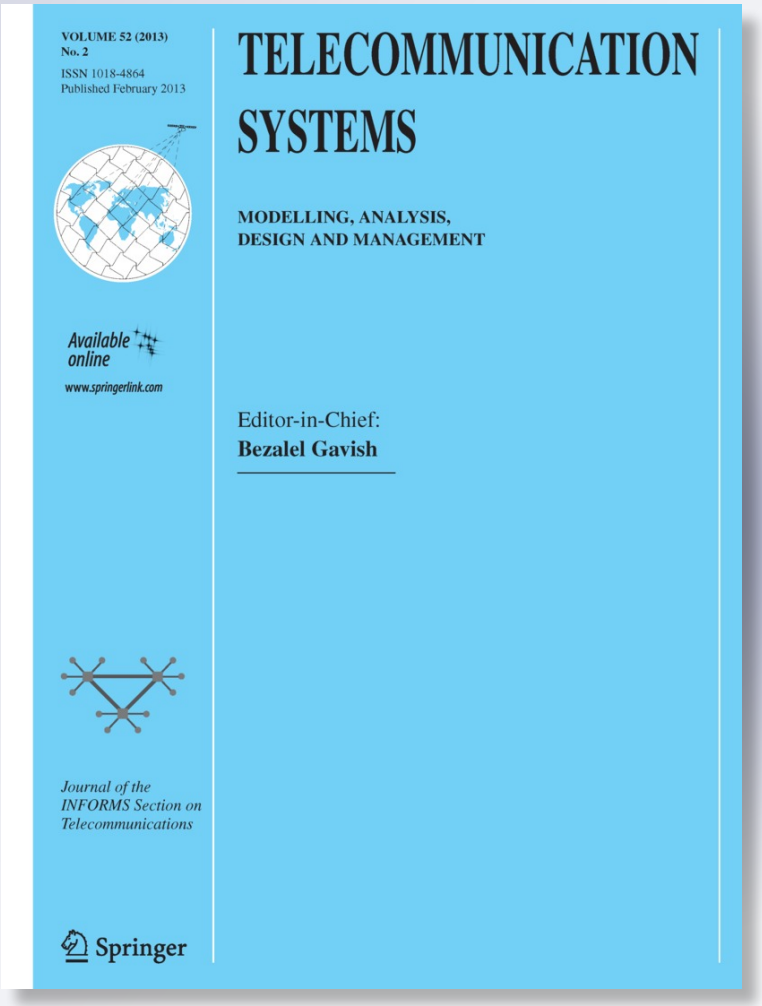
On information hiding in retransmissions

Wojciech Mazurczyk, Miłosz Smolarczyk & Krzysztof Szczypiorski

Telecommunication Systems
Modelling, Analysis, Design and Management

ISSN 1018-4864
Volume 52
Number 2

Telecommun Syst (2013) 52:1113-1121
DOI 10.1007/s11235-011-9617-y



Springer

Telecommun Syst (2013) 52:1113–1121
DOI 10.1007/s11235-011-9617-y

On information hiding in retransmissions

Wojciech Mazurczyk · Miłosz Smolarczyk · Krzysztof Szczypiorski

Published online: 1 September 2011

Abstract The paper presents an idea and experimental results for RSTEG (Retransmission Steganography), which is an intra-protocol hybrid network steganography method. It is intended for a broad class of protocols that utilises retransmission mechanisms. RSTEG enables hidden communication by not acknowledging a successfully received packet in order to intentionally invoke retransmission. The retransmitted packet carries a steganogram instead of user data in the payload field. Experimental results for TCP-based RSTEG traffic analysis are enclosed which were focused on measuring steganographic bandwidth and influence on TCP network traffic in terms of undetectability.

Keywords RSTEG · Network steganography · Retransmission mechanism

1 Introduction

Network steganography is a new trend in steganography which utilizes hiding secret data in the normal data transmissions of users so that it ideally cannot be detected by third parties. Network steganography methods may be viewed as a threat to network security, as they may be used, among others, as a tool data exfiltration. For this reason, it is important to identify possibilities for covert communication, as knowledge of information hiding procedures may be used to develop countermeasures.

Network steganography methods may be classified as intra- or inter-protocol steganography [1]. If hidden communication is performed using single network protocol as a secret data (steganogram) carrier then it is called intra-protocol steganography. If two or more protocols are utilized it is called inter-protocol steganography (the example of the inter-protocol steganographic is PadSteg [1]). Intra-protocol methods may be classified into following subgroups:

- **Steganographic methods that modify protocol PDUs**, including network protocol headers, payload fields or both. Example methods: HICCUPS (Hidden Communication System for Corrupted Networks [2]), watermarking algorithms, speech codec steganographic algorithms, methods based on the modification of IP, TCP and UDP header fields.
- **Steganographic methods that modify PDUs time relations**, for example, by affecting the order of packets, modifying inter-packet delay or introducing intentional losses [3]. These methods are harder to detect, but gives lower steganographic capacity than methods that utilise protocol-specific fields. Using methods from this group may affect transmission quality.
- **Hybrid steganographic methods** that modify both the content of packets and their timing and ordering. Methods from this group are hard to detect and gives high steganographic bandwidth, but may affect transmission quality. Example of hybrid method is RSTEG (Retransmission Steganography), which was originally introduced in [4] and is presented in detail in this paper.

In the context of the above classification of network steganography methods, we proposed in [4] a new hybrid method called RSTEG, which is intended for a broad class of protocols that utilise retransmission mechanisms. The main innovation of RSTEG is to not acknowledge a successfully and orderly received packet to intentionally invoke retransmission. The retransmitted packet of user data then carries a steganogram in the payload field.

W. Mazurczyk (✉) · M. Smolarczyk · K. Szczypiorski
Institute of Telecommunications, Warsaw University of Technology, ul. Nowowiejska 15/19, Warsaw 00-665, Poland
e-mail: wmazurczyk@tele.pw.edu.pl

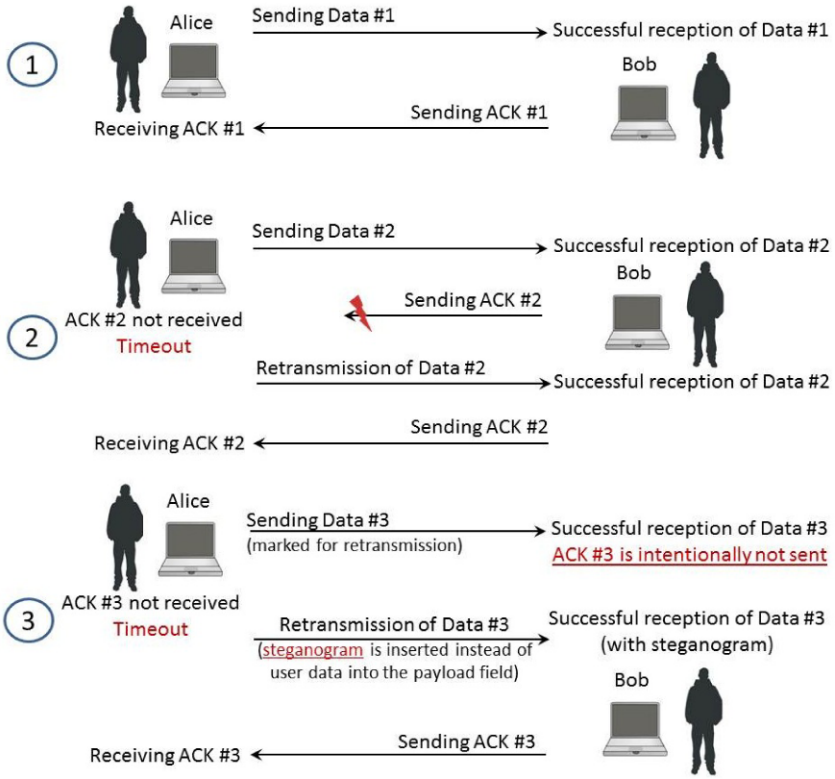


Fig. 1 Generic retransmission mechanism based on timeouts (1, 2); RSTEG (3)

This paper is an extension and summarization of previous work on RSTEG which was presented in [4] and [5] and focuses on RSTEG real network traffic analysis and detection.

2 General idea of RSTEG

RSTEG can be used for all protocols that utilise retransmissions at different layers of OSI RM. A generic retransmission mechanism based on timeouts is presented in Fig. 1. RSTEG may be applied also to other retransmission mechanisms in TCP, such as FR/R (Fast Retransmit and Recovery) or SACK (Selective Acknowledgement).

In a simplified case, a typical protocol that uses a retransmission mechanism based on timeouts obligates a receiver to acknowledge each received packet (Fig. 1, case 1). When the packet is not successfully received, no acknowledgment is sent after the timeout expires, and so the packet is retransmitted (Fig. 1, case 2).

As mentioned in Sect. 1, RSTEG uses a retransmission mechanism to reliably exchange steganograms. Both a sender and a receiver are aware of the steganographic procedure. At some point during the connection after successfully receiving a packet, the receiver intentionally does not issue an acknowledgment message. In a normal situation, a sender is obligated to retransmit the lost packet when the timeframe within which packet acknowledgement should have been received expires. In the context of RSTEG, a sender replaces original payload with a steganogram instead of sending the same packet again. When the retransmitted packet reaches the receiver, it can extract hidden information (Fig. 1, case 3).

Note that the steganogram sender and receiver may not be simultaneously the sender and receiver of the overt communication; in some scenarios they can be both located at network intermediate nodes. In this case it is harder to uncover the steganographic communication as the typical location of the node used for steganalysis is near the sender or receiver of the overt transmission.

The performance of RSTEG depends on many factors, such as the details of the communication procedure (in particular the size of the packet payload, the rate at which segments are generated and so on). No real-world steganographic method is perfect; whatever the method, the hidden information can be potentially discovered. In general, the more hidden information is inserted into the data stream, the greater the chance that it will be detected, for example, by scanning the data flow or by some other steganalysis methods. Detection possibilities details are presented in Sect. 3.5.

3 TCP-based RSTEG: functioning, detection and traffic analysis

Applying RSTEG to TCP is the natural choice for IP networks, as a vast amount of Internet traffic (about 80–90%)

is based on this protocol. For TCP, the following retransmission mechanisms are defined:

- **RTO** (Retransmission Timeouts) in which segment loss detection is based on RTO timer expiration. Results from [6] show that 60–88% of all retransmissions on the Internet were caused by RTO mechanism. In RTO, a segment is considered lost if the receiver does not receive an acknowledgement segment (ACK) after the specified period of time, after which it is retransmitted. The RTO timer value varies in TCP implementation across different operating systems, and it depends mainly on RTT (Round Trip Time) and its variation. If the RTO timer is set to too low of a value, it may cause too many spurious retransmissions; otherwise, the sender will be waiting too long to retransmit a lost segment, which may cause throughput decrease.
- **FR/R** (Fast Retransmit/Recovery) is based on detecting duplicate ACKs (that is, ACKs with the same acknowledgement number). A receiver acknowledges all segments delivered in order. When segments arrive out of order, the receiver must not increase the acknowledgement number so as to avoid data gaps but instead sends ACKs with unchanged acknowledgement number values, which are called duplicate ACKs (dupACKs). Usually, a segment is considered lost after the receipt of three duplicate ACKs. Issuing duplicate ACKs by the receiver is often a result of out-of-order segment delivery. If the number of duplicate ACKs that triggers retransmission is too small, it can cause too many retransmissions and can degrade network performance.
- **SACK** (Selective Acknowledgement) is based on Fast Retransmit/Recovery. It uses an extended ACK option that contains blocks edges to deduce which received blocks of data are non-contiguous. When retransmission is triggered, only missing segments are retransmitted. This feature of SACK decreases network load.

3.1 RSTEG insertion and extracting procedures for TCP

The intentional retransmissions caused by RSTEG should be kept at a reasonable level to avoid detection. To achieve this goal, it is necessary to determine the average number of natural retransmissions in TCP-based Internet traffic as well as to know how intentional retransmissions affect the network retransmission rate. Usually network retransmissions are caused by network overload, excessive delays or reordering of packets [6], and their number is estimated to account for up to 7% of all Internet traffic [6, 7].

RSTEG can be applied to all retransmission mechanisms presented above. It requires modification to both a sender and a receiver. A sender should control the insertion procedure and decide when a receiver should invoke a retransmission. The sender is also responsible to keep the number of retransmissions at a non-suspicious level. The receiver's role is to detect when the sender indicates that intentional retransmission should be triggered. Then, when the retransmitted segment arrives, the receiver should be able to extract the steganogram.

The sender must be able to mark segments selected for hidden communication (that is, retransmission request segments) so the receiver would know which segments retransmissions should be invoked and which segments will contain steganograms. However, marked TCP segment should not differ from those sent during a connection. The following procedure for marking sender segments is proposed. Let us assume that the sender and receiver share a secret Steg-Key (*SK*). For each fragment chosen for steganographic communication, the following hash function (H) is used to calculate the Identifying Sequence (*IS*):

$$IS = H(SK \| Sequence\ Number \| TCP\ Checksum \| CB) \quad (3.1)$$

Note that as H it could be used any secure hash function. *Sequence Number* and *TCP Checksum* denote values from the chosen TCP header fields in segments, $\|$ is the bits concatenation function, and *CB* is a control bit that allows the receiver to distinguish a retransmission request segment from a segment with a steganogram. For every TCP segment used for hidden communications, the resulting *IS* will have different value due to the variety of values in the *Sequence Number* and *TCP Checksum* header fields. All *IS* bits (or only selected ones) are distributed by the sender across a segment's payload field in a predefined manner. The receiver must analyse each incoming segment; based on *SK* and values from the TCP header, the receiver calculates two values of *IS*, namely, one with $CB = 1$ and one with $CB = 0$. Then the receiver checks if and which *IS* is present inside the received segment.

Problems may arise when the segment that informs the receiver of a necessity to invoke an intentional retransmission (which contains user data together with the *IS*) is lost due to network conditions. In that case, a normal retransmission is triggered, and the receiver is not aware that the segment with hidden data will be sent. However, in this case, the sender believes that retransmission was invoked intentionally by the receiver, and so he/she issues the segment with steganogram and the *IS*. In this scenario, user data will be lost, and the cover connection may be disturbed.

In order to address the situation in which the receiver reads a segment with an unexpected steganogram, the receiver should not acknowledge reception of this segment until he/she receives the segment with user data. When the ACK is not sent to the sender, another retransmission is invoked. The sender is aware of the data delivery failure, but he/she does not know which segment to retransmit, so he/she first issues a segment with user data. If delivery confirmation is still missing, then the segment with steganogram is sent.

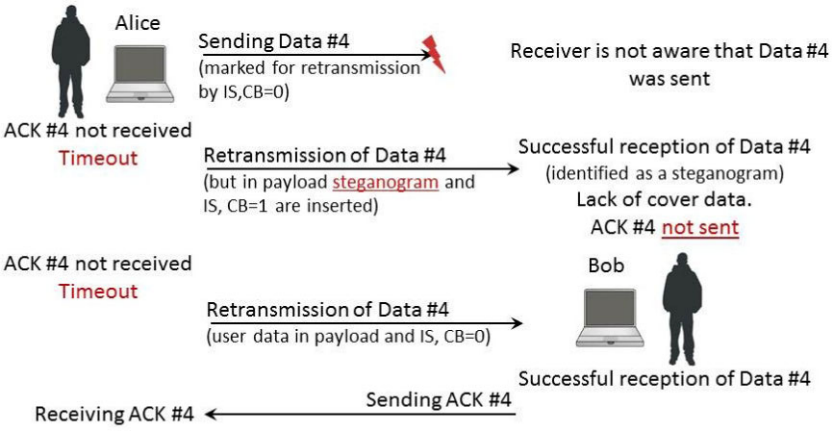


Fig. 2 RTO-based RSTEG segment recovery example

The situation continues until the sender receives the correct ACK. This mechanism of correcting steganogram network losses is illustrated in Fig. 2.

For example, consider the scenario in which 0.5% intentional retransmissions are invoked. If 5% of them are lost, it means that the above-described mechanism will take place only for 0.025% of steganogram segments, thus it will occur rarely.

RSTEG may be applied to the retransmission mechanisms presented previously as follows:

- **RTO-based RSTEG**: The sender marks a segment selected for hidden communication by distributing the *IS* across its payload. After successful segment delivery, the receiver does not issue an ACK message. When the RTO timer expires, the sender sends a steganogram inside the retransmitted segment's payload (see Fig. 1). The receiver extracts the steganogram and sends the appropriate acknowledgement.
- **FR/R-based RSTEG**: The sender marks the segment selected for hidden communication by distributing the *IS* across its payload. After successful segment delivery, the receiver starts to issue duplicate ACKs to trigger retransmission. When the ACK counter at the sender side exceeds specified value, the segment is retransmitted. Payload of the retransmitted segment contains a steganogram. The receiver extracts the steganogram and sends an appropriate acknowledgement.
- **SACK based RSTEG**: The scenario is exactly the same as FR/R, but in the case of SACK, it is possible that many segments are retransmitted because of potential non-contiguous data delivery.

3.2 RSTEG implementation

Experimental RSTEG implementation has been done on Linux 2.6.27.7-9 kernel, which supports all mentioned retransmission mechanisms. It allows measuring steganographic bandwidth, retransmission difference and other RSTEG parameters introduced in Sect. 3.3. RSTEG-related important modifications to the TCP/IP stack are described below.

3.2.1 *Sending procedure modifications*

The sending phase was modified to queue steganograms delivered by application layer and wait for cover data (which also come from the application). When cover data comes to Linux kernel, *IS* (*Identifying Sequence*) is inserted to mark segment for retransmission. In this experimental version predefined data at the end of the segment is used as *IS* to recognize it easily in network traffic dumps. In non-experimental version the *IS* should be distributed across segments payload field.

The Linux TCP/IP stack has some data transfer and kernel operations' optimizations e.g. data collation, putting data in many blocks in memory. These optimizations cannot be used successfully along with RSTEG. The first one could cause joining segments containing secret and cover data. Joining only cover or secret data in one segment does not affect steganographic transmission, but in this experiment we turned off this mechanism to simplify procedure. The second optimization mechanism is used with network cards that support scatter-gather operations, in other cases all user data is linearized by kernel. Changing data, which is split in many memory blocks, requires complicated operations, that is why linearization of user data was always used.

3.2.2 *Receiving procedure modifications*

Receiver's task is to recognize segments containing *IS*. The sequence is computed for each incoming segment and compared with extracted one. If segment is recognized as RSTEG retransmission request, then no ACK is sent, otherwise data is acknowledged and delivered to an application.

If steganogram arrives and receiver detects lack of retransmission request for this segment then no ACK is sent and recovery procedure is applied (see Sect. 3.1).

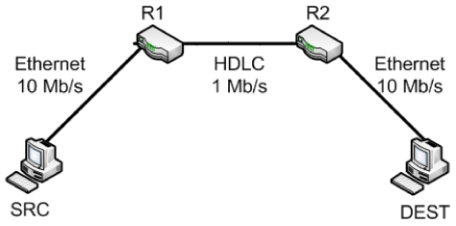


Fig. 3 Experimental network topology

Among all kernel optimization mechanisms only adjacent segment collapsing is unable to work with RSTEG. Adjacent segment collapsing joins data from many segments into one block before delivering to application. RSTEG requires delivering segments separately because steganograms are recognized also in application layer.

3.2.3 Retransmission procedure modifications

Retransmission procedure is the most important phase of steganographic communication. For each segment marked for retransmission by RSTEG, retransmission counter is created. If retransmission is triggered and counter is zero or even then the payload is replaced by steganogram, otherwise segment is retransmitted without change (recovery procedure).

After data replacement it is necessary to update also the checksum unless network card supports *TCP Checksum Offload*, which processes checksum calculation on the network card.

3.3 Experiment methodology

The network topology in Fig. 3 was designed to fit Internet traffic retransmission statistics (see Sect. 3.1). The source node (SRC) transmits TCP traffic and UDP background traffic to destination node (DEST) through 1 Mb/s bottleneck, which causes natural retransmissions.

Traffic parameters presented as below were matched to achieve ∼3–4% of natural retransmissions with zero size buffers on routers R1 and R2.

Simulation parameters (when network retransmission probability (NR_P) equals 3%)

- TCP throughput: 125 kb/s
- UDP throughput: 1 160 kb/s
- Transmission time: 600 s
- Measured time: 540 s
- Measure start delay: 60 s
- Payload size: 1 200 B

Network traffic was measured for 9 minutes, starting after 1 minute from the beginning of transmission. The RSTEG intentional retransmission probability (IR_P) was changed from 0 to 5% with intermediary steps at 0%, 1%, 2%, 3%, 4% and 5%.

In the above simulation scenario, five parameters were measured:

- **Steganographic Bandwidth** (S_B)—the amount of the steganogram transmitted using RSTEG during one second [B/s]. Parameter may be expressed as

$$S_B = \frac{N_S \cdot S_S}{T} \text{ [B/s]}$$

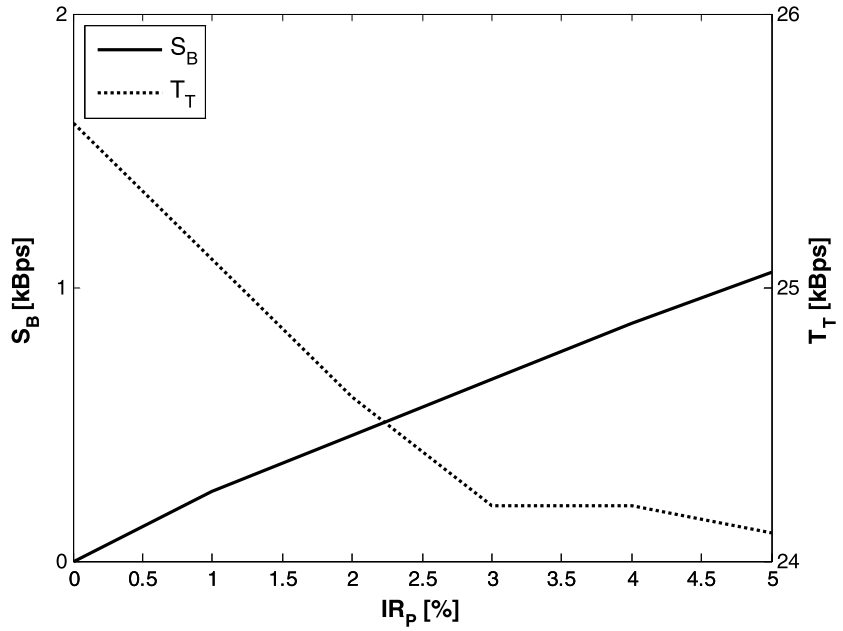
  where N_S denotes the number of segments used for hidden communication, S_S—the size of segment payload and T—the duration of the connection. S_B was measured by receiver̀s application, which counted number of the successfully received steganograms and their size.
- **Retransmissions Difference** (R_D)—the difference between retransmissions in the network after applying RSTEG and in the network before applying RSTEG. This parameter can be used to estimate the influence that RSTEG has on the TCP retransmissions rate. Thus, it illustrates how to choose the correct intentional retransmission probability to limit the risk of detection.
  R_D was measured with *Wireshark* sniffer (www.wireshark.org) in pcap traffic dumps by counting segments suspected to be retransmitted.
- **Steganographic Retransmissions Ratio** (S_R)—the amount of steganographic retransmissions in all retransmissions.
- **TCP Throughput** (T_T)—the effective TCP throughput measured on the DEST node.
- **Effective** IR_P—IR_P measured on the DEST node.

3.4 Traffic analysis of TCP-based RSTEG TCP

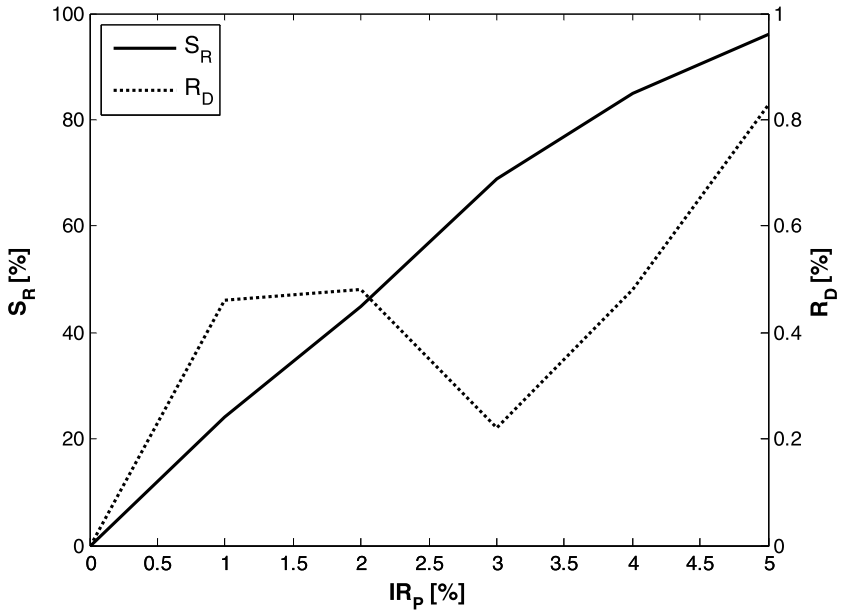
The results achieved in the experiment are presented in Fig. 4 and Table 1. The results prove that bandwidth of the steganographic channel is increasing together with intentional retransmissions, while the increase of R_D is slower as shown in Fig. 4(a). In real world TCP/IP stack implementation the congestion avoidance algorithms are reducing congestion window, which is natural response to retransmissions. That effect causes throughput reduction and smaller retransmissions difference than was intentionally triggered in [1%; 2%] IR_P range. When IR_P reaches 3% its value is around number of retransmissions for normal network conditions (RSTEG-free) and R_D is non-zero because of random choice of segments marked for retransmission. Next, the IR_P reaches up to 5%, which causes more retransmissions in the network again, but still fewer than number of the intentional retransmissions because of congestion avoidance algorithms.

Figure 4(b) shows that in network congestion state intentional retransmissions are reducing the number of natural ones (but of course total number of retransmissions is

Fig. 4 Traffic analysis of TCP-based RSTEG



(a) Steganographic Bandwidth (S_B) and TCP Throughput (T_T)



(b) Steganographic Retransmissions Ratio ($S\ R$) and Retransmissions Difference (R_D)

Table 1 Summarized experimental results

IR_P	S_B		R_D		S_R		TCP throughput		Effective IR_P	
	[B/s]	σ	[%]	σ	[%]	σ	[kB/s]	σ	[%]	σ
0%	0	0	0	0	0	0	25.6	0.2	0	0
1%	252	18	0.46	0.23	24	2.9	25.1	0.2	1.01	0.08
2%	461	37	0.48	0.23	45	4.3	24.6	0.1	1.87	0.15
3%	665	37	0.22	0.18	69	3.4	24.2	0.2	2.74	0.16
4%	867	36	0.48	0.16	85	5.1	24.2	0.1	3.58	0.15
5%	1056	58	0.83	0.2	96	2.3	24.1	0.1	4.39	0.24

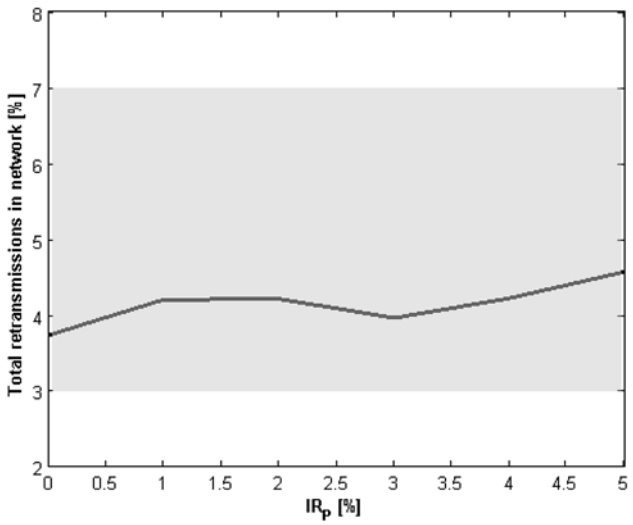


Fig. 5 Total network retransmissions for given RSTEG intentional retransmission probability

still higher). The optimal intentional retransmissions level is 5% when almost all retransmitted segments are used for steganographic communication.

Keeping total retransmissions number on reasonable level is necessary to avoid detection. The experimental results show that we can achieve decent steganographic bandwidth while maintaining non-suspicious level of retransmissions.

3.5 RSTEG detection

Retransmissions in IP networks are a "natural phenomenon", and so intentional retransmissions introduced by RSTEG are not easy to detect if they are kept at a reasonable level. The experimental results presented here show that RSTEG is a very effective and hard to detect steganographic method. Moreover, if the sender can observe the average retransmission rate in a network, then it can also choose an IR_P so as to limit the risk of detection.

One possible detection method is statistical steganalysis based on the network retransmission rate. If for certain TCP connections, the retransmission rate are significantly higher than for others, then potential usage of RSTEG may be detected. Such a steganalysis method involves monitoring of the TCP retransmission rates for all connections in a sub-network. If RSTEG is utilized in Internet and the total retransmissions level does not exceed estimated average Internet retransmissions level (3–7% [6, 7]) there is only small possibility to detect steganographic communication. The results in Fig. 5 show that the total retransmissions level fits this range, even if RSTEG intentional retransmissions level reaches 5% which means almost 1 kB/s steganographic bandwidth.

However, there is a solution that makes the steganalysis of TCP-based RSTEG easier to perform. The proposed steganalysis method may be implemented with a passive warden [8] (or some other network node responsible for steganographic communication detection). Passive warden must be able to monitor all the TCP traffic and for each TCP connection it must store sent segments for the given period of time, which depends on the retransmission timer i.e. passive warden must store the segment until it is acknowledged by the receiver so the retransmission is not possible any more. When there is a retransmission issued, passive warden compares originally sent segment with retransmitted one and if the payload significantly differs RSTEG is detected and the segment is dropped. However, it should be noted that this may cause serious performance issues if passive warden monitors all the TCP connections and must store a large number of the segments.

On the other hand, it must be noted that based on results presented in [9] up to 0.09% (1 in 1100) of TCP segments may be corrupted due to network delivery. As a result, an imperfect copy of a segment may be sent to the receiver. After reception of the invalid segment, verification is performed based on the value in the *TCP Checksum* field, and the need to retransmit is signalled to the sender. Thus, in this scenario, the original segment and the retransmitted one will differ from each other. Occurrences of this effect in IP networks mask the use of RSTEG. Thus, the steganalysis methods described above may fail, because the warden will drop retransmitted segments when differences among segments are discovered, and as a result, user data will be lost.

It is worth noting that even for the low rates of intentional retransmission (0.09%) that are required to mask RSTEG, if we assume that the TCP segments are generated at a rate of 200 segments/s, with the connection lasting 5 minutes and the segment's payload size being 1000 bytes, then this results in $S_B = 180$ Bps, which should be considered as high steganographic bandwidth.

To summarise, measures to detect RSTEG have been proposed and can be utilised, but if the rate of intentional retransmissions is very low, the detection of hidden communications may be difficult.

4 Conclusions

RSTEG is an intra-protocol hybrid steganography method. This paper described how RSTEG can be integrated into Linux kernel. Experimental results obtained on real TCP/IP stack traffic analysis were presented.

Achieved results proved that RSTEG is an effective steganographic method which offers high steganographic bandwidth when compared with other network steganography methods. The steganographic bandwidth depends mainly on RSTEG intentional retransmissions level (IR_P) which affects number of usually retransmitted segments during connection. RSTEG causes less increase of total retransmissions level than IR_P level which makes it harder to detect. Keeping total retransmissions at reasonable level is crucial for RSTEG otherwise it may be vulnerable to detection.

RSTEG steganalysis may be based on a passive warden which must be able to monitor all the TCP traffic. For each TCP connection it must store segments sent until they are acknowledged by the receiver so the retransmission is not possible any more. When retransmission occurs, passive warden compares originally sent segment with retransmitted one and if there is a difference existence of RSTEG is detected and the segment is dropped. However, such warden behavior may lead to serious performance issues because of need to monitor all the TCP connections and storing a large number of the segments.

Acknowledgement This work was supported by the Polish Ministry of Science and Higher Education under Grants: N517 071637 and IP2010 025470.

References

1. Jankowski, B., Mazurczyk, W., & Szczypiorski, K. (2010). Information hiding using improper frame padding. In *Proc. of 14th international telecommunications networks strategy and planning symposium (NETWORKS)*, 27–30 September 2010 (pp. 77–82). ISBN 978-1-4244-6703-7.
2. Szczypiorski, K. (2003). HICCUPS: hidden communication system for corrupted networks. In *Proc. of: ACS'2003*, Miedzyzdroje, Poland, October 22–24, 2003 (pp. 31–40).
3. Berk, V., Giani, A., & Cybenko, G. *Detection of covert channel encoding in network packet delays* (Tech. Rep. TR2005-536). Department of Computer Science, Dartmouth College, Nov. 2005. URL: http://www.ists.dartmouth.edu/library/149.pdf.
4. Mazurczyk, W., Smolarczyk, M., & Szczypiorski, K. (2009). Retransmission steganography and its detection. *Soft Computing*, *15*(3), 505–515.
5. Mazurczyk, W., Smolarczyk, M., & Szczypiorski, K. (2010). Retransmission steganography applied. In *Second international workshop on network steganography (IWNS) co-located with the 2010 international conference on multimedia information networking and security (MINES 2010)*, Nanjing, China, November 4–6, 2010.
6. Rewaskar, S., Kaur, J., & Smith, F. (2007). A performance study of loss detection/recovery in real-world TCP implementations. In *Proc. of the IEEE international conference on network protocols, ICNP 2007*, Beijing, China, October 16–19, 2007 (pp. 256–265). ISBN 1-4244-1588-8.
7. Chen, C., Mangrulkar, M., Ramos, N., & Sarkar, M. *Trends in TCP/IP retransmissions and resets*. (Technical Report). URL: http://www-cse.ucsd.edu/classes/wi01/cse222/projects/reports/tcp-flags-13.pdf.
8. Fisk, G., Fisk, M., Papadopoulos, C., & Neil, J. (2002). Eliminating steganography in Internet traffic with active wardens. In *Lecture notes in computer science: Vol. 2578. 5th international workshop on information hiding* (pp. 18–35).
9. Stone, J., & Partridge, C. (2000). When the CRC and TCP checksum disagree. In *Proc. of SIGCOMM 2000* September 2000.

Wojciech Mazurczyk holds an M.Sc. (2004) and a Ph.D. (2009) in telecommunications from the Faculty of Electronics and Information Technology, Warsaw University of Technology (WUT, Poland) and is now an Assistant Professor at WUT and the author of over 50 scientific papers and over 25 invited talks on information security and telecommunications. His main research interests are information hiding techniques, network security and multimedia services. He is also a research co-leader of the Network Security Group at WUT (secgroup.pl). Personal website: http://mazurczyk.com.

Miłosz Smolarczyk holds an B.Sc. (2010) in telecommunications from the Faculty of Electronics and Information Technology, Warsaw University of Technology (WUT). He is Network Security Group (secgroup.pl) member. His research interests include network security and steganography.

Krzysztof Szczypiorski holds an M.Sc. (1997) and a Ph.D. (2007) in telecommunications both with honours from the Faculty of Electronics and Information Technology, Warsaw University of Technology (WUT), and is an Assistant Professor at WUT. He is the founder and head of the International Telecommunication Union Internet Training Centre (ITU-ITC), established in 2003. He is also a research leader of the Network Security Group at WUT (secgroup.pl). His research interests include network security, steganography and wireless networks. He is the author or co-author of over 110 publications including 65 papers, two patent applications, and 35 invited talks.