Rev. Roum. Sci. Techn.– Électrotechn. et Énerg.
Vol. **64**, *4*, pp. 429–434, Bucarest, 2019

NETWORK FORENSIC INVESTIGATIONS OF TUNNELED TRAFFIC: A CASE STUDY

JAN PLUSKAL[1], MICHAL KOUTENSKÝ[1], MARTIN VONDRÁČEK[1], ONDŘEJ RYŠAVÝ[1]

Key words: Network traffic forensics, Generic stream encapsulation, Network forensic and analysis tool.

The increasing importance of network forensics in the investigations conducted by Law Enforcement Agencies is indisputable. Today's Internet does not carry ordinary TCP/IP traffic but utilizes many other encapsulations and tunneling protocols. In this paper, we overview the most used tunneling protocols and their features concerning digital forensic analysis. A case study of generic stream encapsulation describes how the investigator can obtain encapsulated application data from within.

1. INTRODUCTION

Internet applications use different communication protocols to exchange content. Most of network forensic analysis tools can correctly identify the communicating application and extract the content communication if encryption is not used. However, encryption is not the only obstacle for network forensic tools. Application communication can be also encapsulated in other protocols that provide an extra network layer in addition to the Internet's TCP/IP stack. These tunneling protocols are supposed to protect the encapsulated communication. It may be because the carried protocol is not compatible with the transport network technology or the additional security is necessary. Tunneling protocols are the basis for building virtual private networks. The local traffic needs to be sent over the Internet, which opens various possibilities for attackers. By using tunneling protocols, it is possible to protect the encapsulated communication with strong encryption to avoid eavesdropping and communication alteration. However, this benefit of network security represents a challenge for network forensics.

This publication extends the original paper "Network forensics in generic stream encapsulation (GSE) overlay networks" published in In 6th Conference on the Engineering of Computer Based Systems (ECBS '19), September 2–3, 2019, Bucharest, Romania [1].

1.1. PROBLEM DESCRIPTION

Network data acquisition faces many challenges. One of the complications for evidence recovery from captured network data is the use of encryption and tunneling. When end-to-end encryption was used the content of messages is protected but it is still possible to identify individual connections. In the case of tunneling protocols, multiple connections are multiplexed in the tunnel. The original design goal of tunneling protocols was to interconnect networks through possible incompatible network technology. The captured content of the tunnel can be easily extracted, and individual connections recovered. However, modern tunneling protocols include security measures by applying encryption to transferred content. Therefore, connections can only be recovered at exit points of the tunnel.

1.2. CONTRIBUTION AND PAPER STRUCTURE

The present paper provides an overview the common points in the network topology that can be used by law enforcement agencies (LEA) to conduct lawful interception. We provide a summary of most used tunneling protocols and discuss their features with respect to digital forensic analysis. For each protocol, the possibility of content extraction is explained. Also, a brief overview of methods for the classification of encapsulated traffic is provided. The issue of connection recovery from tunneled communication is demonstrated using the GSE protocol as an example.

2. LAWFUL INTERCEPTION POINTS IN NETWORK TOPOLOGY

The goal of lawful network data acquisition is to collect enough information for evidence extraction. As most of the Internet traffic is encrypted, the analysis of metadata represents the most important approach. There are many possible locations in the network topology that may be used for lawful interception and their selection depends on various circumstances. This section describes the locations and adequate techniques used to collect digital evidence out of network devices.

The end-user machine is the place where any kind of data is presented to the user, or stored. If encryption is used to protect data in transfer, this is the place where it happens. If the device can be accessed by an investigator, several techniques for obtaining the evidence via the installation of agent software that can intercept API hooks [2], capture network traffic [3], capture screen [4] or maliciously modify [5].

Internet service provider (ISP) The most typical lawful interception probe deployment occurs in the ISP network [6]. The LEA possessing a search warrant can [7] compel the ISP to reveal the retained data [6] or to intercept the suspect's communication [8] using LEA's deployed network probes [6]. Technically, there are several types of interception or traffic manipulation that can be done.

Network layer defines a physical connection between devices connected to a shared segment identified by MAC addresses that are resolved by ARP protocol. ARP can be misused to redirect the communication to an interception device [9], but it can also be error-prone [10]. The common encapsulation and tunneling protocols are VLAN, L2TP described in Section 3.

Internet layer The majority of traffic interception probes assume that traffic is redirected into them. Interception rules that are typically based on the IP address of the

[1] Brno University of Technology, Faculty of Information Technology, Božetěchova 2, 61266 Brno, Czech Republic, E-mails: {ipluskal, koutenmi, ivondracek, rysavy}@fit.vutbr.cz.

suspect, defines which IP flows should be intercepted, *i.e.*, captured for future analysis. Interception up to 1 Gbps speeds can be done on regular PCs without additional configuration. Speeds up to 10 Gbps require that data are not copied between the kernel and userspace, *e.g.*, usage pf_ring [11], or n2disk [11]. Speeds past the 10 Gbps [12] requires custom kernel optimizations, *e.g.*, pf_ring and CPU core to NIC queue mapping. Typical encapsulations are IPsec, PPTP, IPIP, 6in4 described in Section 3.

Transport & application layer On the transport layer, we may utilize "policy-based-routing" to define rules that describe communication we are interested in to capture, or redirect to capturing probe. Typical encapsulation protocols are GRE, SSTP, Ayiya described in Section 3. On the application layer, we can go deeper and manipulate communication, *e.g.*, conduct SSL/TLS inspection, filtering, or capturing [13, 14].

Datacenter accommodates the complexity of network architecture to their size [15]. Smaller providers [16] use from common network design segmenting a network into smaller subsets on Internet layer. mid to large providers [17, 18], and cloud providers commonly use software defined networking (SDN) [19] to create virtual networks over well-designed base network layer. Customers can usually define network typologies dynamically as they create their visualized infrastructures [18]. All protocols described in Section 3 can be used.

3. ENCAPSULATION AND TUNNELING PROTOCOLS

The structure of the TCP/IP networking stack used on the Internet is quite rigid. There is a fixed number of layers, each providing certain functionality. This setup works fine for common scenarios, but occasionally the need to use a different configuration arises.

Encapsulation is a core concept for computer networks and is the basis for the layer model. As data moves downwards through the stack, from application to the physical medium, the contents get wrapped–encapsulated– at each layer in additional protocol information. When processing received data, each layer interprets its own information and forwards the encapsulated payload to the layer above.

Tunneling and encapsulation are likewise strongly related concepts. While common protocols encapsulate data of higher layer protocols, tunneling protocols may also encapsulate data of protocols of the same or lower layers.

Table 1

Summary of tunneling protocol features

Protocol	Authentication	Encryption
IPSec	Built-in	Built-in
GRE	No	No
PPTP	Using PPP	Using PPP
L2TP	Using PPP	Using PPP
SSTP	Using SSL	Using SSL
IPIP	No	No
6in4	No	No
Ayiya	No	No

This effectively allows extending the stack, repeating some layers multiple times, and can be considered a form of recursion.

Common use-cases for tunneling include transporting data over network segments with an unsupported network or data-link layer protocols or providing the illusion of being connected to a remote LAN via VPN.

3.1. COMMON TUNNELING PROTOCOLS

There exist a number of tunneling protocols varying in their application and scope. Some have very narrowly defined capabilities while others attempt to be general and extensible, see Table 1 for comparison.

IPsec is a suite of protocols that work with the IP family to provide confidentiality and integrity of transmitted data [20]. While not strictly a tunneling protocol, it can operate in a tunneling mode where the secured IP packet is encapsulated in a new packet. The operation of IPsec roughly consists of three components: security association (SA), authentication header (AH) [21] and encapsulating security payload (ESP) [22]. When a party is interested in communicating securely, it negotiates a SA which holds the necessary cryptographic parameters. Afterward, the communicating parties can include AH in their packets, which can be used to verify the integrity of the received data. AH achieves this by computing a hash from the fields of the IP header as well as the included payload and the SA. It is the last property that differentiates AH from a basic checksum and protects the data from being modified in transit. As AH protects parts of the IP header in addition to the payload, it also provides a form of authentication. The ESP can provide integrity as well as confidentiality using encryption. In transport mode, ESP only encrypts the payload; in the aforementioned tunneling mode, ESP encrypts both the IP header and the payload and encapsulates them in a new IP header.

GRE is an encapsulation protocol developed by Cisco to allow for encapsulation of link and network protocols in a generic way [23]. The protocol itself is very simple and provides no security features such as encryption or authentication. The payload packet is encapsulated in the GRE header, which is then encapsulated in the delivery protocol. The GRE header contains a protocol number identifying the encapsulated protocol. Additionally, a checksum might be present. Earlier RFCs included several other fields that specified, *e.g.*, the number of allowed recursions of encapsulation or routing information [23]. Their use has been deprecated [24].

PPTP is a tunneling protocol originally designed to carry PPP traffic over IP networks [25]. It operates on the link layer and uses a client/server model, where the server is called the PPTP network server and the client PPTP access concentrator. For encapsulation of the payload, PPTP uses an enhanced version of GRE. Each PAC-PNS pair establishes a tunnel and a control connection which runs over TCP. This control connection is used to manage both the tunnel and any user sessions using it. PPTP uses security mechanisms from PPP for authentication and encryption; the most commonly known are Password Authentication Protocol and Challenge-Handshake Authentication Protocol.

L2TP aims to tunnel PPP packets in a way that is as transparent as possible [26]. It decouples the layer 2 and PPP endpoints, allowing them to exist at different devices connected by a packet-switched network. The overall design is reminiscent of that of PPTP. The two endpoints are called the L2TP Network Server and L2TP Access Concentrator, filling similar roles as their PPTP equivalents. These two endpoints establish a tunnel which

consists of a control connection and zero or more sessions. The control channel is reliable, while the channel used for transmitting data messages is not. In IP networks, the transport protocol to carry the L2TP messages is UDP, which avoids the issues brought by stacking several TCP connections on top of each other. L2TP supports the CHAP-like tunnel authentication mechanism but provides no integrity or confidentiality, leveraging features provided by PPP instead. However, it is commonly used in combination with IPsec to encrypt the payload via ESP and/or AH.

SSTP tunnels PPP frames over SSL/TLS, using TCP as its transport protocol [27]. In this case, security is provided by SSL using encryption and integrity checking. The structure of the SSTP header is quite simple, with the only interesting field being the C flag. When set, the encapsulated payload contains an SSTP control message; otherwise the higher-layer protocol.

IPIP is a protocol meant to alter the normal routing process by encapsulating the IPv4 packet in another IPv4 packet and sending it to an intermediate node [28]. The entry point of the tunnel wraps the IPv4 packet in another IPv4 header destined to the tunnel endpoint. After traversing the tunnel, the inner IPv4 packet is decapsulated and processed normally, routed and forwarded to its true destination. The protocol is simple, using no additional headers, as it is limited to one type of outer-inner protocol; most of the complexity lies in rules on how to properly handle ICMP messages. On its own, it provides no additional security features over basic IPv4.

6in4 is a transition mechanism allowing IPv6 traffic to traverse networks with only IPv4 support [29]. A tunnel is established between two devices, and the IPv6 traffic is transported by encapsulating it in IPv4. A special IP protocol number is defined for this purpose. 6in4 itself provides no security-related features such as authentication or integrity.

Ayiya attempts to solve some of the issues that transition protocols such as 6in4 have with establishing tunnels that travel through NATs. [30] These NATs need to be manually reconfigured to properly handle 6in4, which in some cases is not possible. Ayiya solves this by tunneling IP traffic not directly over IP, as is the case with 6in4 or IPIP, but over a transport layer protocol such as TCP or UDP. It aims to be general, independent of both the payload protocol and the transport protocol being used, thus the name Anything in Anything. It is even possible to tunnel the payload protocol directly over the network protocol, in the vein of IPIP, for IP over IP tunnels with minimal overhead where possible. Ayiya defines a custom header that is placed between the payload and the delivery protocol. The header contains an identity field to help determine which sender the packet has originated from, as the source port number and IP address may change arbitrarily during the connection, due to NAT, DHCP, IPv6 privacy extensions *etc*. An operation code field may specify special handling of the received packet, such as echoing it back to the sender. In addition, it contains an optional signature and authentication, providing some security features out of the box. A heartbeat message is used to keep the tunnel open, as not receiving any packets for a certain period of time results in closing the tunnel.

3.2. IDENTIFICATION OF ENCAPSULATION PROTOCOLS

To properly parse a protocol and extract information from it, it is necessary to correctly identify it. As there is no field identifying the application protocol in common transport protocol headers (TCP, UDP), and port numbers alone aren't sufficient to identify the protocol being used, several approaches have been developed to solve this issue.

Deep packet inspection (DPI) is a content-based method that attempts to identify protocols by looking inside the payload [31]. It looks for known signatures in the transmitted data to identify the data flow as a particular protocol. The signature matching process can be as simple as looking for a value in the first few bytes of the payload (application header) or complex heuristics requiring access to whole flows. DPI achieves high accuracy; the chief downside of this approach is that it needs to be able to access the data being transmitted to function properly, and it needs to inspect every packet passing through the interface. If the application uses encryption, DPI fails to provide meaningful results.

Connection patterns can be used to classify traffic into categories without inspecting the payload [32]. Sequences of flows are matched against heuristics using a set of rules. As different types of traffic (such as web or P2P) display different connection patterns over time, this information is sufficient to categorize the observed flows. While significantly simpler and less computationally intensive than DPI, this method only achieves rough categorization; it does not identify specific protocols.

Statistical methods are based on flow properties such as duration, packet size or arrival times [33]. Measurements of various protocol attributes are taken, and these are compared to existing models. It is possible to include some DPI attributes and treat them as statistical properties, resulting in a hybrid approach. Creating models by extracting fingerprints can be done manually; however, this is a very time-consuming process. Available algorithms, therefore, try to automate the process of creating new protocol models, requiring only pre-classified training data instead, utilizing machine learning [34].

As tunneling protocols work above the network or transport layer, these approaches can be used to detect encapsulated traffic as well. Few of the protocols described in the previous section provide encryption by themselves, and most offer some kind of signature available in the header that can be matched. Moreover, the accuracy of identification is of high priority, as we don't want to simply categorize the traffic to gather statistics but identify the encapsulated traffic as well. For this we need to correctly identify the protocol being used; DPI, therefore, appears to be a reasonable choice for encapsulation identification. The problem is further complicated by the possibility of IPsec being used to secure the tunneled traffic independent of the protocol being used; this is, in fact, the recommended approach by L2TP [26]. Additionally, tunneling protocols can tunnel other tunneling protocols, recursively extending the number of layers; to properly extract the application data, it is necessary to identify and decapsulate each of those protocols in turn properly.

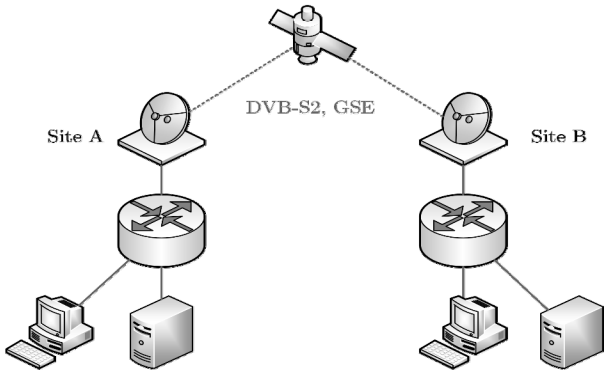

Fig. 1 – This example scenario is presenting a professional application of DVB-S2 and GSE. This architecture offers point-to-point or point-to-multipoint connections over a satellite link in both directions. Traffic between site A and site B is carried using generic stream encapsulation. The figure is based on the GSE implementation guidelines [37].

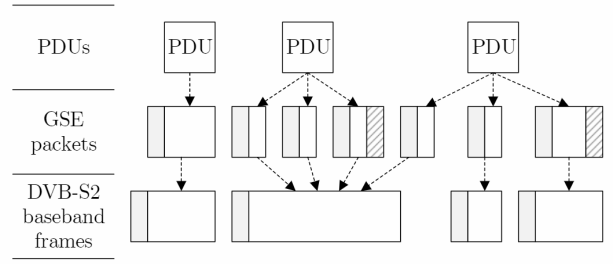

Fig. 2 – The figure shows the encapsulation of network layer PDUs into GSE packets and transmission of GSE packets inside physical layer baseband frames. GSE packets and baseband frames consist of a header (shown as a grey block) and a data field (shown as white space). GSE packet carrying the last fragment also contains CRC-32 (shown as a block with pattern). The figure is based on GSE protocol specification [35, p. 10].

4. GENERIC STREAM ENCAPSULATION (GSE) CASE STUDY

Network protocol generic stream encapsulation (GSE) was defined by the digital video broadcasting project (DVB), and it offers a way to transport IP traffic over a generic physical layer, usually over DVB physical infrastructure [35, p. 6]. GSE, as a native IP encapsulation protocol on DVB bearers, was introduced with the second-generation satellite transmission system called DVB-S2 (Fig. 1). Generic data transmission on the first generation of DVB standards was formerly possible using the multi-protocol encapsulation (MPE) on MPEGTS packets. However, MPE suffered significant overhead. GSE is also included in the Satlabs System Recommendations for DVB-RCS terminals [36].

Outline of GSE procedures operation of GSE allows transmission of variable size generic data encapsulated into baseband frames. GSE can encapsulate not only IPv4 traffic but a wide range of other protocols including IPv6, Ethernet, ATM, MPEG, and others. It supports addressing using 6-Byte MAC addresses, 3-Byte addresses, and even a MAC address-less mode [35, p. 6]. Encapsulation and decapsulation procedures performed by the DVB broadcast bearers are transparent to the rest of the network topology and the carried traffic. Shall a network layer PDU be transmitted over a satellite connection, GSE packets serve as a data link layer (Fig. 1).

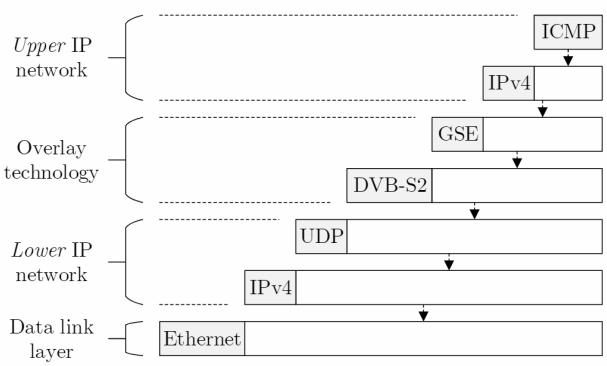

Fig. 3 – Example of IP traffic encapsulated in the GSE layer, which is carried by another IP traffic. The resulting virtual topology can be characterized as an established overlay network.

This GSE layer provides encapsulation, fragmentation, and slicing. Created GSE packets are then carried in baseband frames, *e.g.*, DVB-S2, on the physical layer (Fig. 2). The receiving side performs a reassembly process, integrity check, and a final decapsulation of transmitted PDUs [38].

Moreover, it is also possible to transport GSE packets over, for example, standard IP network infrastructure. In this case, the DVB-S2 traffic can be carried like a generic payload on the application layer with the use of the UDP as a transport layer. Therefore, given UDP datagrams carry DVB-S2 baseband frames, which further carry GSE packets encapsulating selected protocol communication. This approach effectively establishes an overlay network infrastructure, because IP traffic can practically carry GSE packets, which can carry another layer of IP traffic. At this point, the UDP/IP layer below GSE can be considered the carrier *(encapsulating) traffic* whereas, for example, the IP layer above GSE can be described as the carried (*encapsulated*) traffic. This approach is presented in Fig. 3.

According to specifications and recommendations published by SatLabs, the implementation of a receiver with an Ethernet interface can be divided into a demodulation/decoding device, and a device focused on baseband processing. In such a case, the *L3 Mode Adaptation Receiver Header* can be prepended to received data [39, p. 10]. The receiving device would then process *DVB-S2 L3 Mode Adaptation Receiver Header*, *DVBS2 baseband frame*, and *GSE packets* to analyze transmitted communication.

Fragmentation, slicing, padding and reassembly process As noted earlier, GSE procedures can encapsulate different protocol data units in one or more GSE packets. In general, GSE packets have variable lengths, and they can be sent in different baseband frames individually or in a group. Therefore, fragmentation, slicing, padding and reassembling can occur. In this context, fragmentation refers to a situation when a PDU and extension header is fragmented into multiple GSE packets (Fig. 2). Slicing indicates a case when a GSE packet itself is divided into several contiguous baseband frames [35, p. 8]. Noted slicing, therefore, refers to physical layer fragmentation, which shall be transparent to the GSE layer [37, p. 27]. Concerning DVB-S2 applications, GSE slicing does not occur [37, p. 31].

Shall a single PDU be fragmented into several GSE packets, each packet is assigned a *fragmentation identifier (Frag ID)* label in the GSE header [35, p. 17]. Frag ID is used to match fragments belonging to the same original

PDU. This approach enables the simultaneous transmission of fragments from up to 256 different original PDUs. GSE packets carrying a complete PDU and GSE packets with PDU fragments can be distinguished using start and end flags in the GSE header. The protocol of carried PDU is indicated by protocol type/extension field in the GSE header of the first fragmented packet and every not fragmented packet. The packet with the last PDU fragment further carries a CRC-32 field used to check integrity after the reassembly process (Fig. 2). It is important to note that for example, DVB-S2 allows multiplexed transmission of multiple streams, each identified by its *input stream identifier (ISI)* [37, p. 32] in baseband header [40, p. 20]. The reassembly process has to be carried out independently for each received stream [35, p. 21]. Some of the possible GSE packet formats are presented in the technical specification [35, pp. 31–32].

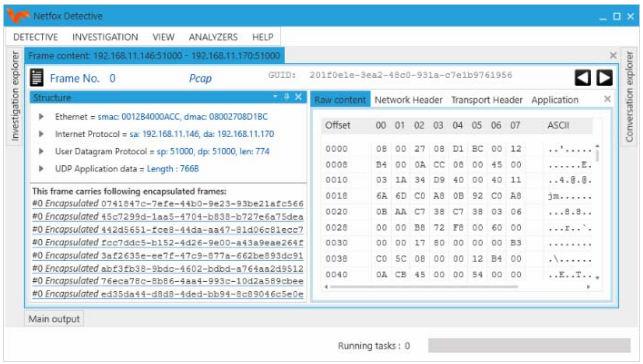

Fig. 4 – View of the frame content of the Netfox Detective presenting a frame carrying eight other encapsulated frames. It is possible to navigate between encapsulated frames using shown links labeled with GUID of the target frame.

Concerning GSE addressing modes noted earlier, an additional fourth mode called *label re-use* can be used when multiple GSE packets are carried in a single baseband frame. Shall label re-use be indicated, current GSE packet without address belongs to the same address as the last previously processed GSE packet. A more detailed analysis of GSE protocol is beyond this paper's scope. GSE packet format is defined in the protocol specification [35, p. 12]. Further information can be found in standards, recommendations, and guidelines covering GSE and DVB-S2 [35, 41, 42, 37, 43].

4.1. EVALUATION

Every layer of decapsulated traffic is subject to further network forensic analysis performed by the Netfox Detective[1]. The information is presented in the GUI. The view informs the user whether the current frame in encapsulated or not. It is also possible to navigate between views showing individual encapsulating frames (Fig. 4) and encapsulated frames. The implementation has been evaluated on publicly available dataset [2], and results (amount of correctly identified and extracted GSE communications) were comparable to the reference Wireshark implementation. A set of integration tests was implemented that verify the correct processing of GSE traffic in future releases and prohibit regression bugs from being introduced.

[1] https://github.com/nesfit/NetfoxDetective
[2] https://wiki.wireshark.org/DVB-S2 (last accessed 2019-12-12).

5. CONCLUSIONS

Network forensic analysis currently faces many challengesthat stems from the fact that most of the Internet traffic is encrypted. Thus, the analysis relies on the metadata of messages and the behavioral characteristics of the communication. In this paper, we have considered another issue for network forensics, namely, the use of tunneling protocols. We have identified the problem that tunneling represents for evidence extraction. Then we have presented an overview of different existing tunneling protocols and their characteristics with respect to digital forensics. Finally, we have demonstrated the case study using the GSE protocol, which allows transporting IP traffic via satellite connections. The experimental GSE protocol analyzer implements the method for full content extraction. Thus it can be used to preprocess the data for network forensic analysis tools that are unable to directly cope with tunneled communication. If tunneling protocols apply encryption to protect the encapsulated traffic, the content extraction is not possible in general. However, several approaches were proposed for the detection of the application class of encapsulated communication. The paper provides a brief overview. Their adaptation for different tunneling protocols belongs to the intentions of our future work.

Received on December 1, 2019

REFERENCES

1. J. Pluskal, M. Vondráček, O. Ryšavy, *Network ` Forensics in GSE Overlay Networks*, In: Proceedings of the 6th Conference on the Engineering of Computer Based Systems, ACM, 2019.
2. V. Lifliand, A.M. Ben-Menahem, *Encrypted network traffic interception and inspection*, US Patent 8,578,486, 2013.
3. L. Temoshenko, H. Nhan, C. M Parker, R. L King, J. D. Douglas, N. A Mitchell, G. R Everhart, D. W. Currie, *System and method for intercepting packets in a pipeline network processor*, US Patent 7,046,663., 2006.
4. S. Belikovetsky, O. HaCohen, N. Lauderdale, *Deception using screen capture*, US Patent 10,425,445, 2019.
5. D. Javaheri, M. Hosseinzadeh, A. M. Rahmani, *Detection and Elimination of Spyware and Ransomware by Intercepting Kernel-Level System Routines*, IEEE Access 6 , 78321– 78332 (2018).
6. A. Imbimbo, F. Attanasio, *Apparatuses, methods, and computer program products for data retention and lawful intercept for law enforcement agencies*, US Patent 9, 204,293, 2015.
7. M. Ponsford, *The Lawful Access Fallacy: Voluntary Warrantless Disclosures, Customer Privacy, and Government Requests for Subscriber Information*, Canadian Journal of Law & Technology **15**, *1* (2017).
8. A. Muñoz, M. Urueña Pascual, R. Aparicio Morenilla, G. Rodríguez de los Santos López, *Digital Wiretap Warrant: Improving the security of ETSI Lawful Interception* (2015).
9. A. Foss, *Method for automatic traffic interception*, US Patent 7,567,573, 2009.
10. P. Branch, Ana Pavlicic, G. Armitage, *Using MAC addresses in the lawful interception of IP traffic*, In: Proc Australian Telecommunications Networks & Applications Conference (ATNAC), pp. 9– 11, 2004.
11. L. Deri *et al.*, *Improving passive packet capture: Beyond device polling*, In: Proceedings of SANE, 2004. Amsterdam, Netherlands, pp. 85–93, 2004.
12. L. Deri, A. Cardigliano, F. Fusco, *10 Gbit line rate packet-to-disk using n2disk*, In: 2013 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp. 441–446, 2013.
13. S. Aryan, H. Aryan, J A. Halderman, *Internet censorship in Iran: A first look, Presented as part of the 3rd {USENIX}*, In: Workshop on Free and Open Communications on the Internet, 2013.
14. P. Winter, T. Pulls, J. Fuss, *ScrambleSuit: A Polymorph Network Protocol to Circumvent Censorship*. arXiv preprint arXiv:1305.3199 (2013).

15. D. Spiekermann, J. Keller, T. Eggendorfer, *Improving Lawful Interception in Virtual Datacenters*, In: Proceedings of the Central European Cybersecurity Conference, 2018.
16. L. Reith, *Method and system for lawful interception of packet switched network services*, US Patent 7,447,909, 2008.
17. J. Cartmell, A. Chandra, P. R Chitrapu, *Lawful interception for local selected IP traffic offload and local IP access performed at a non-core gateway*, US Patent 9,591,031, 2017.
18. G. Amato, *Lawful intercept management modules and methods for LI-configuration of an internal interception function in a cloud based network*, US Patent 9,866,435, 2018.
19. N. McKeown, *Software-defined networking*, INFOCOM keynote talk **17**, *2*, pp. 30–32 (2009).
20. Karen Seo, S. Kent, *Security Architecture for the Internet Protocol*, RFC 4301, 2005.
21. S. Kent, *IP Authentication Header*, RFC 4302, 2005.
22. S. Kent, *IP Encapsulating Security Payload (ESP)*, RFC 4303, 2005.
23. D. Farinacci, S.P. Hanks, T. Li, P.S. Traina. *1994. Generic Routing Encapsulation (GRE)*, RFC 1701.
24. T. Li, D. Farinacci, S.P. Hanks, D. Meyer, P.S. Traina, *Generic Routing Encapsulation (GRE)*, RFC 2784, 2000.
25. G. Zorn, G.-S. Pall, K. Hamzeh, *Point-toPoint Tunneling Protocol (PPTP)*, RFC 2637, 1999.
26. A.J. Valencia, G. Zorn, W. Palter, G.-S. Pall, M. Townsley, A. Rubens *Layer Two Tunneling Protocol "L2TP"*, RFC 2661, 1999.
27. Microsoft Corporation, *Secure Socket Tunneling Protocol (SSTP)* (18 ed.), 2018.
28. C. E. Perkins, *IP Encapsulation within IP*, RFC 2003, 1996..
29. R. E. Gilligan, E. Nordmark, *Basic Transition Mechanisms for IPv6 Hosts and Routers*, RFC 4213, 2005.
30. J. Massar, *AYIYA: Anything In Anything. Internet-Draft draft-massar-v6ops-ayiya-02. Internet Engineering Task Force*, https://datatracker.ietf.org/doc/html/draft-massar-v6ops-ayiya-02 Work in Progress, 2004.
31. A.W. Moore, K. Papagiannaki, *Toward the accurate identification of network applications*, In: Lecture Notes in Computer Science, **3431**, pp. 41–54, 2005.
32. W. John, S, Tafvelin, *Heuristics to classify Internet backbone traffic based on connection patterns*. 2008 International Conference on Information Networking, ICOIN (2008), pp. 1–5.
33. E. Hjelmvik, W. John, *Statistical protocol identification with spid: Preliminary results*, In: Swedish National Computer Networking Workshop, pp. 399–410, 2009.
34. S. Zander, T. Nguyen, G. Armitage, *Automated traffic classification and application identification using machine learning*. In: Proceedings - Conference on Local Computer Networks, LCN 2005, pp. 250–257, 2005.
35. ETSI, *ETSI TS 102 606-1 V1.2.1 - Digital Video Broadcasting (DVB); Generic Stream Encapsulation (GSE); Part 1: Protocol*, European Telecommunications Standards Institute, 2014.
36. SatLabs Group, *SatLabs System Recommendations Version 2.1.2*, 2010.
37. ETSI, *ETSI TS 102 771 V1.2.1 - Digital Video Broadcasting (DVB); Generic Stream Encapsulation (GSE) implementation guidelines*, European Telecommunications Standards Institute, 2011.
38. DVB Project Office, *DVB-GSE - Generic Stream Encapsulation. DVB Project Office*, https://www.dvb.org/resources/public/factsheets/dvb-gse_factsheet.pdf, 2015.
39. SatLabs Group, *Mode Adaptation Input and Output Interfaces for DVB-S2 equipment Version 1.3*, 2008.
40. ETSI, *ETSI EN 302 307-1 V1.4.1 - Digital Video Broadcasting (DVB); Second generation framing structure, channel coding and modulation systems for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications. Part 1: DVB-S2*, European Telecommunications Standards Institute, 2014.
41. ETSI, *ETSI TS 102 606-2 V1.1.1 - Digital Video Broadcasting (DVB); Generic Stream Encapsulation (GSE); Part 2: Logical Link Control (LLC)*, European Telecommunications Standards Institute, 2014.
42. ETSI, *ETSI TS 102 606-3 V1.1.1 - Digital Video Broadcasting (DVB); Generic Stream Encapsulation (GSE); Part 3: Robust Header Compression (ROHC) for IP*, European Telecommunications Standards Institute, 2014.
43. ETSI, *ETSI TR 102 376-1 V1.2.1 - Digital Video Broadcasting (DVB); Implementation guidelines for the second generation system for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications; Part 1: DVB-S2; Part 1: DVB-S2*, European Telecommunications Standards Institute, 2015.