Monitoring of Tunneled IPv6 Traffic Using Packet Decapsulation and IPFIX

Martin Elich[1], Matěj Grégr[2] and Pavel Čeleda[1]

[1] CESNET, z.s.p.o., Zikova 4, 160 00 Prague, Czech Republic
martin.elich@gmail.com, celeda@liberouter.org
[2] Brno University of Technology, FIT, Božetěchova 2, 612 66 Brno, Czech Republic
igregr@fit.vutbr.cz

Abstract. IPv6 is being deployed but many Internet Service Providers have not implemented its support yet. Most of the end users have IPv6 ready computers but their network doesn't support native IPv6 connection so they are forced to use transition mechanisms which transports IPv6 packets through IPv4 network. Unfortunately deployment of IPv6 is slow and at this rate, completion of the migration from IPv4 to IPv6 will take several years. Until then tunneled IPv6 traffic will be present on most networks. This means possible security risk because many of nowadays network tools and firewalls just see IPv4 traffic and content of the encapsulated IPv6 traffic is hidden. We do not know, what kind of traffic is inside of these tunnels, which services are used and if the traffic does not bypass security policy. This paper proposes an approach, how to monitor IPv6 tunnels even on high-speed networks. The contribution of this approach is a possibility of monitoring what is inside IPv6 tunnels. This gives network administrators a way to detect security threats which would be otherwise considered as harmless IPv4 traffic. This approach is also suitable for long term network monitoring. This is achieved by the usage of IPFIX (IP Flow Information Export) as the information carrying format. The proposed approach also allows to monitor traffic on 10 Gbps links, because it supports hardware-accelerated packet distribution to multiple processors. A system based on the proposed approach is deployed at the CESNET2 network, which is the largest academic network in the Czech Republic. This paper also presents statistics about tunneled traffic on the CESNET2 backbone links.

Keywords: IPv6, Teredo, ISATAP, 6to4, network monitoring, IPv6 tunnel, IPFIX, FlowMon

1 Introduction

Unallocated IPv4 addresses are almost exhausted [1]. Depletion of addresses does not mean end of Internet, but it seems to be reasonable to implement support also for IPv6 protocol. End users have nowadays IPv6 ready computers, because support for this protocol is available in main operating systems (Windows, Unix, Mac OS). Unfortunately, not every ISP has implemented IPv6

support yet, which together with IPv6 backward incompatibility with IPv4 protocol requires transition mechanisms. 6to4, Teredo and ISATAP are the most used transition techniques. Using these techniques, the end user can access the IPv6 network, even if his ISP does not provide native IPv6 connectivity. These three methods use encapsulation of IPv6 protocol inside IPv4 protocol - tunneling. Encapsulation of IPv6 protocol in IPv4 hides the IPv6 traffic from the network administrator. Tunneled traffic may look like ordinary IPv4 traffic using UDP protocol, so administrators do not know, which IPv6 network service is requested, how much traffic flows through tunnels etc. IPv6 tunnels are created automatically so there is no need for a user intervention. This can cause security problems such as bypassing firewalls, unauthorized use of services etc.

We propose an approach how it is possible to overcome this limitation and to be able to monitor tunneled IPv6 traffic. It features hardware-accelerated packet distribution inside FPGA based network interface card. 10 Gbps line rate processing speed is achieved by packet distribution on multi-core processors. Traffic statistics presented in this paper are generated from IPFIX data collected on CESNET2, which is the largest academic network in the Czech Republic.

The paper is organized as follows. Section 3 describes related work. IPv6 transition techniques are described in Section 4. Proposal of architecture for monitoring tunneled data is in Section 5.3. Section 6 shows several statistics and analysis from network monitoring and Conclusion is in Section 7.

2 Contribution

Contribution of this paper consists of several parts. First, we propose an approach, how to extend IPFIX to provide possibility to monitor tunneled IPv6 traffic. This approach is scalable and can be used in large networks for monitoring IPv4, native IPv6 and tunneled IPv6 traffic. It is possible to use our concept to collect these traffic on high-speeds 10 Gbps links without sampling. Thus statistics, accounting or security problem detection using these data are more accurate. Second, we present several statistics for tunneling mechanisms such as tunneled traffic distribution, protocols used in tunnels and traffic volumes compared to native IPv6 and IPv4 traffic. Deployment of IPv6 protocol is on the rise because new operating systems use this protocol by default. Therefore more services are accessible through IPv6 protocol and traffic distribution is nowadays completely different than before. Hence current statistics are very useful.

3 State-of-the-Art

Several papers discuss and present IPv6 address and traffic analysis. Authors in [6] analyze traffic from a US Tier-1 ISP. Analyzed traffic in their data-set consist mainly of DNS and ICMP packets. They believe that it is because ISP's customers consider IPv6 traffic still as experimental. For IPv6 address assignment they used methodology introduced in [7]. It is interesting to see, how small

amount of hosts in their data-set uses privacy extensions [4]. Nowadays temporary addresses should occur more often, because privacy extensions are enabled by default in Windows 7, Vista or XP. Linux and Mac OS use auto-configured addresses by default. Statistics from a China Tier-1 ISP are presented in [5]. Their observation about address assignment and application usage are similar to ours with some exceptions. Their traffic contains higher proportion of native IPv6 traffic. We believe, that it is due to higher rate of IPv6 deployment in China and Asia. 6to4 is the most used transition mechanism there. This should be same also in Europe, because if a device has public IPv4 address, 6to4 tunneling mechanism is used first. However Teredo is nowadays on the rise, because peer-to-peer programs use Teredo to share data with more peers.

Unfortunately analysis of tunneled IPv6 traffic is missing in many papers. Some statistics are presented in [6] but just for Teredo traffic. Paper [8] observes IPv6 traffic on 6to4 relay but it is quite old. Despite our best efforts we did not find publications about tunneled IPv6 traffic in ISATAP tunnels. Statistics about 6to4 tunnels or Teredo are not so detailed and up to date. This paper tries to update knowledge about nowadays native and tunneled IPv6 traffic.

4 Transition Techniques

IPv6 connectivity is preferred in systems like Windows Vista, or Windows 7 by default. If a Windows station is connected to some local IPv4 network without native IPv6 connectivity and web site or another network service is accessible through both protocols, IPv6 has priority and a host tries to communicate through this protocol first. Because IPv6 is not compatible with the IPv4 protocol, different types of transition techniques were proposed [13]. The most interesting are tunneling techniques, because we do not know, which protocols and services are used inside the tunnels. 6to4, Teredo and ISATAP are todays most used tunneling mechanisms for connection to IPv6 network. These mechanisms are described in the next sections.

4.1 6to4 Tunneling

6to4 tunneling is the most used transition technique today [10]. According to priority in operating systems as Windows 7 or Windows Vista, if a network device has public IPv4 address, 6to4 is the first mechanism to be used. A host with public IPv4 address construct an IPv6 prefix according to following rule. After IPv6 format prefix (001) another 13 bits (0x0002) are assigned by IANA. Expressed as an IPv6 prefix it is 2002::/16. Next 32 bits are taken from globally unique IPv4 address and another 16 bits are site-level aggregation identifier for creating local addressing hierarchy and to identify a subnet. Last 64 bits are used as EUI (End Unit Identifier). Several techniques can be used to create the identifier such as auto-configuration based on EUI-64, manual assignment or randomly generated identifier according to [4]. Default configuration in Windows or Linux use well-known EUI values in practice. Linux use the value 1 by default

and Windows XP, Vista, 7 use IPv4 address in lower 32 bits of the EUI [15]. When sending packets, the 6to4 tunnel wraps an IPv6 datagram into an IPv4 datagram with protocol number 41. Then it sends it to the first available 6to4 relay router. 6to4 relay routers use any-cast prefix 192.88.99.0/24.

4.2 Teredo

Teredo was designed to be able to send network traffic through NAT [9]. It does not encapsulate IPv6 packet in protocol 41 but send it via UDP packet on default port 3544. Teredo tunneling mechanism consists of three components: clients, relays and servers. Teredo client adds or removes tunneling headers and sends a packet to the Teredo server. Servers are usually statically configured on the client and in Windows OS the address is *teredo.ipv6.microsoft.com*. Teredo relays are used for routing and bridging the IPv4 and IPv6 networks. Teredo address is more complicated then 6to4 because relay needs to reach a client behind NAT. The assigned prefix for Teredo is 2001:0::/32. Next 32 bits are IPv4 address of the client's Teredo server following with 16 bits for the flags field and 48 bits corresponding with the Teredo port and client's external address. IPv6 packet is carried in UDP payload that can have four different formats as shown in Figure 1. When simple encapsulation is used only the IPv6 packet is carried as the payload of a UDP. Server may insert origin field in the first bytes of the UDP payload to indicate that some packets were received from third parties [9]. Another possibility can be usage of authentication field when exchanging router solicitation and router advertisement messages between a client and its server. Qualification, secure qualification and creating a NAT hole is behind the scope of this article.

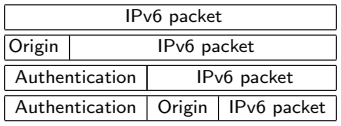

Fig. 1. The format of UDP payload in Teredo packet. In case of the simplest encapsulation whole UDP payload consists of encapsulated IPv6 packet only. In other cases there are also Teredo specific headers before IPv6 packet. Order of these headers is specified in [9].

4.3 ISATAP

ISATAP (Intra-Site Automatic Tunnel Addressing Protocol) is an IPv6 transition mechanism used in local networks to connect islands of IPv6 nodes over IPv4 networks. Connection to the Internet is made by another mechanism similar to 6to4. ISATAP[11] like 6to4 mechanism uses encapsulation of an IPv6 datagram into an IPv4 datagram with protocol number 41. IPv4 addresses are encoded in

the low-order bits of the IPv6 addresses, allowing automated tunneling. Since ISATAP does not support multicast and IPv6 Neighbor Discovery protocol, host must be configured with potential routers list as next-hops. In practice, this list is automatically obtained by querying DNS for *isatap.domain* record. Nowadays ISATAP is usually the last used transition technique. Transition techniques order which a host tries when does not have native IPv6 connectivity is usually 6to4, Teredo, ISATAP.

5 Architecture and Implementation

The proposed approach for tunneled IPv6 traffic monitoring describes whole process of flow generation, exportion and collection. The flow generation is handled by the FlowMon exporter based on preprocessed data from input plug-in. The FlowMon exporter is a software probe which is able to export flow statistics in NetFlow and IPFIX format. The exporter supports input plug-ins, so it is able to generate flow statistics from any source supported by the input plug-in [2].

5.1 Architecture

The proposed approach consist of three layers (see Figure 6). The first layer can be a network card or a more specialized hardware. The purpose of this layer is capturing packets and sending them over the software interface to the input plug-in. What software interface is used depends on the capturing device. If standard network card is used the interface can be provided by PCAP (Packet Capture) library. If a specialized hardware is used, for example FPGA (Field Programmable Gate Array) hardware accelerator like the COMBOv2 card, a more appropriate software interface can be used. In case of the COMBOv2 card it is SZE2(Straight Zero Copy) data interface. This software interface is more suitable for this task as it provides high-speed transfers.

We developed design for the COMBOv2 card which generates a nanoseconds timestamp for each packet and can be distribute packets to several DMA (Direct Memory Access) channels. Packet distribution is one of benefits of proposed approach and is described in Section 5.2.

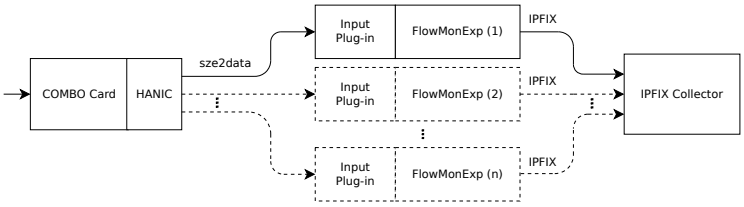

Fig. 2. Architecture overview. Packets are captured by the COMBOv2 card and can be distributed to up to 16 FlowMon exporter instances with loaded input plug-in. IP flows are generated based on processed packets and later exported in IPFIX format.

The second layer reads packets from the software interface and pass them to the input plug-in of the FlowMon exporter. The exporter is able to export NetFlow and IPFIX data. Thanks to the input plug-in support, it is able to generate flow statistics from any source as long as input plug-in supports it [2]. We designed and implemented plug-in for monitoring of IPv6 tunneled traffic but plug-ins can have any other functionality. We have also implemented plug-in for monitoring the regular traffic. Multiple instances of exporter with different plug-ins can read data from the same SZE2 interface so it is possible to process same packet by multiple input plug-ins.

The plug-in for tunneled IPv6 traffic monitoring detects packets, which are part of tunnels, using a defined set of rules. After tunnel is detected, IPv4 header is stripped out and packets are processed by IPv6 header parser. Relevant information from packet are stored to a data structure representing a part of flow (in this case flow containing single packet). This filled data structure is passed to the exporter. More about plug-in functionality can be found in Section 5.3. The exporter generates flow statistics based on data structures from the input plug-in. Flow statistics are exported in IPFIX format using custom IPFIX templates with enterprise-specific information elements to carry information about the tunnel (see Figure 3 for example of template). [12]

The third and last layer is an IPFIX collector.

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5	6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
Set ID	Length = 82
Template ID	Field Count = 14
0 sourceIPv6Address 27	Field Length = 16
0 destinationIPv6Address 28	Field Length = 16
0 flowStartSysUpTime 22	Field Length = 4
0 flowEndSysUpTime 21	Field Length = 4
0 nextHeaderIPv6 193	Field Length = 4
0 sourceTransportPort 7	Field Length = 4
0 destinationTransportPort 11	Field Length = 4
0 OctetDeltaCount 1	Field Length = 8
0 PacketDeltaCount 2	Field Length = 8
1 TunnelIPv4SrcAddr Id. 10	Field Length = 4
Private Enterprise Number	
1 TunnelIPv4DstAddr Id. 11	Field Length = 4
Private Enterprise Number	
1 TunnelSrcPort Id. 12	Field Length = 2
Private Enterprise Number	
1 TunnelDstPort Id. 13	Field Length = 2
Private Enterprise Number	
1 TunnelType Id. 14	Field Length = 2
Private Enterprise Number	

Fig. 3. An example of the proposed IPFIX template. Enterprise-specific information elements are used to carry information about the tunnel.

5.2 Packet Distribution

This section describes packet distribution on the hardware level. Packet distribution is implemented in the HANIC (Hash Network Interface Card) design for the COMBOv2 card. Its purpose is to distribute packets between several instances of the FlowMon exporter. An example of distribution method can be round-robin. This simple method is very effective and provides almost even load on all instances of the FlowMon exporter. Unfortunately it is not suitable for a flow generation because every instance of the FlowMon exporter has its own flow cache. Distribution of packets from one flow to different instances of the exporter would break the flow into more noncontinuous flows. This behavior is unwanted, so more advanced distribution, which meets requirements for correct flow export, is implemented in the HANIC design.

The HANIC design provides more suitable packet distribution using its own packet header parser. The parser can extracts necessary fields for flow identification. Extracted fields are source and destination IP address (128 bits), source and destination port (16 bits), protocol number (8 bits), IP version (4 bits) and number of input interface on the card (1 bit).

If the field is not present in the packet header, all bits of this field are set to 0. Also all missing bits are set to 0 to extend length of IPv4 addresses from 32 bits to 128 bits. The output of parsing unit is a sequence of bits with fixed length of 301 bits. This sequence is then passed to the HASH unit which computes CRC hash with length of $log_2(number\ of\ channels)$. Each packet is send to one of channels according to its hash (the hash is used to address a channel). Current version of the design use hash length of four bits. That allows addressing of 16 channels.

This allows to distribute packets to up to 16 instances of the FlowMonExp without breaking the flow cache. Another advantage is a possibility to process packets on multiple processors which greatly improve overall performance.

5.3 Plug-in Implementation

The input plug-in is implemented as a shared library for Linux operating system. It filters and preprocess each packet to data structure compatible with the FlowMon exporter plug-in API (principle of packet processing is shown in Figure 4). The input plug-in is written to be used with the HANIC design loaded into COMBOv2 card. Packets reading is handled by SZE2 library, which provides complex and low level I/O operations for the COMBOv2 card. The plug-in can be easily modified to use another library, e.g. libpcap, allows reading packets from standard network card. Thus the COMBOv2 card is not needed.

The input plug-in reads packets from the COMBOv2 card in a form of memory chunks. These memory chunks consist of whole packet together with high precision timestamp and card's interface from which packet was read. Each packet is processed by sequence of simple filters. The first filter handles detection of MPLS label. If MPLS label is detected it is stripped until all MPLS labels are processed and MPLS presence is marked. If there was a MPLS label the plug-in

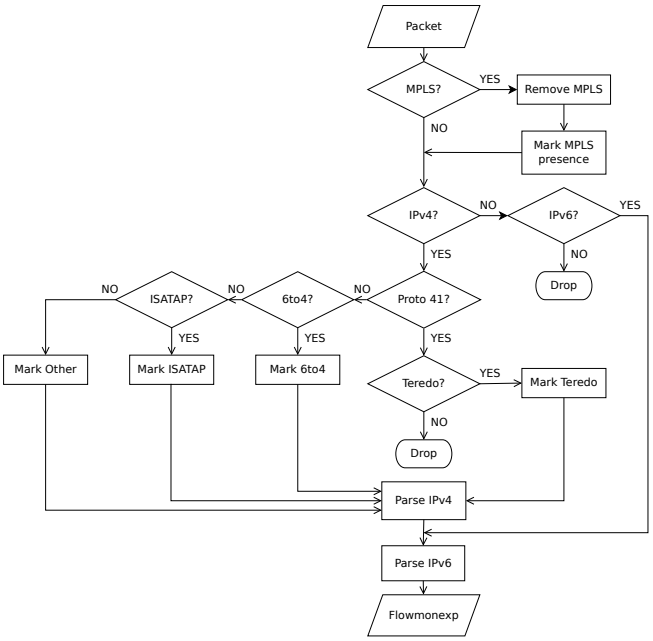

Fig. 4. Input plug-in overview. This diagram describes process of packet filtering and preprocessing. On input is single packet and for IPv6 and tunneled IPv6 packets is filled data structure on output.

determines what kind of header is following the last MPLS label otherwise it extracts protocol number from Ethernet header. If IPv4 or IPv6 header is detected the processing continues, otherwise it stops and the packet is discarded.

In the next stage of processing are IPv6 packets passed directly to IPv6 packet parser. All IPv4 packets are processed by the rest of the filters to detect presence of tunneling. Current version supports the following tunneling mechanisms: Teredo, 6to4 and ISATAP.

The first rule of Teredo detection is IPv4 protocol set to UDP. If this rule is passed then the plug-in tries to determine if there is any sort of Teredo encapsulation in the UDP payload. There are four types of encapsulation of IPv6 packet in UPD payload. The payload must be in one of these forms (see Section 4.2 for more details). If one of these forms of encapsulation is found, pointer to beginning of IPv6 packet is passed to IPv6 packet parser. Last step of processing is confirmation if at least one IPv6 address from IPv6 header is in format which is specified in [9]. If this is confirmed, the plug-in sets type of tunnel to indicate usage of Teredo and pass filled data structure to exporter.

Detection of ISATAP and 6to4 packets is similar as they share some characteristic. IPv4 protocol must be set to value 41. In both mechanisms IPv4 header is followed by IPv6 header. The plug-in checks if there is IPv6 header and pass it to IPv6 packet parser. To decide if IPv6 packet is encapsulated by ISATAP or

6to4 plug-in checks IPv6 addresses and looks for address in format specified for 6to4 or ISATAP. If it is found then type of tunnel is set to corresponding tunneling mechanism. If ISATAP or 6to4 address is not found, the type of tunneling mechanism is set to indicate usage of unknown tunneling mechanism. Filled data structure is passed to the FlowMon exporter.

5.4 Packet Processing Performance

Packet processing performance was measured by throughput test, during which were processed packets from 10 Gbps Ethernet network link. The measurement run on 2.0GHz quad-core CPU and beside throughput was also monitored CPU usage. Throughput was measured for Teredo and 6to4 packets (throughput of ISATAP packets is the same as throughput of 6to4 packets). In the first scenario all packets were processed by single instance of the FlowMon exporter with loaded input plug-in. This setup was unable to process all packets on small packet lengths even with full load on one CPU core. In the second scenario packets were distributed to 4 instances of the FlowMon exporter with loaded input plug-in. Each instance of the FlowMon exporter was running on different CPU core providing more computing power for processing. All packets on all feasible lengths were processed with medium to low CPU load on every core. Results are shown in Figure 5.

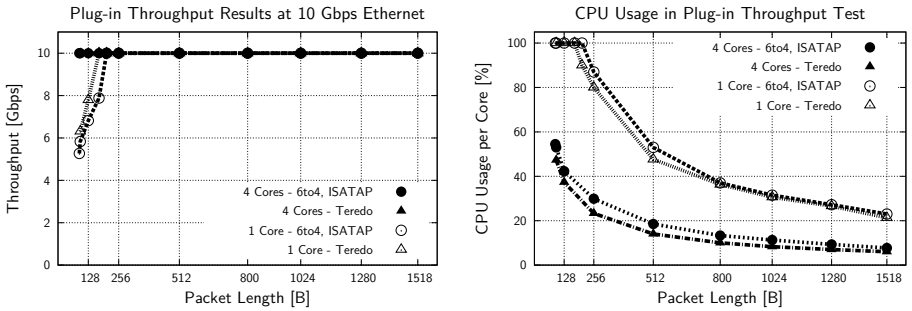

Fig. 5. Throughput of the plug-in at 10 Gbps Ethernet and CPU usage during the test on single and on 4 CPU cores.

To minimize impact of flow generation on performance results all packets in the first scenario originated from single flow. In case of the second scenario four different flows were used. Every flow of the four flows was selected in way so that the hash generated in the COMBOv2 card was different for packets from different flow.

6 System Evaluation - Monitoring of Real Network Traffic

We deployed monitoring system based on the proposed approach on the CESNET2 network (Czech Republic's National Research and Education Network). Three 10 Gbps backbone links which are connecting the CESNET2 network to SANET (Slovak academic network), PIONIER (Polish optical Internet) and NIX.CZ (Neutral Internet eXchange of not only Czech Republic) networks were monitored (for more details see Figure 6). Each network link was monitored by one probe with the COMBOv2 card installed and both directions of network link were connected to it. These probes ran the FlowMon exporter with the input plug-in. Up to this point everything was according to the proposed approach but in this experiment we were forced to slightly change the IPFIX templates in way that they don't satisfy the IPFIX standard. An example of used template is shown in Figure 7. The reason for this change was a IPFIX collector which we used. It was NfSen collector with added support for IPFIX. It doesn't have full support for IPFIX as it doesn't support enterprise-specific elements [16].

We selected 14 days long interval from collected flow statistics and analyzed it. The results are shown is Section 6.1 and Section 6.2.

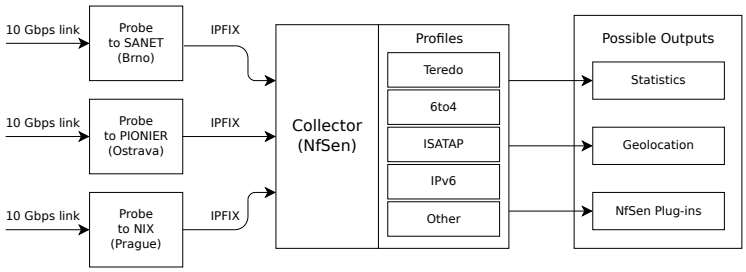

Fig. 6. Overview of monitoring setup. The probes generate flow statistics from IPv4, IPv6 and tunneled IPv6 traffic and send them using IPFIX format to the collector. The collector stores flow statistics and provides them for further processing.

6.1 Observed IPv6 Address Assignment

A host in an IPv4 network obtains a IPv4 address usually from a DHCP server together with default gateway, network mask etc. Address assignment is a little bit different in IPv6 networks. There is also a possibility to use a DHCPv6 server, but usually stateless auto-configuration is used [14], so a host learns just network prefix and default gateway. The lower part of IPv6 address (last 64 bits) is a host identifier and can be assigned manually, based on EUI-64 algorithm or generated randomly according [4]. Tunneling mechanisms use different techniques as described in Section 4. IPv6 address analysis shows some interesting results. We

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5			6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
Set ID			Length = 62
Template ID			Field Count = 14
0	sourceIPv6Address	27	Field Length = 16
0	destinationIPv6Address	28	Field Length = 16
0	flowStartSysUpTime	22	Field Length = 4
0	flowEndSysUpTime	21	Field Length = 4
0	nextHeaderIPv6	193	Field Length = 4
0	sourceTransportPort	7	Field Length = 4
0	destinationTransportPort	11	Field Length = 4
0	OctetDeltaCount	1	Field Length = 8
0	PacketDeltaCount	2	Field Length = 8
0	TunnelIPv4SrcAddr	410	Field Length = 4
0	TunnelIPv4DstAddr	411	Field Length = 4
0	TunnelSrcPort	412	Field Length = 2
0	TunnelDstPort	413	Field Length = 2
0	TunnelType	414	Field Length = 2

Fig. 7. An example of used IPFIX template. Enterprise-specific information elements are replaced by regular elements with nonstandard IDs.

use similar methodology for address classification as in [7] but some addresses are analyzed in detail.

Table 1 shows average number of unique IPv6 addresses in native, 6to4, Teredo and ISATAP traffic per day. There is very high number of Teredo addresses. Further examinations shown that Teredo is used mainly for p2p sharing. We believe that it is because bittorrent clients such as *µTorrent* have implemented Teredo support, to be able to share data with more peers. Teredo is first mechanism used when a device is behind a NAT. It provides global IPv6 address so the device can be accessed same way as with public IPv4 address.

We detected several Teredo servers as well. Native and 6to4 addresses are more analyzed and results are showed in Table 2. One table describes in detail 6to4 addresses in native and tunneled traffic. Autoconf means, that EUI is generated according to EUI-64. Linux and Windows rows describes, how many hosts use Windows and Linux/Unix operating systems. This detection is based on default values for the EUI fields [15]. Privacy means, that EUI is generated according to privacy extensions. The table confirms, that tunneling protocols are use by default in recent Windows operating systems (Vista, 7). The other table shows address structure of global IPv6 addresses in native and tunneled traffic. This also confirms, that privacy extensions are used by default in Windows operating systems. Low addresses are usually configured statically and auto-configuration is used by operating systems such as Linux, Unix and Mac OS.

6.2 Observed Tunneled Traffic Characteristics

The first interesting fact about IPv6 tunneled traffic is, according to our findings, that it generates more traffic then native IPv6 traffic. This fact is true for all of three metrics (by flow, by packets and by bytes) and is shown in Table 3. As

Traffic	Unique Addresses	Notes
Native IPv6	8059 (10.1%)	details in Table 2
6to4	20090 (25.3%)	details in Table 2
Teredo	51330 (64.5%)	detected 13 Teredo servers
ISATAP	82 (0.1%)	

Table 1. IPv6 unique addresses - average per day.

6to4	Native	Tunneled Traffic
Autoconf	2.7%	1.4%
Linux	1.2%	0.3%
Windows	91.2%	85.6%
Privacy	4.9%	12.7%

IPv6	Native	Tunneled Traffic
Autoconf	9%	4.2%
Privacy	69.2%	69%
Low	21.8%	26.8%

Table 2. 6to4 and other global IPv6 addresses in detail.

described earlier, the reason for this can be presence of tunneling mechanisms in recent versions of Windows operating system. Traffic is mostly generated by universities. This can be another reason small IPv6 traffic, because even thou universities should be deploying IPv6 they still prefer IPv4 in academic networks. Most universities have enough global IPv4 addresses so they are not forced to deploy IPv6.

	Flows	Packets	Bytes
IPv4	98.39%	99.19%	99.13%
Native IPv6	0.10%	0.12%	0.21%
Tunneled IPv6	1.50%	0.69%	0.66%

Table 3. Traffic shares of IPv4, IPv6 and tunneled IPv6 traffic.

Majority of IPv6 tunneled traffic use Teredo mechanism (see Table 4). This corresponds with previous analysis of IPv6 addresses, where Teredo addresses are also the most significant traffic contributors. The least used mechanism is ISATAP that may be given by fact that it is the least preferred option of tunneling in Windows operating systems.

	Flows	Packets	Bytes
Teredo	88.18%	89.10%	88.85%
ISATAP	0.06%	0.03%	0.03%
6to4	11.76%	11.76%	11.12%

Table 4. Monitored tunneling mechanisms distribution.

Another interesting fact is that in tunnels are computers communicating mainly with hosts which use the same tunneling mechanism on their side. More

details can be found in Table 5. Only small part of communication is with hosts using native IPv6 addresses.

	to Teredo	**to 6to4**	**to ISATAP**	**to Native IPv6**
Teredo Tunnel	59.6%	30.0%	0.1%	10.3%
6to4 Tunnel	2.5%	89.8%	<0.1%	7.7%
ISATAP Tunnel	1.2%	19.4%	64.4%	15.1%

Table 5. Distribution of traffic inside IPv6 tunnels by number of flows.

We also observed very different distribution of application protocols in tunneled IPv6 traffic. One of the most used protocols in IPv4 and native IPv6 traffic is HTTP (Hypertext Transfer Protocol). In tunneled IPv6 traffic its share was very small and the traffic was overall spread to hundreds of UDP and TCP ports with high numbers. This and the fact that majority of tunneled traffic is transferred over Teredo was reason for further examination of IPFIX data. We come to conclusion that tunneled IPv6 especially Teredo is used for p2p sharing. Reasons, why p2p programs use Teredo are described in Section 6.1.

By Flows	**IPv4**	**Native IPv6**	**Tunneled IPv6**
HTTP	38.25%	1.99%	0.35%
HTTPS	3.26%	<0.01%	0.08%
DNS	10.39%	61.76%	0.45%

By Packets	**IPv4**	**Native IPv6**	**Tunneled IPv6**
HTTP	49.99%	65.50%	2.98%
HTTPS	1.72%	<0.01%	2.85%
DNS	0.45%	1.68%	0.05%

By Bytes	**IPv4**	**Native IPv6**	**Tunneled IPv6**
HTTP	56.80%	76.16%	0.38%
HTTPS	1.17%	<0.01%	0.33%
DNS	0.07%	0.42%	0.01%

Table 6. Protocol distribution in tunneled and native traffic.

7 Conclusion

Current flow-based traffic monitoring techniques can not easily analyze tunneled traffic. It is especially problem in IPv6 networks. In IPv6 networks tunnels are created automatically, without users or administrators intervention. Because IPv6 protocol is not compatible with current IPv4, these tunneling mechanisms would be needed for several years. Network administrators need an approach,

which is able to monitor tunneled traffic on high-speed networks, is scalable and can be integrated into current monitoring systems. In this paper we propose such approach.

Monitoring of 10 Gbps link is possible using hardware-accelerated network cards. We implemented plug-in for the FlowMon exporter, which can monitor tunneled IPv6 traffic and export obtained data using IPFIX format. Collected data can be further analyzed by network security monitoring tools including IPS and IDS. We successfully deployed the proposed solution on academics backbone links in the Czech Republic and analyzed the collected data.

Acknowledgments. This work is supported by the Research Intent of the Czech Ministry of Education MSM6383917201.

References

1. IPv4 Address Report, [online], cited [30.09.2010] `http://www.potaroo.net/tools/ipv4/index.html`.
2. INVEA-TECH a.s., [online], cited [30.09.2010] `http://www.invea-tech.com/products-and-services/flowmon/flowmon-probes`.
3. Aben, E., Measuring IPv6 at Web Clients and Caching Resolvers, [online] cited [2010-10-04] `http://labs.ripe.net/Members/emileaben/content-measuring-ipv6-web-clients-and-caching-resolvers-part-1/`.
4. Narten, T., Draves R., Krishnan S., RFC 4941, Privacy Extensions for Stateless Address Autoconfiguration in IPv6,September, 2007.
5. Shen W., Chen Y., Zhang Q., et al., Observations of IPv6 traffic, In Computing, Communication, Control, and Management, 2009, vol. 2, ISBN 978-1-4244-4247-8, p. 278 - 282.
6. Karpilovsky E., Gerber A., Pei D., Rexford J., Shaikh A., Quantifying the Extent of IPv6 Deployment, In Passive and Active Network Measurement, 2009, p. 13 - 22.
7. Malone D., Observation of IPv6 Addresses, In Passive and Active Network Measurement, 2008, p. 21 - 30.
8. Savola P., Observations of IPv6 Traffic on a 6to4 Relay, ACM SIGCOMM CCR vol. 35, no. 1, pp. 23-28, Jan. 2005.
9. Huitema C. Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs). RFC 4380, February 2006.
10. Carpenter B., Moore K. Connection of IPv6 Domains via IPv4 Clouds. RFC 3056, February 2001.
11. Templin D. T. F., Gleeson T., Intra-Site Automatic Tunnel Addressing Protocol (ISATAP). RFC 5214, March 2008.
12. Claise B., Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information. RFC 5101, January 2008.
13. Nordmark E., Gilligan R., Basic Transition Mechanisms for IPv6 Hosts and Routers. RFC4213, October 2005.
14. Thomson S., Narten T., Jinmei T., IPv6 Stateless Address Autoconfiguration. RFC4862, September 2007.
15. Warfield H. M., Security Implication of IPv6, Internet Security Systems, 2003.
16. Krejčí R., Network Traffic Collection with IPFIX Protocol, [online], cited [2010-10-04] `http://is.muni.cz/th/98863/fi_m/xkrejc14_dp.pdf` .