Two Methods for Detecting Malware

Maciej Korczyński, Gilles Berger-Sabbatel, and Andrzej Duda

Grenoble Institute of Technology, CNRS Grenoble Informatics Laboratory UMR 5217
681, rue de la Passerelle, BP 72
38402 Saint Martin d'Hères Cedex, France
`Maciej.Korczynski@imag.fr,Gilles.Berger-Sabbatel@imag.fr,Andrzej.Duda@imag.fr`

Abstract. In this paper, we present two ways of detecting malware. The first one takes advantage of a platform that we have developed. The platform includes tools for capturing malware, running code in a controlled environment, and analyzing its interactions with external entities. The platform enables us to detect malware based on the observation of its communication behavior. The second approach uses a method for detecting encrypted Skype traffic and classifying Skype service flows such as voice calls, skypeOut, video conferencing, chat, file upload and download in Skype traffic. The method is based on the Statistical Protocol IDentification (SPID) that analyzes statistical values of some traffic attributes. We apply the method to identify malicious traffic—we have successfully detected the propagation of Worm.Win32.Skipi.b that spreads over the Skype messenger by sending infected messages to all Skype contacts on a victim machine.

1 Introduction

In this paper, we consider the problem of detecting malware. The standard detection method consists of identifying malware by searching some patterns in the code (a signature). The signature method however suffers from several drawbacks and needs to be enhanced with other approaches. We consider two other methods that may provide an additional possibility of detecting malware. The first one is based on the observation of the communication behavior caused by suspected code. It takes advantage of our platform for botnet-related malware analysis. It is composed of tools for capturing malware, running code in a controlled environment, and analyzing its interactions with external entities.

The second method provides a means for detecting encrypted Skype traffic and classifying Skype service flows such as voice calls, skypeOut, video conferencing, chat, file upload and download in Skype traffic. The method is based on the Statistical Protocol IDentification (SPID) that analyzes statistical values of some traffic attributes. We apply the method to identify malicious traffic—we have successfully detected the propagation of Worm.Win32.Skipi.b that spreads over the Skype messenger by sending infected messages to all Skype contacts on a victim machine. We have focused on the selection of an appropriate set of attribute meters based on propagation characteristics to classify malicious flows with high accuracy.

2 Platform for botnet-related malware analysis

We have presented the design of the platform for botnet-related malware analysis and its applications in the previous work [1,2,3]. It has the following functionalities (cf. Figure 1):

– *Malware capture*: for this purpose, we use Dionaea, which a popular low-interaction honeypot, that mainly captures malware propagated through vulnerabilities in the Microsoft SMB services.
– *Malware classification*: as soon as malware is captured, it is automatically classified according to the network connections it attempts to perform to contact its command and control service (C&C) [3]. To this end, malware is run on a virtual machine without actual connection to the Internet but with a DNS service provided by the host machine. The queried DNS addresses and attempted connections are observed and recorded with the Mwna software tool briefly described in Section 2.1. This allows detecting malware with really unknown behavior thus avoiding the analysis of already known malware.
– *Analysis of malware network activity*: it is performed under the control of an operator using Mwna. The analysis focuses on identifying the C&C and detecting malicious activities.

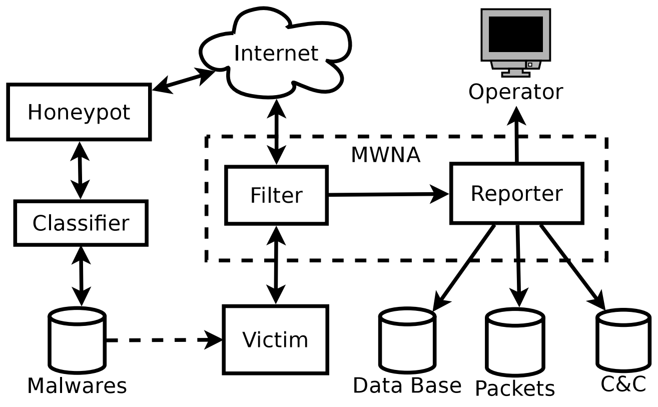


Fig. 1. Architecture of the platform

2.1 Mwna

Mwna (Malware Network Analyzer) runs on a gateway between a computer (real or virtual) infected by malware (the victim) and the Internet. It is composed of two programs communicating through Tcp sockets, the *filter*, and the *reporter*:

- *The filter*: it uses the *netfilter_ queue* mechanism of Linux to intercept packets flowing through the network stack in the kernel. The process of packet matching can be specified by rules (drop, trace, higher-level analysis, etc.). Packets are analyzed up to the application level when necessary. It controls connection establishment, its termination, and DNS activity (queries and replies). A number of application-level protocols can be detected and at least partially analyzed even if they do not run on their standard ports: IRC, HTTP, SMTP. The HTTPS protocol is identified and verified up to the point it enters the encrypted mode (but certificates are not currently checked). The NTP, IPP, and the Netbios Name and Session service protocols are verified when they run on their standard ports. Unknown textual protocols can also be detected.

  The *filter* includes a system of dynamically loaded plugins allowing to extend it with higher level functionalities that include monitoring and blocking of malicious activities:
  - Interaction with C&C: in the current state of the program, C&C can mainly be detected as IRC-based protocols. C&C based on non standard textual or binary protocols and HTTP can also be detected with few false positives. When C&C interaction through IRC is detected, all communication between the victim and the attacker is recorded.
  - ICMP and TCP scans are detected and blocked.
  - Denial of Service attacks (DoS): TCP SYN flooding, UDP flooding, and HTTP flooding attacks are detected and blocked by dropping packets sent to the attacked host.
  - Mail transmission attempts are detected and can optionally be blocked by resetting the connection to the mail server.
  - Spam transmission can be detected based on attempts to send a mail through multiple relays or to multiple different destinations.
  - The HTTP payload is analyzed to detect the transmission of Windows executables. The detection is based on the identification of executable headers and does not rely on the Content-type headers or file name extensions.

  As the *filter* is interfaced with the kernel, it must be executed with the administrator privileges. Detected events are sent to the *reporter* using a very simple textual protocol.
- *The reporter*: it is in charge of displaying messages for the operator, recording events in a Sqlite database, storing communication traces in a file, and recording C&C interactions. It does not need to be executed with the administrator privileges.

The separation of MWNA into several processes has several advantages:

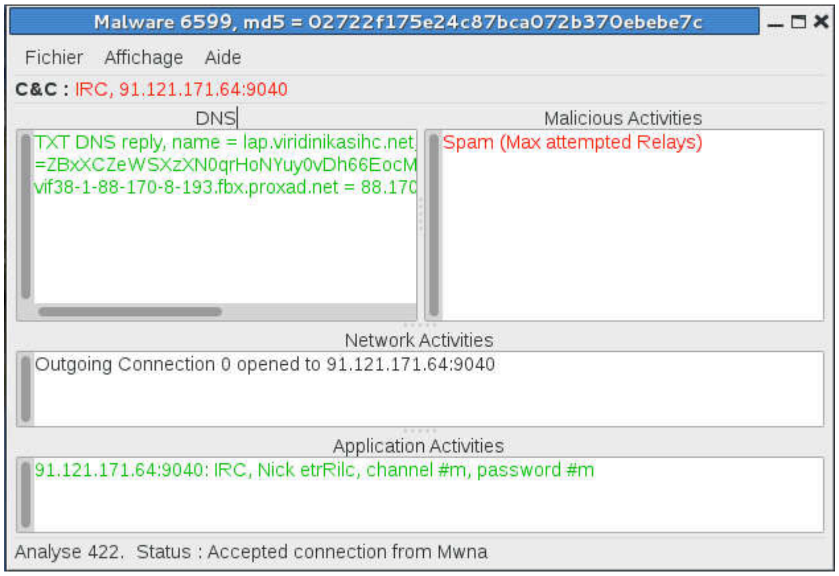
- As the *reporter* runs without administrator privileges, it can use high level tools such as database or graphic user interface without security risks.
- High latency operations (e.g. user interaction, disk I/O) are removed from the critical path of packet processing.
- Better performance can be achieved on multi-core architectures.

- The *reporter* can be run on a separate machine or on a production network, without security risks.
- The *reporter* could be eventually replaced with a program performing a fully automatic analysis of the malware network activity.

2.2 Detecting malware activities

In this section, we will present two examples of malware activities. In the first one, we will present the graphic interface of *Mwna* with the malware that at the first glance seems to attempt sending a spam. The second example is the malware actually sending spam.

2.3 Port scanning malware



Malware 6599, md5 = 02722f175e24c87bca072b370ebebe7c

C&C : IRC, 91.121.171.64:9040

TXT DNS reply, name = lap.viridinikasihc.net
=ZBxXCZeWSXzXN0qrHoNYuy0vDh66EocM
vif38-1-88-170-8-193.fbx.proxad.net = 88.170

Spam (Max attempted Relays)

Outgoing Connection 0 opened to 91.121.171.64:9040

91.121.171.64:9040: IRC, Nick etrRilc, channel #m, password #m

Analyse 422. Status : Accepted connection from Mwna

Fig. 2. Screen capture of Mwna

Figure 2 presents a screen capture of the reporter while analyzing malware captured in December 2012. The title bar of the window shows the MD5 digest of the malware. The line under the menu bar shows that the C&C has been identified as IRC and presents the IP address and the port number.

The right text area under the line indicates the DNS replies. In the studied case, there is only a reply to a TXT query on lap.viridinikasihc.net and a query on the address of the host machine. The reply to the TXT query seems to be some kind of encrypted data, which should include the address of the C&C.

The left text area shows the detected malicious activities: in this case, an attempt to send a spam is detected, because a compromised machine tries to connect to different SMTP servers. All these attempts fail, as the access provider only allows connections to its own SMTP server.

The text area under the previous ones displays network activities: in this case, only the connection to the IRC server.

The bottom text area displays the application-level activities: in this case, the malware joins an IRC channel.

At this point, the tool indicates that spam transmission is attempted, but there are two problems:

– The addresses of the SMTP servers are not DNS resolved,
– There is not enough data exchanged to specify a spam payload and a list of recipients.

We have redirected the connections to the SMTP port to a program that emulates a SMTP server without actually transmitting messages, but we did not find any transmission once the connection is set up. Finally, if we disable the spam detection feature, we can see that these connection attempts are in fact scans on the SMTP port, probably with the goal of discovering open mail relays.

2.4 Spam sending malware

In this second example, the malware communicates with its C&C using a modified IRC protocol identified as an *unknown textual protocol*.

Four DNS queries try to find the mail servers of AOL, Yahoo, Google, and Hotmail, which seems typical for spam sending malware. Then, the malware downloads several files from HTTP servers. Five of them are Windows executables and two others are queries to get the official IP address and a domain name of the machine on which the malware runs.

Then, the malware attempts to connect to mail servers and spam transmission is detected.

As in the previous case, we have analyzed the malware while the SMTP traffic is redirected to a local program. We could see two other downloads that appear to be encrypted data, hence, the spam payload and the recipient list may be contained in these files or encapsulated in the downloaded executable files. 383 mails have been sent in 5 minutes using 83 connections to many mail relay addresses. In most connections, the malware only sent one mail, or no mail at all, but in 17 cases, it sent up to 164 emails. All these mails are advertisements for a russian site selling drugs.

2.5 Discussion

These two examples show that our platform allows to detect and identify malicious activities of malware. Spam transmissions seem to be the most frequent activity. In this case, we are able to gather enough data to determine the strategy of spamming tools and to intercept the spam payload.

3 Classification method of encrypted Skype traffic

In this section, we present the second method for detecting malware activities. More specifically, we identify malicious flows concealed in encrypted TCP Skype traffic tunneled over the TLS/SSL protocol [4]. We have considered a worm called Worm.Win32.Skipi.b alias Skipi [5] that spreads over the Skype Instant Messaging (IM) system.

We apply a method for classifying Skype service flows [6,7] based on the Statistical Protocol IDentification (SPID) algorithm [8]. SPID is based on *traffic models* that contain a set of *attribute fingerprints* represented as probability distributions. They are created through frequency analysis of traffic properties called *attribute meters* of application layer data or flow features. An example of such an attribute meter is *byte frequency* that measures the frequency at which all of the possible 255 values occur in a packet. Other attribute meters defined in detail later include for instance byte offset, byte re-occurring, direction change, and packet size.

SPID operates in three steps. First, packets are classified into bi-directional flows. All connections are represented as 5-tuples according to the source IP address, source port, destination IP address, destination port, and transport layer protocol. However, only packets carrying data are significant, because the analysis is based on both the application layer data and flow features. Then, each flow is analyzed in terms of attribute meters to obtain a collection of attribute fingerprints associated with a particular type of traffic.

In the initial training phase, the method creates *traffic models*—attribute fingerprints representative for the traffic we want to detect. During the classification phase, the method computes attribute fingerprints on the flows to classify and compares them with traffic models by means of the Kullback-Leibler (K-L) divergence [9]:

$$D(P||Q) = K\text{-}L(P,Q) = \sum_{x \in X} P(x) log_2 \frac{P(x)}{Q(x)}. \tag{1}$$

The K-L divergence is a measure of the difference between two probability distributions $P(x)$ and $Q(x)$. $P(x)$ represents the distribution of a particular attribute of an observed flow and $Q(x)$ is the distribution corresponding to a known traffic model. Classification consists in comparing $P(x)$ with all known traffic models and selecting the protocol with the smallest average divergence $D(P||Q)$ and greater than a given threshold. We need to correctly set the divergence threshold to decrease the false positive rate for known traffic models—we only take into consideration the K-L divergence average values above the threshold.

In the first phase, it detects Skype traffic after a TCP three-way handshake based on the first five packets of the connection by considering some attribute meters reflecting application level data. Then, it changes the set of attribute meters to payload independent features to detect service flows in the Skype traffic: voice/video, skypeOut, chat, and file transfer. This phase requires a larger

Table 1. Attribute meters used in classification.

No.	Attribute meter name	Inspected bytes per packet	Inspected packets per flow
1	byte-frequency	100	8
2	action-reaction of first bytes	3	6
3	byte value offset hash meter	32	4
4	first 4 packets byte reoccurring distance with byte	32	4
5	first 4 packets first 16 byte pairs	17	4
6	first 4 ordered direction packet size	0	4
7	first packet per direction first N byte nibbles	∼8	1
8	packet size distribution	0	All
9	direction packet size distribution	0	All
10	byte pairs reoccurring count	32	All
11	first server packet first bit positions	∼16	1

number of packets to analyze to be effective: our calibration sets this value to 450 packets. Finally, the method considers more packets (the threshold is set to 760) to further distinguish between voice and video flows, and between file upload and download.

3.1 Attribute meters for Skype

Table 1 presents the set of attribute meters that we have defined for classifying Skype traffic.

- **byte frequency**: in each packet it measures and returns the frequency of individual bytes in the payload. Encrypted data seems to have equally distributed byte frequencies, whereas the plain text may exhibit different distributions. The SSL protocol tends to provide some unencrypted information related to a session, such as SSL version, message type, compression method selected by the server, etc., in the first bytes of the encrypted packets.
- **action-reaction of first bytes**: it creates hash values based on the first 3 bytes of each packet that was sent in a different direction than the previous one. It is better to analyze packets sent alternately in different directions instead of looking at all packets, because we can easily analyze the request-response phase between a client and a server. The meter is especially useful in primary identification of a SSL Skype connection.
- **byte value offset hash**: it combines individual byte values in each packet with the offset at which the bytes are positioned. The meter considers up to 32 bytes of the 4 first packets. The SSL is one of the protocols that use several positions in particular packets (e.g. in `Client Hello` or `Server Hello` messages). As a result, the combination of bytes with their positions provides some additional information with respect to the byte frequency.
- **first 4 packets byte reoccurring distance with byte**: it creates a short hash value (usually a 4-bit representation) and combines it with the distance

between the two occurrences. The measurement detects the bytes that occurred more than once within 16 previous bytes. It was specifically created to identify banners in plain text packets like e.g. `TT` in HTTP GET and POST messages, but it also applies to the case of the encrypted SSL content.

- **first 4 packets first 16 byte pairs**: it combines neighboring bytes in a 16-bit value and converts to a 8 bit hash value (the size is determined by the fingerprint length). It analyzes only application layer data regardless of the flow information, i.e. packet size, directions, or inter-arrival times. The meter indicates that there are some specific, not random two-byte combinations like e.g. list compression methods supported by the client in the SSL `Client Hello` message sent to the server.
- **first 4 ordered direction packet size**: the meter returns the compressed version of the packet size that represents a range in which the packet lies instead of the exact value. Measurements are separately done for each of four first packets in connection and the returned value is associated with the packet direction and the order number. It is a flow based attribute created for early traffic recognition.
- **first packet per direction first N byte nibbles**: it analyzes the first packet in each direction and inspects its first few bytes depending on the fingerprint length (8 bytes for a fingerprint length of 256). It provides a measure combining the packet direction, byte offset, and a compact representation of the byte value so-called *nibble*, (it divides a byte into two 4-bit groups, performs an XOR calculation, and returns the resulting 4-bit value). The first packet in each direction and the first few bytes corresponding to these packets say a lot about the application layer protocol and might also provide some hidden information of the underlying service.
- **packet size distribution**: it computes the distribution of the packet size. It provides some hints about the encrypted flows, because the size of Skype packets is somehow deterministic depending on the type of traffic.
- **direction packet size distribution**: this attribute is very similar to the first 4 ordered direction packet size meter. The only difference is that it inspects all packets in a connection and does not mark each measurement with the order number of the packet in a connection. It is an example of a flow based attribute especially suitable for detailed Skype classification: it is able to classify flows in which packet sizes per direction are different, which enables to distinguish file upload from download.
- **byte pairs reoccurring count**: it detects bytes that reoccur in two consecutive packets. In addition, it takes into account the direction of a given packet and its predecessor.
- **first server packet first bit positions**: it looks into the first few bytes of the first packet coming from the server, inspects each bit, and returns the bit values with respect to the bit offset position. The idea is that when connecting to TCP-based services, the server sends some typical welcome messages.

3.2 Identification of Malicious Skype Flows

Finally, we want to test our detection method with malicious flows concealed in Skype traffic. We have considered a worm called Worm.Win32.Skipi.b alias Skipi that spreads over the IM system by sending URL-embedded encrypted chat messages to all Skype contacts on a victim machine. We have selected an appropriate set of attribute meters to use in the SPID algorithm to detect malicious traffic.

Worms propagating in chat messengers have become one of security threats in recent years [10,11,12]. However, existing Internet worm detection mechanisms [13] cannot be directly applied to the detection of malicious flows spreading via instant messengers. The mechanisms cannot distinguish between the encrypted legitimate traffic and messages generated by worms: an infected user does nothing but sending correctly looking messages to other end users.

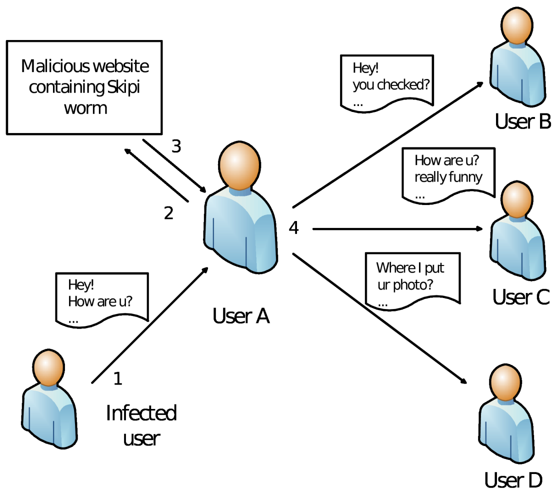

Malicious website containing Skipi worm
3
2
Hey! you checked? ...
User B
How are u? really funny ...
4
User C
Where I put ur photo? ...
Hey! How are u? ...
User A
1
Infected user
User D

Fig. 3. Propagation of the Skipi worm.

In our experiments, we obtain the ground truth information (the traffic generated by Skipi) by running the worm on a laptop having Skype installed. The Skype client had a small number of contacts so that it could easily spread the worm to them. Figure 3 illustrates the propagation process of Skipi through the Skype instant messenger. Let us assume that user A receives a chat message from one of his/her friends after the friend was infected by Skipi (Step 1). The worm message is composed of a selection of some text strings to convince the receiver to open a malicious link to a photo. An example of such a message sent by the worm is shown below:

```
On 30/03/2011, at 10:19, user wrote:
```

```
> how are u ? :)
On 30/03/2011, at 10:19, user wrote:
> I used photoshop and edited it
On 30/03/2011, at 10:19, user wrote:
> http://www.%infectedURL%.jpg
On 30/03/2011, at 10:19, user wrote:
> oops sorry please don’t look there :S
On 30/03/2011, at 10:19, user wrote:
> :)
```

When user A clicks on the link to download the photo, it actually points to the URL where a copy of the Skipi worm is located (Step 2). Once the worm copy is downloaded (Step 3) and executed on the system, it starts to propagate by sending similar messages to all active entries in the contact list (Step 4). We focus on analyzing and detecting worm propagation traffic. We have observed that malicious messages are closely associated with signaling traffic: the client sends TCP segments to check the availability of other users in the contact list. As soon as, it receives a confirmation, Skipi immediately sends the chat messages containing the link redirecting the user to the malware location. After running the malware in different network conditions, we have gathered a collection of TCP flows containing malicious messages in encrypted traffic.

We have selected 6 attribute meters: 3, 7–11 in Table 1 to create the fingerprints of the Skipi traffic. The model was created based on 30 TCP connections corresponding to 1484 observations. For evaluation, we have compared the traffic model with 6 models of Skype flow services used in the previous classification, i.e. voice, video, file upload and download, skypeOut, and, in particular, the legal chat service. We set SPID to inspect 30 first packets containing payload in each connection and used the K-L divergence threshold of 0.8. We run the tests with traces containing Skype flows as well as other Internet traffic composed of various applications.

We have computed $F\text{-}Measure$ defined as:

$$Precision = \frac{TP}{TP + FP}, \tag{2}$$

$$Recall = \frac{TP}{TP + FN}, \tag{3}$$

$$F\text{-}Measure = \frac{2 * Precision * Recall}{Precision + Recall}, \tag{4}$$

for all flows and for each of 6 selected attribute meters. The True Positive (TP) term refers to all malicious Skype flows that were correctly identified as Skipi, False Positives (FPs) refer to all flows that were incorrectly identified as Skipi traffic. Finally, False Negatives (FNs) represent all flows of Skipi traffic that were incorrectly identified as legitimate Skype service flows or other Internet traffic.

Figure 4 presents the F-Measure in function of the number of inspected packets with payload: it rapidly rises after the 18th packet, while after inspecting 30 packets, the F-Measure is equal to 87.5% with Recall of 100%. This means the method has detected all malicious flows with no False Negatives, which is early enough to take prevention actions such as blocking Skype traffic of the user sending malicious chat messages to all Skype contacts after the first attempt. The figure also shows that after the 45th packet, the F-Measure becomes flat due to the fact that two-way communication between the Skipi worm and the victim user contains between 40 and 140 packets depending on the message sent through the chat messenger.

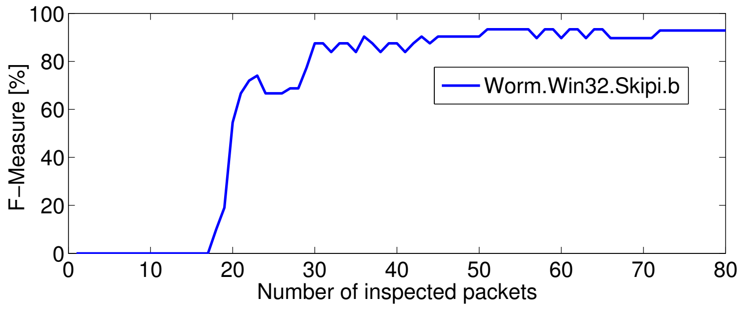


Fig. 4. Detection of the Skipi worm: the F-Measure in function of the number of inspected packets.

4 Conclusions

In this paper, we have presented two methods of detecting malware that go beyond the traditional signature based detection. The first one is based on the observation of the communication behavior caused by suspected code. The detection uses a platform with tools for capturing malware, running code in a controlled environment, and analyzing their interactions with external entities. The second approach uses a method for detecting encrypted Skype traffic and classifying Skype service flows such as voice calls, skypeOut, video conferencing, chat, file upload and download in Skype traffic. The method is based on the Statistical Protocol IDentification (SPID) that analyzes statistical values of some traffic attributes. We have applied the method to identify malicious traffic—we have successfully detected the propagation of Worm.Win32.Skipi.b that spreads over the Skype messenger by sending infected messages to all Skype contacts on a victim machine.

Acknowledgments

This work was partially supported by the EC FP7 project INDECT under contract 218086.

References

1. Gilles Berger-Sabbatel, Maciej Korczyński, and Andrzej Duda. Architecture of a Platform for Malware Analysis and Confinement. In *Proceedings MCSS 2010: Multimedia Communications, Services and Security*, Cracow, Poland, June 2010.
2. Gilles Berger Sabbatel and Andrzej Duda. Analysis of Malware Network Activity. In *Proceedings MCSS 2011: Multimedia Communications, Services and Security*, Cracow, Poland, June 2011.
3. Gilles Berger Sabbatel and Andrzej Duda. Classification of Malware Network Activity. In *Proceedings MCSS 2012: Multimedia Communications, Services and Security*, Cracow, Poland, June 2012.
4. Tim Dierks and Eric Rescorla. The Transport Layer Security (TLS) Protocol, Version 1.2. "RFC 5246", August 2008.
5. Nguyen Cong Cuong. Skype-New Target of the Worm Spreading via IM. http://blog.bkav.com, May 2010.
6. Maciej Korczyński. *Classifying Application Flows and Intrusion Detection in the Internet Traffic*. PhD thesis, École Doctorale Mathématiques, Sciences et Technologies de l'Information, Informatique (EDMSTII), Grenoble, France, November 2012.
7. Maciej Korczyński and Andrzej Duda. Classifying Service Flows in the Encrypted Skype Traffic. *2012 IEEE International Conference on Communications (ICC'12)*, pages 1064–1068, June 2012.
8. E. Hjelmvik and W. John. Statistical Protocol Identification with SPID: Preliminary Results. In *Proceedings of 6th Swedish National Computer Networking Workshop*, May 2009.
9. S. Kullback and R. A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22:49–86, 1951.
10. N. Leavitt. Instant Messaging: a New Target for Hackers. *Computer*, 38(7):20–23, july 2005.
11. Stephen Swoyer. Enterprise Systems: IM Security Exploits Explode in 2007. http://www.esj.com, August 2008.
12. Kaspersky Lab Detects New IM Worms Capable of Spreading via Almost All Instant Messengers. http://www.kaspersky.com, August 2010.
13. Guanhua Yan, Zhen Xiao, and Stephan Eidenbenz. Catching Instant Messaging Worms with Change-Point Detection Techniques. In *Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats*, pages 1–10. USENIX Association, 2008.