J Supercomput (2017) 73:3053–3064
DOI 10.1007/s11227-016-1924-3

A network attack forensic platform against HTTP evasive behavior

Zhen Li[1] · Haiqing Pan[1] · Wenhao Liu[1] · Fei Xu[1] · Zigang Cao[1] · Gang Xiong[1]

Published online: 24 November 2016

Abstract With the increasing amounts of data streams and sophistication of attacks, there is a need for network forensic systems that store and examine very large amounts of network flow data. HTTP, as the most popular protocol on the Internet, is usually exploited to carry malware and evasive attacks besides the normal services. By analyzing HTTP evasive behaviors, a network forensic system can find malware attacks and trace back its origin. In this paper, we study how malware and network attacks in real-world exploit HTTP to hide their malicious activities and present an Evasive Network Attack Forensic System (ENAFS), which can effectively discover evasive network attacks on HTTP and integrally draw the attack samples and their metadata for further analysis. We have run ENAFS on seven days of traffic from the ISP of CSTNET, where it has detected and stored more than 110 million HTTP mismatch instances, covering 1607 different kinds of mismatch types. After further scanning and analyzing these instances, two typical types of evasive attacks have been found. ENAFS can also trace back the origin of an evasive attack which is proved by a case study in this paper.

Keywords Network forensics · HTTP · Evasive attack · Abnormal behavior

1 Introduction

Network forensics is a dedicated investigation technology that enables capture, recording and analysis of network packets and events for investigative purposes. It involves

✉ Gang Xiong
xionggang@iie.ac.cn

Zhen Li
lizhen@iie.ac.cn

[1] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China

monitoring network traffic and determining if there is an anomaly in the traffic and ascertaining whether it indicates an attack [1,2]. Being responsible for more than half of the total traffic volume in the Internet, HTTP has become the preferred protocol for many Internet services and Internet users exchange most of their content via HTTP [3]. At the same time, many malware exploits HTTP to hide their malicious activities and transfer through the Internet due to its popularity and flexibility. Analyzing HTTP traffic is becoming a new focus on network forensics.

Prior analysis of large HTTP traces has shown many inconsistencies and errors may occur in an HTTP packet. One potential error can be made is by relying on the Content-Type header of HTTP to identify the mime type of transferring files. A typical example is that one Content-Type is declared in the HTTP header while in fact the actual data is of another different type to evade security inspections. Such kind of behavior has been observed in some famous malware, such as Zeus, Torpig, Bredolab [4]. Some Advanced Persistent Threat (APT) attacks, like APT30 [5] and APT Operation Poisoned Helmand [6], also used this trick to bypass detection. Therefore, designing a forensic system to record and analyze the malware and APT attacks behind the evasive behaviors are very valuable for discovering potential network threat and enhancing network security.

However, the Content-Type mismatches in HTTP take 35% of the total HTTP volume [3], and most of them are caused by innocent configuration mistakes. Therefore, it is not a sensible way to simply detect the mismatch by a general rule since there must be a lot of false positives. Meanwhile, the traditional rule-based intrusion detection systems (IDS), e.g. Snort [7,8], can only detect limited known Content-Type mismatch attacks based on their rules with few clew in logs which is helpless to investigate and trace these attacks.

To mitigate the problem, we design Evasive Network Attack Forensic System (ENAFS) to record the HTTP Content-Type mismatch data and automatically analyze the latent evasion behaviors. Our system not only can detect those evasive attacks that disguise as an image or a text file to distribute malware, but also can also trace the attack and analyze the campaign. We've deployed the proposed system in a real-world network, the ISP of CSTNET (China Science and Technology Network). The system processed 1.2 billion HTTP requests per day, found 166 million mismatch data and recorded thousands of latent evasive attack samples.

The main contributions of this paper are as follows:

- A novel network attack traffic forensic system is proposed and implemented to record traffic samples and detect evasive network attacks.
- The proposed system is deployed in the ISP of CSTNET. From 1607 mismatch types recorded by the system, several specific mismatch types that network attacks tend to use to hide their activities have been found.
- Some previously-unknown malware has been detected in the ISP of CSTNET network traffic. These malware activities and samples have been preserved by the system for further analysis.

2 Related work

The first focus of our paper, network forensics, is being researched for decades and many works have been done regarding network forensic frameworks, process models and analysis tools. A good overview, including pointers to much of the literature, appears in the survey paper [9]. It clearly stated the definition, categorization and motivation for network forensics and makes an exhaustive survey of various network forensic frameworks proposed as well as the functionality of various network forensic analysis tools and network security monitoring tools available for forensic examiners. Hviz [10] is proposed to present the event timeline of HTTP and HTTPS activities of a workstation in a visualized way. It employs aggregation, frequent item set mining and cross-correlation between hosts to reduce the number of user browsing events displayed to investigators. Mukkamala and Sung [11] used two artificial intelligence techniques (Artificial Neural Networks (ANNs) and Support Vector Machines (SVMs), to identify significant features for offline network intrusion analysis, and test them with 1999 DARPA intrusion data. Amine Boukhtouta [12] fingerprints maliciousness at the traffic level by using Deep Packet Inspection and IP packet headers classification. Amine Boukhtouta points out that these two approaches are complementary with each other as Deep Packet Inspection is relatively fast while the IP packet headers approach shows excellent accuracy with low rates of false positives and negatives. However, the techniques proposed in these two papers [11,12] are not suitable for real-time network traffic detection. A Network Forensics Readiness and Security Awareness Framework [13] is proposed to collect available logs from connected network devices and filter out anomaly intrusion by applying Support Vectors Machine. A network forensics tool [14] is presented to capture and replay data traffic in Supervisory Control and Data Acquisition systems by developing novel user interface and backend software for a special-purpose network interface card. Kaushik and Joshi [15] propose a network forensic system for ICMP attacks and just focus on some specific attacks on ICMP protocol. This system can only detect known attacks since it is based on rules. Cohen [16] proposes a network forensic framework to reassemble streams, parse HTTP protocol and list the traffic content visually. NetStore [17] is an efficient storage infrastructure for network forensics and monitoring proposed by Giura and Memon, while Vallentin and Paxson designed VAST [18,19], a distributed platform for high-performance network forensics and incident response that provides both continuous ingestion of voluminous event streams and interactive query performance.

The second focus of our paper, HTTP traffic measurement and analysis, has also been the focus of many studies in recent years. Pitfalls in HTTP Traffic Measurements and Analysis [3] quantify the potential error of three HTTP traffic analysis issues: Non-consideration of persistent or pipelined HTTP requests, mismatches between the Content-Type header field and the actual content, and mismatches between the Content-Length header and the actual transmitted volume based on passive traffic measurements of 20.000 European residential broadband customers. AMICO [20] is a novel system for measuring and detecting malware downloads in live web traffic. It learns to distinguish between malware and benign file downloads from the download behavior of the network users themselves, however, it requires a labeled dataset of

past benign and malware file downloads. Nazca [21] monitors network traffic and distinguishes between downloads of legitimate and malicious programs. They found out even during the exploitation and installation phases of the malware's lifecycle, the protocol that overwhelmingly dominates is HTTP.

Our work is based on the real-time traffic analysis and focus on HTTP traffic analysis to detect anomalies. But we further research on the mismatch scenarios and apply it in the proposed ENAFS to detect and trace the latent evasion attacks.

3 Design of system model

We propose a model for network attack forensic, which include *suspicious traffic pre-filtering*, *sample detection* and *attack tracing*. This model of ENAFS is shown in Fig. 1 and explained below.

3.1 Suspicious traffic pre-filtering

The module *suspicious traffic pre-filtering* process network flow and filter out those traffic, which has the phenomenon that the MINE-type of HTTP payload mismatches the declared *Content-type* header. After filtering, our system will record the suspicious sample and metadata into *Sample Attribute Database* for attack tracing.

The process of traffic pre-filtering is shown in Fig. 2. This system only analyzes the packets of *POST* and *GET* method and filter out other's for one reason that other request method is hardly to be seen carrying messages. The other is that *POST* and *GET* cover 97% of HTTP packets in our previous one-month measurement and the results are shown in Fig. 3. HTTP payload zipped with gzip protocol cover an obvious

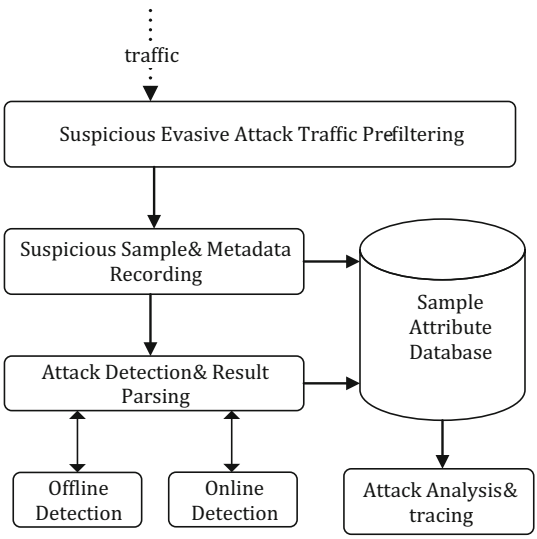


Fig. 1 Network attack forensic system model

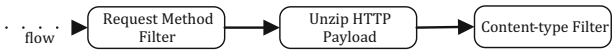
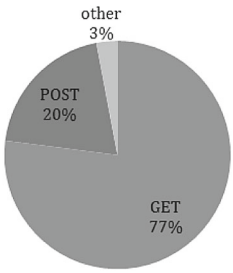


Fig. 2 Process of traffic pre-filtering

Fig. 3 Percentage of HTTP request method



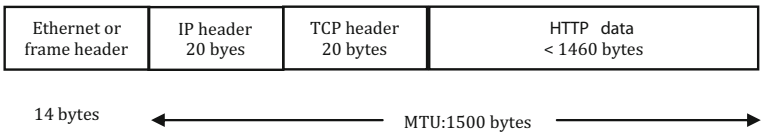


Fig. 4 The length of HTTP payload

portion as a big part of web server prefers to compress their HTTP payload for reducing bandwidth. Then decompress the payload zipped with gzip protocol, and reassemble HTTP payload from network flow by our system. After performing a comparison between HTTP payload MINE-type and the MINE-type declared in Content-type header field. Drop those match HTTP sessions whose payload MINE-type is the same with that one declared in Content-type header field according to the comparison results.

Queue buffer and multi-thread technology are used to reduce I/O overhead and improve I/O efficiency. And, we also propose an optimized algorithm to decrease memory fragmentation. If the length of the payload is unknown, then the buffer size can be expressed by Formula (1), otherwise the buffer size is expressed by Formula (2):

$$\text{BufferSize} = \text{Min}\left\{\text{AllocTimes} \times c, \left\lfloor \frac{\text{MaxSize}}{c} \right\rfloor \times c\right\} \tag{1}$$

And that:

$$\text{BufferSize} = \text{Min}\left\{\text{len}, \ \text{Maxsize}\right\} \tag{2}$$

C, the maximum length of payload in one packet, is set to be 1460, according to Fig. 4. MaxSize is user-defined maximum size of allocated space and AllocTimes refers to the memory request frequency. Len is the length of the HTTP session payload, which include one or more than one packets.

3.2 Sample detection

Module *sample detection* need to automatically detect large amounts of suspicious sample, so a fast and effective detection schema has to be deployed by combining online detection method and offline detection method, considering the advantage of these two methods. Online detection method detects malware by transporting them to online scanning platform and judges whether it is malicious, while offline detection detects malware by deploying antivirus software on the local host. As online detection combines much more scanning engines, it can get a more comprehensive detection result. On the contrary, the detection result decided by how many antivirus software is deployed. Online scanning consumes large network bandwidth and is relatively slow, while offline scanning has a better performance.

To improve the performance and make full use of high configuration of physical host, virtualization [22] and cluster technology are used to deploy multiple antivirus software on one physical host. To improve the offline detection result and representation, some antivirus software is selected to stand for the following types of scanning engine:

- Typical local antivirus software in China.
- Standable open source antivirus software for its flexible deployment.
- Antivirus with heuristic analysis, which has the ability to detect previously unknown computer viruses.
- Widely used and with good performance Commercialize antivirus software.

Hence, the offline detection is used to scan all the suspicious samples that system recorded, but only highly suspicious samples of typical payload MINE-type will be uploaded for online verification as shown in Fig. 5. The detection results are stored into Sample Attribute Database automatically.

The *Attack Tracing* module can analyze and trace specific campaigns by mining *sample attribute database* and analyzing the relations of different attack according to sample metadatas and detection results.

4 System deployment

In this section, we will introduce the deployment of ENAFS according to the system model shown in session 3.

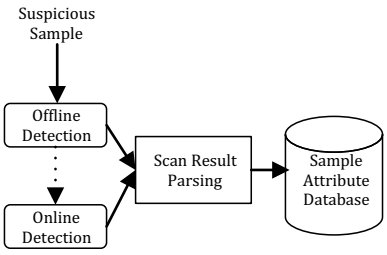


Fig. 5 Process of sample detection

4.1 Pre-filtering process

ENAFS aims at discovering the latent evasive attack using a fake HTTP content-type header. Therefore, we first need to find out the real HTTP payload type and compare it with the HTTP content-type header.

To achieve this goal, HTTP packets are extracted based on PF_RING and use libmagic (file type determination library in GNU/Linux) to identify the HTTP payload MIME type and perform a string comparison between libmagic's results and content-type header.

In most of the cases, the comparison results are correct, but sometimes they are not accurate when meeting the following situations.

- Network transmission errors may cause libmagic fail to detect the HTTP payload header and make it return a general type application/octet-stream.
- Libmagic returns the same MIME type with the Content-Type header, but in different description. With Microsoft Application, for example, libmagic describe it as the type "application/x-dosexec", while Content-Type header describe it as "application/x-msdownload".
- In the Text, Media, and Image classes, libmagic and the Content-Type header agreed on the general category of the type (e.g., image) but disagree on the actual file format (JPEG vs. PNG).

To reduce false positive in comparison, we create a hash map to handle the inaccurate comparison results to make sure that the results are performing as expected.

To reduce resource consumption and improve performance, we filter out some invalid packets or packets we are not interested in. Such as we only concentrate on analyzing packets of *POST* and *GET* method As *POST* and *GET* method are the main request method to be used to transport files or payload. As too short or too long payload may be innocent, those packets, whose length is bigger than 2 GB or smaller than 6byte will be dropped. Response code 404 indicates that the resource does not exist and this packet carries little message, so we drop it too.

After comparison, ENAFS will rerecord the metadata of content mismatched HTTP sessions, including source and destination IP address and port, URI, User-Agent and Server field in the HTTP header. In addition, the complete payload sample will also be reassembled and stored with a hash value for attack tracing and further detection. Analysts can take advantage of the comprehensive and well-organized information in database to trace back the attacks. For example, the URI and IP address can be used to locate the victim and C&C server, User-agent field can tell us victim's device (e.g. Mobile or PC) and OS type. Sample hash can be used to reveal the relationship of the same attack from different sources.

4.2 Sample detection module deploy

Two methods are implemented to detect the suspicious malicious samples. One is offline scanning and the other is online verification.

For offline scanning, five antivirus software, such as 360 AntiVirus, Avira, ClamAV, ESET NOD32, and Kaspersky are selected to be used. 360 AntiVirus is the

Table 1 Information about antivirus used by ENAFS sample detection platform

Antivirus	Signature-based scanning	Heuristic scanning	Open source
Avira Antivirus	Yes	Yes	No
360 Antivirus	Yes	Yes	No
ClamAV Antivirus	Yes	No	Yes
ESET NOD32 Antivirus	Yes	Yes	No
Kaspersky Antivirus	Yes	Yes	No

most famous antivirus in China, so it is the representative free antivirus software in China. ClamAV is the most famous open source antivirus software, and it is choosed to be representative of open source antivirus software. Avira AntiVirus is selected for its heuristics outperform virtually every antivirus. We also pick up two widely used and with well performance commercial Antivirus, ESET NOD32 AntiVirus, and Kaspersky AntiVirus to our detection system. The information of these five antivirus [23] is shown in Table 1. The only open source antivirus, Clamav, does not have heuristic scanning, while the remaining four have both signature-based scanning and heuristic scanning.

As mentioned in Sect. 3.1, an offline antivirus detection cluster based on VMware virtualization is used to detect all suspicious samples and the virtual host cluster diagram is shown in Fig. 6.

According to the statistic of desktop OS market share from StatCounter [24], the operating system (OS) Windows 7 is the most widely used and Windows series OS shares more than 80% during four months from January to April. Take the similarities among Windows series OS into consideration, Windows 7 OS is selected to deploy malware detection platform.

These five antivirus is deployed separately in separate virtual machines, each of them receives suspicious samples from Sample Auto-distribution Module, and executes scanning automatically by the scripts. After scanning, results will be transferred to Scan Result Parsing module. Scan Result Parsing module analyzes scan result files received from antivirus, and then store the scan result and its metadata into the database.

For online verification, we choose VirusTotal, a well-known online scan service which aggregates 55 antivirus products and 61 online scan engines [25] to check for viruses.

Currently the system only uploads all the executable samples through the private API, a higher request rate quota submission way provided by VirusTotal, for online verification.

5 System assessment and tracing

5.1 System assessment

In this section, the performance will be illustrated. It is impossible to evaluate the recall rate of evasive attack detection in a real-world network, as we cannot know the sum of

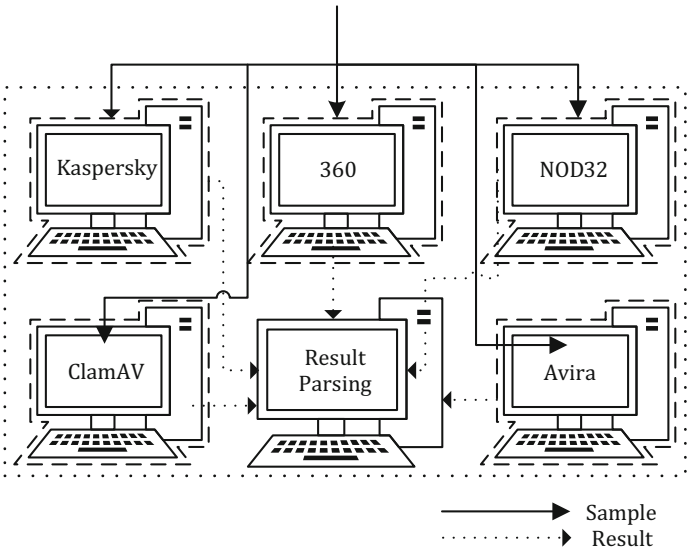


Fig. 6 Virtual host cluster diagram

Table 2 ENAFS offline data set detection results

Malware	Attack type	Content-type, payload-type	Unique samples	Recall rate (%)	Precision rate (%)
Zeus	Botnet	Application/x-dosexec, image/jpg	40	100	100
Bredolab	Botnet	Application/x-dosexec, text/php	10	100	100
Tro-downloader	Evasive trojan downloader	Application/x-dosexec, image/x-icon	120	100	100

packets whose Content-Type is declared in the HTTP header while in fact the actual data is of another different type. So an offline data set from third part is used to verify our detecting system. The VRT Lab [4] has provided some malicious malware traffic contains evasive attack activities which are suitable for use as benchmark of ENAFS. As is presented in Table 2, our system holds a highly reliable result for detecting malicious malware in third-party.

ENAFS was also used in our network board to detect malicious attack activities and test the performance of ENAFS online. We deployed ENAFS on the boarding entry of CERNET network, for 7 days, which serves China's research institution and universities and the bandwidth is 4Gbps. The average resource occupancy is shown in Table 3. The average CPU consumption is 360% and the average memory consumption is 15.36GB.

Table 3 Average resource occupancy of ENAFS

Bandwidth	CPU	Memory
4 Gbps	360%	15.36 GB

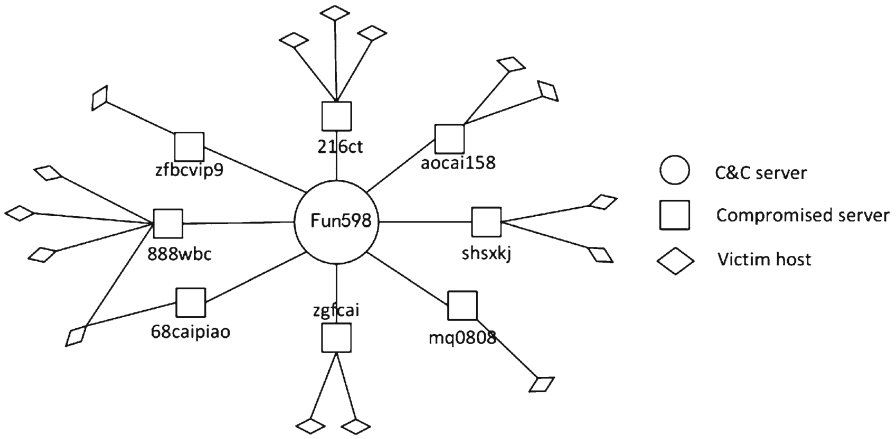


Fig. 7 Evasive network attack tracing graph

As a result, ENAFS processed over 8.4 billion HTTP sessions and found out more than 110 million mismatch instances, covering 1607 different kinds of type mismatches. All these mismatch HTTP metadata and payload samples were stored in the attack tracking database.

After scanning and analyzing these samples, two types of evasive attacks had been found. One is that malware use image headers or text to hide its executable attribute. The other is that malicious scripts, such as webshell, which claims as image to cheat the server-side format verification.

5.2 Attack tracing

Finally, we tried to trace back the origin of a typical evasive attack recorded by the system, which is the important session of detection and characterization of criminal activity [26]. This attack sample was captured on a lottery website. It is an executable file named as*favicon.ico*, which is a common name for a web server, to disguise its real malicious purpose. When victims visit the lottery website, it will trigger drive-by downloads and infect the host. The system also captured other similar instances for *favicon.ico*. There are 15 samples in 8 websites trying to evade detection in such way, and all of them come from illegal lottery sites. Among them, two samples are previously-unseen malware which passed the antivirus scanning. Further analysis showed that all of these malware has relation with the domain *fun598.com*. Figure 7 shows the source tracing graph of the attack generated by ENAFS records.

6 Conclusion and future work

In the paper, we developed ENAFS to discover evasive network attacks on HTTP, which is able to integrally preserve attack samples and their metadata from the live traffic. We use several techniques to reduce the false positive and improve the performance of our system. With the help of sample auto-detection platform integrated in ENAFS, analysts can quickly discover the evasive network attack behind the traffic, analyze its sample and trace back into its source.

The proposed ENAFS was evaluated on the ISP network for 7 days, during which the system handled 8.4 billion HTTP sessions, downloaded and scanned 110 million samples. From the detection results, we found a typical attack and revealed its malicious activity through our system.

Currently we are analyzing samples that the system has collected, expecting to find more HTTP Evasive behavior and add these features into our system. Our platform works on HTTP protocol layer. Data transported via HTTPS protocol are encrypted using TLS/SSL (HTTPS means HTTP over SSL/TLS), so the platform cannot work if it processes the raw encrypted traffic. However, the attack forensic platform is suitable for HTTPS in the scenario that we can decrypt the HTTPS into HTTP using a proxy gateway, or man-in-the-middle(MITM) attack. Our method is scalable on the HTTP network behaviors. In the future, we would like to extend our system to support more protocols such as FTP, SMTP, POP3.

Acknowledgements Thanks to the collaboration from VirusTotal and the private API, which improve our performance and shorten the online detection time. This work is supported by the National Science and Technology Support Program (No. 2012BAH46B02) and the Strategic Priority Research Program of the Chinese Academy of Sciences (No. XDA06030200).

References

1. Urias V, Young J, Hatcher S (2010) Implications of cloud computing on digital forensics. GSTF Int J Comput 1(1):131–137
2. Stiawan D, Idris MY, Abdullah AH (2015) Penetration testing and network auditing: Linux. J Inf Process Syst (JIPS) 11:104–115
3. Schneider F, Ager B, Maier G, Feldmann A, Uhlig S (2012) Pitfalls in HTTP traffic measurements and analysis. Passiv Act Meas 7192:242–251
4. VRT Labs (2014) Content-type mismatch. https://labs.snort.org/papers/contentmi-smatch.html
5. FireEye (2015) APT30 and the mechanics of a long-running cyber espionage operation. https://www2.fireeye.com/rs/fireye/images/rpt-apt30.pdf
6. Operation Poisoned Helmand (2014) https://www.threatconnect.com/operation-poisoned-helmand/
7. Snort (2016). http://www.snort.org/
8. Khamphakdee N, Benjamas N, Saiyod S (2014) Improving intrusion detection system based on snort rules for network probe attack detection. In: 2nd International Conference on Information and Communication Technology (ICoICT), pp 69–74
9. Pilli ES, Joshi RC, Niyogi R (2010) Network forensic frameworks: survey and research challenges. Digit Investig 7(1/2):14–27
10. Gugelmann D, Gasser F, Ager B, Lenders V (2015) Hviz: Http(s) traffic aggregation and visualization for network forensics. Digit Investig 12(Suppl 1):1–11
11. Mukkamala S, Sung AH (2003) Identifying significant features for network forensic analysis using artificial intelligence techniques. Int J Digit Evid IJDE 1(4):1–17

12. Boukhtouta A, Mokhov S-A, Lakhdari N-E, Debbabi M, Paquet J (2016) Network malware classification comparison using dpi and flow packet headers. Comput Virol Hacking Tech 12:69–100
13. Al-Mahrouqi A, Abdalla S, Kechad T (2015) Efficiency of network event logs as admissible digital evidence. In: Science and Information Conference
14. Parry J, Hunter D, Radke K, Fidge C (2016) A network forensics tool for precise data packet capture and replay in cyber-physical systems. In: Australasian Computer Science Week Multiconference (ACSC2016)
15. Kaushik AK, Joshi RC (2010) Network forensic system for ICMP attacks. Int J Comput Appl 2(3):14–21
16. Cohen M (2008) PyFlag—an advanced network forensic framework. Digit Investig 5:112–120
17. Giura P, Memon N (2010) NetStore: an efficient storage infrastructure for network forensics and monitoring. In: Proceedings of the 13th International Conference on Recent Advances in Intrusion Detection, Ottawa, Ontario, Canada, pp 15–17
18. Vallentin M, Paxson V, Sommer R (2016) VAST: a unified platform for interactive network forensics. In: 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16), pp 345–362
19. Vallentin M, Charousset D, Schmidt TC, Paxson V, Wahlisch M (2014) Native actors: how to scale network forensics. In: Proc. of ACM SIGCOMM, Demo Session, New York, August 2014
20. Vadrevu P, Rahbarinia B, Perdisci R, Li K, Antonakakis M (2013) Measuring and detecting malware downloads in live network traffic. ESORICS'13, pp 556–573
21. Invernizzi L, Miskovic S, Torres R, Saha S, Lee S-J, Kruegel C, Vigna G (2014) Nazca: detecting malware distribution in large-scale networks. In: Proceedings of the ISOC Network and Distributed System Security Symposium
22. Huh J-H, Seo K (2016) Design and test bed experiments of server operation system using virtualization technology. Human-centric computing and information sciences (HCIS). Springer, Berlin, pp 1–21
23. Comparison of antivirus software (2016) https://en.wikipedia.org/wiki/Compari-son_of_antivirus_software
24. StatCounter. http://gs.statcounter.com/#desktop-os-ww-monthly-201604-201604-bar
25. VirusTotal (2015) https://en.wikipedia.org/wiki/Virus-Total
26. Shahabi C, Kim S-H, Nocera L, Constantinou G, Lu Y, Cai Y-H, Medioni G, Nevatia R, Banaei-Kashani F (2014) Janus-multi source event detection and collection system for effective surveillance of criminal activity. J Inf Process Syst (JIPS) 10(1):1–22