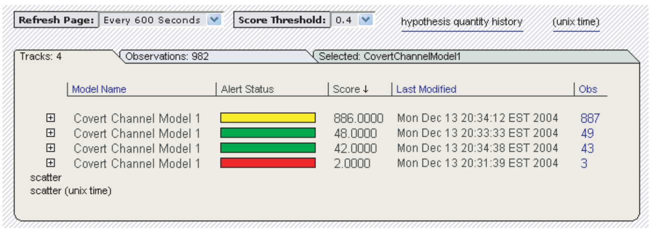
$$Track\ Score = f \cdot \left(1 - \frac{C_{\mu\ Track}}{C_{max\ Track}}\right)$$

If OBS is NOT consistent with the track $Track\ Score = 0$.

The first case considers only tracks with one observation. The score is dependent only on the ratio. In the second case tracks with more than one observation are considered. In this case we assume that the longer the track is, the higher the score will be.

3 Results

A covert channel communication was simulated and many models were built and inserted into TRAFEN. We found that some models performed better than others. All of them were able to detect covert channel communication but some of them returned also false positive. We built a front-end displaying the most likely hypothesis, the different tracks together with their score, see Figure 7.



Refresh Page: Every 600 Seconds
Score Threshold: 0.4
hypothesis quantity history (unix time)

Tracks: 4 | Observations: 982 | Selected: CovertChannelModel1

Model Name	Alert Status	Score ↓	Last Modified	Obs
⊞ Covert Channel Model 1	🟨	886.0000	Mon Dec 13 20:34:12 EST 2004	887
⊞ Covert Channel Model 1	🟩	48.0000	Mon Dec 13 20:33:33 EST 2004	49
⊞ Covert Channel Model 1	🟩	42.0000	Mon Dec 13 20:34:38 EST 2004	43
⊞ Covert Channel Model 1	🟥	2.0000	Mon Dec 13 20:31:39 EST 2004	3

scatter
scatter (unix time)

Figure 7: www.pqsnet.net/display.

4 Conclusion and Future Work

Process query system is a crucial tool in the analysis and detection of network threats. We applied it to solve the problem of detecting covert channel and we found very promising results. We plan to investigate different types of exfiltration of information and develop detection techniques.

5 Acknowledgments

Supported under ARDA P2INGS Award No. F30602-03-C-0248. Points of view in this document are those of the author(s) and do not necessarily represent the official position of ARDA.

References

[1] V. Berk and N. Fox. Process Query Systems for Network Security Monitoring. *Proceedings of the SPIE*, March, 2005.

[2] G. Cybenko, V. Berk, V. Crespi, R. Gray, and G. Jiang. An Overview of Process Query systems. *Proceedings of the SPIE*, April, 2003.

[3] G. Cybenko, V. Berk, and C. Roblee. Large-Scale Autonomic Server Monitoring Using Process Query Systems. *Proceedings of the SPIE*, March, 2005.

[4] J. Giles and B. Hajek. An Information-Theoretic and Game-theoretic Study of Timing Channels. *IEEE Trans. on Information Theory*, 48(9):2455–2477, Sept 2002.

[5] R. A. Kemmerer. Shared Resource Matrix Methodology: An Approach to Identifying Storage and Timing Channel. *ACM Transaction on Computer Systems*, 1(3):256–277, Aug 1983.

[6] R. A. Kemmerer. Shared Resource Matrix Methodology: An Approach to Identifying Storage and Timing Channel : Twenty years later. *Proc. 18th Annual Computer Security Applications Conference (ACSAC)*, pages 109–118, 2002.

[7] B. W. Lampson. A Note on the Confinment Problem. *Proc. of the Communication of the ACM*, 16(10):613–615, Oct 1973.

[8] N. Ogurtsov, H. Orman, R. Schroeppel, S. O'Malley, and O. Spatscheck. Covert Channel Elimination Protocols. *Technical Reports TR96-14. Department of Computer Science, University of Arizona*, 1996.