Covert Channels in the IP Time To Live Field

Sebastian Zander, Grenville Armitage, Philip Branch
Centre for Advanced Internet Architectures (CAIA)
Swinburne University of Technology
Melbourne Australia
{szander, garmitage, pbranch}@swin.edu.au

***Abstract*— Covert channels are used for the secret transfer of information. Unlike encryption, which only protects the information from unauthorised observers, covert channels aim to hide the very existence of the communication. The huge amount of data and vast number of different network protocols in the Internet makes it an ideal high-capacity vehicle for covert communication. Covert channels pose a serious security threat as they can be used for a number of malicious activities. In this paper we present a novel covert channel inside the IP header's Time To Live (TTL) field. The sender manipulates the TTLs of subsequent packets transmitting covert information to the receiver. Since network elements along the path also modify the TTL, the capacity and stealth of this channel depend on the "natural" TTL variation. We analyse this variation in multiple traffic traces and propose an encoding scheme, which makes the TTL covert channel look similar to "natural" variation. We also discuss methods to eliminate and detect this covert channel.**

***Index Terms*— Security, Covert Channels, Network Protocols**

I. Introduction

Often it is thought that using encryption is sufficient to secure communication. However, the simple fact that there is communication is often enough to raise suspicion and take further actions. While encryption is useful to protect information from unauthorised observers, using covert channels aim to hide the very existence of the communication. Covert channels often use means of communication not normally intended to be used, making the job of identifying them quite elusive.

Covert channels were introduced by Lampson in the context of single hosts as a mechanism, by which a high security process leaks information to a low security process that would otherwise not have access to it [1]. While a serious threat even for monolithic systems, the potential of covert channels in computer networks is greatly increased. The huge amount of data and vast number of different protocols in the Internet makes it ideal as a high-capacity vehicle for covert communications. In computer networks overt channels, such as network protocols, are exploited as carriers for covert information [2], [3]. Even if only one bit per packet can be covertly transmitted, a large Internet site could lose 26GB of data annually [4]. Furthermore, the covert channel capacity in computer networks will further increase in the future due to new high-speed network technologies.

Covert channels can be used for a number of purposes, some posing serious security threats:

- Criminals, terrorists and intelligence agencies can use covert channels for communication. Using only encryption would not prevent adversaries to detect communication patterns, which is often sufficient to detect the onset of criminal activity, discover organisational structures or justify obtaining police warrants.
- Hackers that have compromised computer systems usually either ex-filtrate data or use the systems for malicious purposes (such as launching Denial of Service attacks). The generated network traffic – if not covert – would alert system administrators, which then would discover the compromised systems.
- Recent attempts by some governments to limit the freedom of speech and civilian use of (strong) cryptography have lead to proposals for utilising covert channels to circumvent these measures.

Covert channels in computer network protocols are similar to techniques for hiding information in audio, visual or textual content (steganography). While steganography requires some form of content as cover, covert channels require some network protocol as carrier. The ubiquitous presence of a small number of network protocols suitable as carriers (e.g. the Internet Protocol [5]) makes covert channels widely available and usable even in situations where steganography cannot be applied.

We propose a novel covert channel in the IP header's Time To Live (TTL) field [5]. Although this header field was never intended to be used for communication, we demonstrate that a covert sender can encode information in the TTL fields of subsequent packets, which can be later decoded by a covert receiver. However, since the TTL field of packets in flight is modified by network devices (e.g. routers, firewalls) and packets can traverse different paths from sender to receiver, there is "natural" TTL variation.

The implications of this are twofold. Firstly, the encoding of the covert channel should be chosen so that the covert channel looks similar to "natural" TTL variation. Secondly, the capacity of the covert channel depends on the "natural" TTL variation (channel noise). We analyse TTL variation in multiple traffic traces captured in the Internet and propose an encoding scheme, which make the TTL covert channel look similar to "natural" TTL variation.

The paper is organised as follows. In section II we describe existing covert channels inside protocol header fields. In section III we propose a covert channel in the IP TTL field that looks similar to "natural" TTL variation and also discuss methods to eliminate and detect it. In section IV we analyse

"natural" TTL variation from multiple traffic traces captured in the Internet. Section V concludes and outlines future research.

II. Related Work

Unused or reserved bits in packet headers, packet padding, or various header fields can be used as covert channels. Handel et al. proposed using the unused bits of the IP [5] Type of Service (TOS) field , or of the TCP [6] flags field [7]. Ahsan et al. introduced a technique where the IP Don't Fragment (DF) bit is used [8]. The DF bit can be set to arbitrary values if the sender knows the Maximum Transfer Unit (MTU) of the path to the receiver and only sends packets of less than MTU size. A number of researchers proposed to hide covert data in padding, if the standard does not enforce specific values [4], [7], [9]. For example, Ethernet pads frames to a minimum length of 60 bytes, and the IP and TCP header are padded to 4-byte boundaries if header options are present. Fisk et al. proposed transmitting covert data in TCP reset (RST) segments [4]. Segment with RST flag set abort the connection and usually contain no data.

Popular header fields for inserting covert information are the IP identification field and the TCP initial sequence number. The IP Identification (ID) field is used to uniquely identify IP packets for a certain time period, necessary to reassemble fragmented packets [5]. Rowland proposed to multiply each byte of covert information with 256 and use it as IP ID [10]. Ahsan et al. proposed to transmit covert information in the high byte of the IP ID (as XOR of the data and a secret key) and generate the low byte randomly [8].

TCP Sequence numbers are used to coordinate which data has been transmitted and received to guarantee reliable transport. The first sequence number selected by a client is called the Initial Sequence Number (ISN). The ISN must be selected carefully, so that sequence numbers of new incarnations do not overlap with sequence numbers of earlier incarnations of a connection [6]. Rowland proposed to multiply each covert byte with 256^3 and use it as ISN [10]. Murdoch et al. pointed out that it is easy to detect covert channels if the resulting value distribution differs from the normal distribution generated by real operating systems [11]. They developed ISN covert channels tailored for Linux/OpenBSD, where the covert channel distribution looks exactly like the normal distribution.

Giffin et al. developed a method for covert messaging through TCP timestamp options, which are used by many operating systems to improve TCP performance [12]. The sender inserts covert information in the timestamps low order bits, because these bits are effectively random for slow TCP connections. The algorithm slows the TCP stream so that the timestamps on packets are valid when they are sent.

Jones et al. proposed a covert channel in the IP TTL field to trace back IP packets without using the source address field (needed to effectively react to Denial of Service (DoS) attacks). [13]. While Jones et al. channel is only used for packet marking, Qu et al. have proposed a covert channel in the IP TTL field usable for general communication [14].

However, the channel encoding proposed by Qu et al. does not consider normal initial TTL values and "natural" TTL variation occurring in the network, and is therefore easy to detect. In this paper we we analyse "natural" TTL variation and initial TTL values, and propose a TTL covert channel that looks similar to a normal flow with "natural" TTL variation.

III. TTL Covert Channel

In this section we first describe the covert channel communication model, and then how the IP TTL field is used as covert channel. Finally, we discuss methods to eliminate and detect the TTL covert channel.

A. Communication Model

The de-facto standard covert channel communication model is the prisoner problem introduced by Simmons [15]. Two people, Alice and Bob[1], are thrown into prison and intend to escape. To agree on an escape plan they need to communicate, but all their messages are monitored by Wendy the warden. If Wendy finds any signs of suspicious messages Wendy will place Alice and Bob into solitary confinement – making an escape impossible. Alice and Bob must exchange innocuous messages containing hidden information that, they hope, Wendy will not notice. There are different types of wardens [16]:

- A *passive* warden can only spy on the channel but cannot alter any messages.
- An *active* warden is able to slightly modify the messages, but without altering the semantic context.
- A *malicious* warden may alter the messages without impunity, but in reality malicious wardens are rare [16].

Extending this scenario towards communication networks, Alice and Bob are using two networked computers to communicate. They run some innocuous looking overt communication between their computers, with a hidden covert channel. Alice and Bob share a secret useful for determining covert channel encoding parameters and for encrypting/authenticating the hidden messages. For practical purposes Alice and Bob may well be the same person, for example a hacker ex-filtrating restricted information. Wendy manages the network and can monitor the passing traffic for covert channels or alter the passing traffic to eliminate or disrupt covert channels. Figure 1 depicts the communication model (Alice sending to Bob).

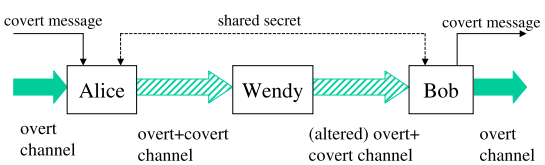

Fig. 1. The prisoner problem – model for covert channel communication

In computer networks Alice and Bob do not have to be the sender and receiver of the overt communication. One or

[1]Cryptographic protocols are usually illustrated using participants named alphabetically (Alice, Bob) or with names where the first letter matches their role (Wendy the warden).

both of them may act as a middleman. If Alice can observe and manipulate an existing overt communication from an innocent sender that reaches Bob, she can insert a covert channel into it. Bob does not need to be the receiver of the overt communication, but merely must be able of observing it to decode the hidden information. If Bob can also alter the overt communication, he can even remove the covert channel preventing the receiver of the overt communication from discovering it. A middleman could be located for example inside a network router or inside an end host's network stack.

B. Channel Encoding

The IP TTL header field limits the lifetime of an IP packet, preventing packets from living forever during routing loops [5]. A packet's TTL is set by the sender and decremented by each network element along the path processing the packet's IP header (e.g. routers and firewalls). Packets are discarded if their TTL becomes zero while still in transit.

The covert sender cannot directly encode covert information in the TTL field because using any other values apart from the few normal initial TTLs [17], [18] would be highly suspicious, and as a middleman it cannot drastically alter the TTL without the risk of packets indefinitely looping (TTL increase) or not reaching their destination (TTL decrease). Encoding covert information directly as TTL values would also require that the covert receiver always knows how many hops the sender is away. Since the number of hops can change at any time due to routing changes this is impractical.

We propose to encode the covert information using two symbols: *low-TTL* signals a 0-bit whereas *high-TTL* signals a 1-bit. Low- and high-TTL are two particular TTL values chosen by the covert sender based on the overt flow. The covert sender uses either the default initial TTL (if also the overt sender) or the (lowest) TTL of the intercepted packets (if a middleman) as high-TTL, and high-TTL minus one as low-TTL. Decrementing the default TTL eliminates the risk of packets stuck in routing loops, while it is still very likely that packets reach their final destination. (Common operating systems use initial TTL values of at least 64 and the number of hops between two hosts of the Internet is generally assumed to be smaller than 32 [17].)

The receiver observes a flow of packets with two different TTL values, and will decode packets with the higher TTL as 1-bit and packets with the lower TTL as 0-bit. Before the receiver can start the decoding it needs to see both TTL values to identify low-TTL and high-TTL. Therefore, the sender should not send long sequences of zero or one bits at the start.

C. Channel Errors and Capacity

We found that "natural" TTL variation in packet flows between senders and receivers is not very common, but present in a significant amount of flows (see section IV). This means that often a covert sender and receiver could expect a channel without noise caused by "natural" TTL variation. The current channel encoding only copes with slow "natural" TTL variation, for example infrequent path changes. After a path change the receiver will decode few bits incorrectly, but assuming a mix of zero and one bits is sent, it will quickly discover a TTL lower than low-TTL or higher than high-TTL and can adapt its decoding accordingly. Another issue is packet loss and reordering. TCP flows provide reliable in-order delivery of packets, but UDP flows can experience packet loss and reordering. To ensure a reliable transport, well-known techniques such as error correcting codes or, segmentation and retransmission could be used.

Without errors the capacity of the TTL covert channel is one bit per packet of the overt channel. With packet rates of up to hundreds of packets/second (see section IV-E) the bit rate of the covert channel is up to hundreds of bits/second for a single overt flow. Multiple overt flows could be utilised in parallel to increase the capacity. A more comprehensive study of channel errors, capacity and the development of a more robust channel encoding is left for future work.

D. Elimination

All channels outlined in section II can be eliminated by enforcing standard-conform header fields and padding, or by rewriting header fields such as the IP ID, TCP ISN, and timestamps. In practice however, limited processing capabilities of the active warden may prevent complete elimination.

To eliminate the TTL covert channel the warden needs to set the TTLs of all packets of a flow to the same value. The warden should set the TTLs to a value equal to the smallest TTL of the flow (thereby reducing larger TTLs) for the reasons explained in section III-B. A warden that is not close to the sender and therefore unable to manipulate all packets cannot completely eliminate the channel, but would possibly reduce the capacity because of the added "noise".

E. Detection

All channels described in section II, except the undetectable channel in [11], are fairly simple to detect, because they use non-standard or invalid header field values or combinations, or the resulting value distributions look abnormal. The TTL channel is not undetectable, but is potentially harder to detect than most previous channels. If the covert sender is also the overt sender and the warden is only one hop away, any observed TTL variation would indicate the presence of a covert channel. However, if the covert sender is a middleman and/or the warden is multiple hops away from the sender detection is not that trivial.

The warden needs to differentiate between normal TTL variation (see section IV) and the covert channel. We have chosen the encoding of the channel so that the number of TTLs (see section IV-C) and the TTL amplitude (see section IV-D) appear normal. A high frequency of TTL changes is suspicious, but the covert sender can sacrifice capacity to improve stealth by reducing the change frequency (see section IV-E). However, a warden could probably still detect the TTL covert channel by a more in-depth analysis of the actual change pattern and comparison with normal change patterns.

IV. "NATURAL" TTL VARIATION IN THE INTERNET

In this section we investigate the "natural" TTL variation occurring in Internet packet flows. Existing work focused on TTL to study hop counts between Internet hosts on longer time scales [19], [20]. In contrast, we study TTL changes within short time scales of single traffic flows from one particular viewpoint (capture device or covert receiver).

A. Datasets

We use three different packet traces as input: a trace captured on a public game server at our lab in Melbourne (CAIA), a trace captured on a public game server connected to the Grangenet research network [21] in Canberra (Grangenet), and a public trace captured at a 1Gbit/s aggregated uplink of a student dorm Asymmetric Digital Subscriber Line (ADSL) access network (Twente) [22].

The CAIA and Grangenet traces were continuously captured from May 2005 to June 2006, and the Twente trace was captured in July 2004 (we use only the first week). The CAIA trace contains only game traffic, the Grangenet trace contains game and web traffic, and the Twente trace contains a mix of traffic (including web, peer-to-peer, game, email traffic).

We grouped packets into flows according to IP addresses, port numbers and protocol. The end of a flow is determined by the TCP connection teardown or a 600 second idle timeout (whichever occurs first). Although we treat flows as bidirectional for the TCP state tracking and flow timeout, we later separate the two directions into unidirectional flows because TTL variation can differ in either direction. In the following the notion of flow is always unidirectional. For the CAIA and Grangenet traces we have removed all flows originating from the servers, because their TTL is constant (initial TTL). We only consider flows with at least four packets and an average packet rate of at least one packet/second (pps) because for other flows the potential covert channel capacity is very low.

B. Number of Flows with TTL Changes

Table I shows the number of flows and bytes with/without TTL changes, and the percentages of flows and bytes with TTL changes. It also shows percentiles of the distribution of estimated hop counts between sender and receiver. A small but significant number of flows has packets with different TTL values. This seems beneficial for the use of TTL as covert channel: noise caused by "natural" TTL variation would be uncommon, but flows with varying TTLs are not uncommon enough to immediately suspect a covert channel. The CAIA and Grangenet traces exhibit slightly larger TTL variation, which is probably caused by larger average hop counts. In the Twente trace ~50% of the flows have very small hop counts because they originate from the ADSL clients.

C. Number of Different TTLs per Flow

Figure 2 shows the cumulative distributions of the number of different TTL values in flows with varying TTL. In 92% of CAIA and 99% of Grangenet and Twente flows, the TTL changes only between two distinct values (please note the y-axis limits). Therefore, a covert channel using more than two distinct TTL values would be suspicious.

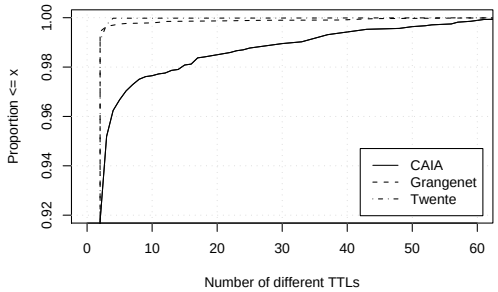

Fig. 2. Cumulative distributions of number of different TTL values per flow

D. Amplitude of TTL Changes

Figure 3 shows the cumulative distributions of the amplitude of the TTL changes. We define the amplitude as the difference between the minimum and maximum TTL values of a flow. For 90% of the CAIA trace flows the amplitude is one. For the Grangenet and Twente traces a large number of flows have large amplitudes presumingly caused by firewalled TCP flows.

Modern firewalls send packets part of the TCP handshake or teardown on behalf of clients (e.g. for SYN flood attack protection). The TTLs in these packets are set to the initial TTL of the firewall, which can differ from the initial TTL used by the host behind the firewall. Different operating systems use different initial TTLs, the most common being: 64 (Linux, FreeBSD), 128 (Windows 98/XP), 255 (Cisco) [17], [18]. Therefore, the difference of the two TTLs will likely be 64, 127, or 191 plus/minus the distance between firewall and host.

We also plot the CDFs for Grangenet and Twente flows untouched by firewalls (which look similar to that of the CAIA trace). This means a covert channel using amplitudes other than one would be suspicious.

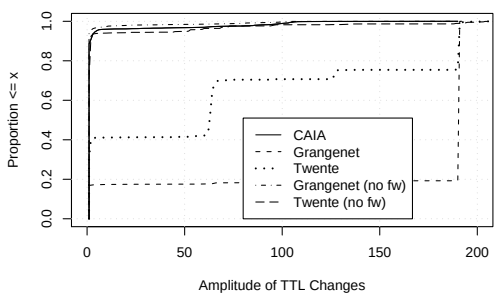

Fig. 3. Cumulative distributions of the amplitude of TTL changes per flow

E. Frequency of TTL Changes

Figure 4 shows the cumulative distributions of the average frequency of TTL changes (excluding firewalled TCP flows).

TABLE I

FLOWS AND BYTES WITH AND WITHOUT TTL CHANGES AND ESTIMATED HOP COUNT PERCENTILES

Dataset	Flows w/o TTL change	Flows with TTL change	Volume w/o TTL change	Volume with TTL change	Estimated hop count percentiles (1%, 25% 50%, 75%, 99%)
CAIA	128,617	2766 (2.1%)	114.5 GB	6.0 GB (5.0%)	7, 11, 14, 19, 26
Grangenet	282,898	8582 (2.9%)	28.1 GB	0.9 GB (3.1%)	5, 12, 15, 18, 26
Twente	1,354,585	24,603 (1.8%)	62.0 GB	1.8 GB (2.8%)	3, 3, 4, 11, 19

We define the average frequency of TTL changes as the number of TTL changes divided by the total number of consecutive packet pairs. On average most flows only change TTL values every 2-3 packet pairs or slower. A TTL covert channel cannot change TTL values faster without being suspicious.

Multiplying the per-packet-pair change rates with the average packet rates of the flows, we find on average most flows have at most 10-15 TTL value changes per second. The Twente trace has higher frequencies, because its flows have packet rates up to few 100pps, whereas in the CAIA and Grangenet traces packet rates are only 20-30pps (characteristic for game flows). This results in maximum channel capacities of up to 10-15bit/s for single overt flows.

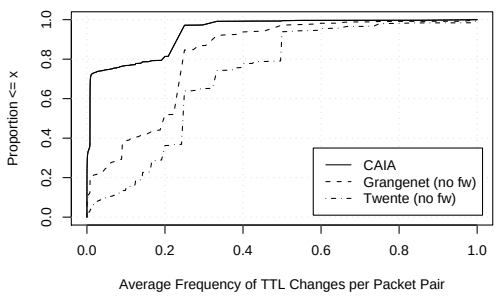

Fig. 4. Cumulative distributions of the frequency of TTL changes per flow

V. CONCLUSIONS AND FUTURE WORK

In this paper we presented a novel covert channel inside the IP header's Time To Live (TTL) field. We analysed the short-term "natural" variation of TTLs in traffic flows based on real traffic traces captured in the Internet. We found that a small but significant number of flows experience TTL changes during their lifetime, and we showed the main characteristics of these TTL changes. The covert channel encoding was chosen so that the covert channel looks similar to "natural" TTL variation with regard to the analysed characteristics. Therefore, the TTL covert channel is potentially harder to detect than many covert channels proposed previously. The maximum capacity of the channel is likely up to several 100bit/s (depending on the overt channel used as carrier). We also have outlined suitable techniques to eliminate and detect the TTL covert channel.

Future work will extend the analysis of "natural" TTL variation towards a larger set of traffic traces and a more in-depth study of the change patterns. We will also evaluate the capacity of the TTL covert channel in the presence of "natural" TTL variation and packet loss. Furthermore, we will investigate how difficult it is to detect the channel with anomaly detection methods. The ultimate goal is to develop an improved channel encoding that is robust against "natural" TTL variation and packet loss, while providing good stealth.

REFERENCES

[1] B. Lampson, "A note on the confinement problem," *Communication of the ACM*, vol. 16, pp. 613–615, October 1973.

[2] M. A. Padlipsky, D. W. Snow, P. A. Karger, "Limitations of End-to-End Encryption in Secure Computer Networks," Tech. Rep. ESD-TR-78-158, Mitre Corporation, August 1978.

[3] C. G. Girling, "Covert Channels in LAN's," *IEEE Transactions on Software Engineering*, vol. SE-13, pp. 292–296, February 1987.

[4] G. Fisk, M. Fisk, C. Papadopoulos, J. Neil, "Eliminating Steganography in Internet Traffic with Active Wardens," in *Proceedings 5th International Workshop on Information Hiding*, October 2002.

[5] J. Postel, "Internet Protocol," RFC 0791, IETF, Sept. 1981. `http://www.ietf.org/rfc/rfc0791.txt`.

[6] J. Postel, "Transmission Control Protocol," RFC 0793, IETF, Sept. 1981. `http://www.ietf.org/rfc/rfc0793.txt`.

[7] T. Handel, M. Sandford, "Hiding data in the OSI network model," in *Proceedings of the First International Workshop on Information Hiding*, pp. 23–38, 1996.

[8] K. Ahsan, D. Kundur, "Practical data hiding in TCP/IP," in *Proceedings ACM Workshop on Multimedia Security*, December 2002.

[9] M. Wolf, "Covert Channels in LAN Protocols," in *Proceedings of the Workshop on Local Area Network Security (LANSEC)*, pp. 91–101, 1989.

[10] C. H. Rowland, "Covert channels in the TCP/IP protocol suite," *First Monday, Peer Reviewed Journal on the Internet*.

[11] S. J. Murdoch, S. Lewis, "Embedding Covert Channels into TCP/IP," in *Proceedings of 7th Information Hiding Workshop*, June 2005.

[12] J. Giffin, R. Greenstadt, P. Litwack, R. Tibbetts, "Covert messaging through TCP timestamps," in *Proceedings of the Privacy Enhancing Technologies Workshop (PET)*, pp. 194–208, April 2002.

[13] E. Jones, O. Le Moigne, J.-M. Robert, "IP Traceback Solutions Based on Time to Live Covert Channel," in *Proceedings of 12th IEEE International Conference on Networks (ICON)*, pp. 451–457, November 2004.

[14] H. Qu, P. Su, D. Feng, "A typical noisy covert channel in the IP protocol," in *38th Annual International Carnahan Conference on Security Technology*, pp. 189–192, October 2004.

[15] G. J. Simmons, "The Prisoners' Problem and the Subliminal Channel," in *Proceedings of Advances in Cryptology (CRYPTO)*, pp. 51–67, 1983.

[16] S. Craver, "On Public-Key Steganography in the Presence of an Active Warden," in *Proceedings of the Second International Workshop on Information Hiding*, pp. 355–368, April 1998.

[17] C. Jin, H. Wang, K. Shin, "Hop-Count Filtering: An Effective Defense Against Spoofed DoS Traffic," in *Proceedings of the 10th ACM International Conference on Computer and Communications Security (CCS)*, pp. 30–41, October 2003.

[18] N. Davids, "Initial TTL Values." `http://members.cox.net/~ndav1/self_published/TTL_values.html`.

[19] A. Fei, G. Pei, R. Liu, L. Zhang, "Measurements on Delay and Hop-Count of the Internet," in *Proceedings of IEEE GLOBECOM - Internet Mini-Conference*, 1998.

[20] F. Begtasevic, P. van Mieghen, "Measurements of the Hop Count in the Internet," in *Proceedings of Workshop on Passive and Active Measurement (PAM)*, pp. 183–190, April 2001.

[21] Grid and Next Generation Network (GrangeNet). `http://www.grangenet.net/`.

[22] R. van de Meent, "M2C Measurement Data Repository," December 2003. `http://m2c-a.cs.utwente.nl/repository/`.