Hiding TCP Traffic: Threats and Counter-measures

Libor Polčák, Radek Hranický, Petr Matoušek

ipolcak@fit.vutbr.cz, xhrani00@stud.fit.vutbr.cz, matousp@fit.vutbr.cz

Faculty of Information Technology
Brno University of Technology
Božetěchova 2, 612 66 Brno, Czech Republic

Abstract

Computer networks were designed to be simple and routers do not validate the integrity of the processed traffic. Consequently, an attacker can modify his or her traffic with the aim of confusing any analyser that intercepts the traffic, e.g. monitoring and security software or lawful interception. This paper studies the attack that is based on sending additional colliding TCP segments with the same sequential number but different content. The segments with the correct message are delivered to the other communicating party of the TCP connection while the fake segments are dropped en route. The goal of the fake segments is to confuse analysers into decoding a different message to the one that is received by the other communicating party. The other communicating party does not need to be aware of the attack and therefore does not need any specific software. Although this paper discuss the advantages and disadvantages of the attack for an attacker, our ultimate goal was to find counter-measures against the attack. Our contribution can be divided into four following parts. 1) We converted the attack to IPv6 and searched for possibilities that may force a middle box to drop fake packets. 2) We developed a tool called LDP, which behaves as a TCP proxy server that masks outbound TCP traffic of a whole network. 3) We identified several counter-measures. In addition, we implemented LNC, a tool that identifies the attack in pcap files and removes the fake segments. Since LNC is a stand-alone tool, it also deals with traces generated by other software than LDP as long as it is based on the same attack vector. 4) LDP and LNC were tested in both laboratory environment and on the Internet. The experiments validated that the attack is applicable for a communication with a server that is not under the control of an attacker. Several parameters of the attack were evaluated during the experiments; mainly the number and the length of fake packets and their influence on the performance of the attack and counter-measures.

Keywords: Law Interception, data hiding, TCP, covert channels.

1 Introduction

Internet has become an integral part of human lives because it brought new benefits and opportunities for both personal and business sector. Therefore, it is important to ensure its stability and functionality. However, this is not an easy task in the current packet-switched computer networks. Computer networks were developed for with the goal of being robust and tolerant to failures of a part of a network. Best effort delivery principle ensures that middle-boxes, such as routers and switches, relay all packets to the destination. Networks were designed to be dumb at the core and the intellect was shifted to the edges of the network, mostly to the end devices.

In order to be robust, network protocols were defined in layers. Each layer fulfils a specific task. Nowadays, the dominant model is TCP/IP suite. Internet Protocol (IPv4 and IPv6) and Transmission Control Protocol (TCP) are the most common protocols of the suite. The task of IP is to deliver datagrams from one host to another. To prevent loops in the network, IP drops packets that traverse too many routers. TCP creates a bi-directional data stream that allows two network applications to exchange

data in both directions. Data in the stream are segmented by TCP and passed to IP to be transported to the other host. TCP has to deal with duplicate and out-of-order segments or any other error which may appear as the result of the unreliability of IP. To achieve this, each segment is marked with a sequential number, which determines the position of each byte of the segment in the TCP stream.

Criminal activities are also part of the Internet. As the consequence of the best effort delivery principle, regular middle-boxes do not differentiate between benign and malicious traffic. Over time, several approaches that deal with security threats emerged. Firewalls are the most common of security devices. Their goal is to separate a trusted part of the network and untrusted Internet. Firewalls operate on the border between these two parts where they drop traffic that is evaluated to be harmful. Intrusion detection systems (IDS) and intrusion prevention systems (IPS) search for specific traffic patterns that are associated with malicious activities. IDSes report security incidents to network operators whereas IPSes reconfigure the network devices to stop the threat or minimise its impact.

Firewalls and IDS/IPS are deployed for both private and corporal networks. In addition, law enforcement agencies (LEAs) also fight against criminal activities; including electronic crime and criminals that communicate through Internet. During e-investigation, one of the tasks is to decode TCP streams and use them as evidence in lawsuit.

However, the design of computer networks gives malicious users the opportunity to hide their traffic and confuse traffic analysis. Network *covert channels* [17] allow seemingly innocent communication to be used as an overt channel for secret transfers of sensitive information. That means that the significant data are not detected by security devices and criminal investigators. *Confusion* [1] in the network aims to spoil traffic analysers into believing that the real traffic was not transferred at all or it hides the real communication among a *noise* so that security devices decode false information. Confusion may take place on different layers. For example, electrical characteristics might hide traffic from less sensitive devices [1], a bug in a specific application might be misused to hide sending of an e-mail [13], etc.

This paper focuses on a specific attack [1, 11] that confuses traffic analysers into believing that the exchanged message is different to the one which is seen by the other communicating party. The attack manipulates one direction of a TCP stream so that it is not straightforward to decode the data that were transferred in the stream. The sender or a proxy near the sender transmits the *original message* together with *cover messages* and *noise*. The attack is based on a transmission of colliding TCP segments with the same sequential number but different content. IP headers are modified so that the fake TCP segments are dropped by the network before they can reach the receiver. Only the segments with the original message are delivered to the receiver. However, the middle-boxes see both original and crafted segments and, consequently, decode spoofed message. Some of them decode the message carried in the first segments that are seen with each sequential number whereas others decode message from the last segments seen with each sequential number. To confuse those middle-boxes that recognize the attack, the correct message is shuffled among noise segments so that it is not clear which datagram carries the correct segment. The attack is depicted in Figure 1. The attack is completely transparent for the receiver and does not require any specific protocol; the confusion is performed at TCP level.

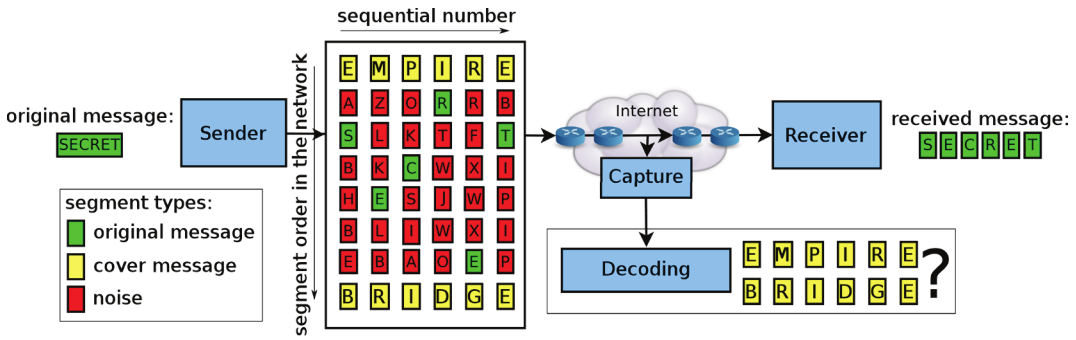

Figure 1: Attack overview: A sender sends the original message shuffled among noise and cover messages. Even though the receiver correctly receives the original message, decoding of the message from the data captured on the Internet is not straightforward. Naïve analysis decodes one of the cover messages.

In our research we study further details of the attack. To our best knowledge we are the first one to describe and test the attack in IPv6 networks. We studied the suitability of the attack with respect to application layer protocols because of its huge overhead. Moreover, we implemented and tested modifications of the attack that reduce the overhead and make the attack work on asymmetric routes. However, the ultimate goal are counter-measures. We revisited the work of Cronin et al. [1] and provide an updated view of the current state of a few network analysing and security software and their ability to deal with the attack. We also designed a tool that is able to detect the attack in a copy of network traffic stored in pcap files, which are often used in e-investigation. In addition, we studied the properties of the attack and propose methods for its detection, e.g. based on NetFlow).

This paper is organized as follows. Section 2 describes related work. The scheme of the attack and the modifications are discussed in Section 3. The experiments are described in Section 4. Detection of the attack and counter-measures are examined in Section 5. Section 6 concludes the paper.

2 Related Work

As this work deals with information hiding in computer networks, it is natural to consider cryptography, e.g. Transport Layer Security (TLS), Secure Shell (SSH), IPsec or Pretty Good Privacy (PGP), as an alternative to the studied attack. Even though cryptography allows hiding of the content of a communication, its downside is that the randomly-looking data transfers can be distinguished by the monitoring party. The studied attack is a bigger threat. If the monitoring party is not aware of the attack, it may decode a cover message and consequently be deceived. Moreover, the attack can be performed even if the other communicating party does not cooperate. In contrast, encrypted connection cannot be initiated without support on both communicating sides.

The studied attack may be considered as a form of a covert channel as it satisfies TCSEC definition [2]: *covert channel is a communication channel that allows a process to transfer information in a manner that violates the system's security policy*. Various work studied covert channels in computer networks [3, 17]. It is possible to hide information in packet headers [3, 6, 10, 16] or specifically crafted payload [8]. Stream metadata, e.g. packet length [12], packet ordering [4] and deliberate packet loss [14] are other options to hide traffic. Unlike the studied attack, these covert channels need modifications on both sides of the TCP connection.

Lucena et al. [6] studied IPv6 header and extension header fields for covert channels. Although our aim is different as we do not use any header field as an overt channel, we consulted this study during discussion about possibilities for dropping certain packets en route. Similarly, Zander et al. [18] studied covert

channels in TTL field. Our goal is to use TTL or Hop Limit (in IPv6) to drop packets, not as an overt channel.

It was already pointed out [1] that the studied attack may be used to spoil the results of e-investigation. Cronin et al. [1] studied methods which may confuse eavesdroppers even in the case when both parties do not cooperate and an attack is carried by only one side of the communication. We base our work on this study [1]. Our work differs in following aspects. Firstly, we provide real tools [5] for both attack execution and its detection. Secondly, we provide up-to-date state of current monitoring and security tools (see subsection 5.3). Thirdly, we studied the feasibility of the attack in real network experiments. Finally, we successfully ported the attack to IPv6.

3 Attack Scheme

This section summarizes the considered attack and its implementation for IPv6 networks. We base our work on the results of Cronin et al. [1]. In addition to their work, we transformed the attack to IPv6. Moreover, we adjusted the attack to automatically detect the distance to the other communicating party, which is crucial when the attack is based on a known number of intermediate nodes. We implemented a tool called LDP [5] that is capable of the attack. In addition to performing the attack, the tool behaves similarly to a router. Hence LDP masks traffic of a whole network. LDP was successfully tested on asymmetric routes.

3.1 Generic changes to the attack

This subsection covers the generic changes to the attack that are not specifically connected to IPv6. Whereas Cronin et al. [1] discussed one cover message that was sent before the noise, we discovered (see subsection 5.3) that some software decodes the message from the last segments that were received with the specific sequential number. Therefore, we added support for another cover message in the last segments to LDP.

During experiments, the attack overhead was huge (see subsection 4.2). To overcome this limitation, we introduced configurable-sized segments. However, the longer the segments are the easier it is to identify the segments with original message among the noise. Alternatively, the noise segments cannot be generated completely randomly but according to some frequency analysis. Otherwise, classical cryptanalysis techniques (letter, digraph, and trigraph frequency counting, short words or patterns discovery etc.) can identify the original message quickly. Additionally, instead of modifying the original TCP segments, it is possible to configure LDP to just send the first and last cover message along the original segments. In this configuration, it is expected that the segment size is so large that cryptanalysis is easy. Hence the only motivation is to confuse naïve decoders.

3.2 Basic attack scheme in IPv6 Networks

Our basic approach, implemented in LDP, is to drop packets on one of the routers on the way as a result of insufficient Hop Limit (HL), see Figure 2 for an example. Other possibilities for packet dropping in IPv6 are discussed in the subsection 3.3.

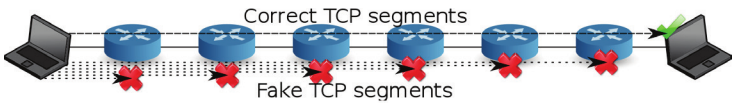

Figure 2: Correct segments are delivered to the destination while fake segments are dropped along the path.

As Figure 2 shows, it is necessary that LDP can learn the minimal HL to reach the destination. This value is denoted as R_{min} in the rest of the paper. Similarly, $D_{max} = R_{min} - 1$ denotes the maximal HL value which ensures the drop of the packet by a router. For correct function, it is necessary to learn the pair (R_{min}, D_{max}) for each remote host. LDP initiates each TCP connection with one SYN segment and stores the value of HL in the response. Most operating systems use default HL of 64 or 128 (see e.g. [9]). For symmetric routes it is possible to subtract the received HL from the expected original value of HL and compute D_{max}. However, routes may be asymmetric [7], therefore, LDP tries to analyse the path and cope with the asymmetry. The expected number of hops is validated using ICMPv6 echo requests. By repeatedly adding or subtracting 1 hop, LDP searches for the correct value of D_{max} (no echo reply received) and R_{min} (the other host replies). Nevertheless, even this method is not correct in all cases. Obviously, it does not work when ICMP echo messages are dropped by any middle-box. In addition, some paths are load balanced. In rare cases when each of the paths has a different number of hops, each has its own R_{min} and the algorithm fails. Our experiments with servers in Europe and North America did not show that such network topologies are common.

Once D_{max} is found, it is possible to ensure that the fake segments are not received by the other communicating party. As the location of a possible eavesdropper or a security device is not clear, it is reasonable to send a sufficient amount of traffic with HL high enough to reach routers near the end of the path. The goal is to make the difficulty of traffic decoding similar at each possible point of interception (IAP). In addition, it might be argued that a HL variety that is seen on each IAP is desirable. Should the eavesdropper see only two HL values, it would be easy to differentiate correct and fake segments by HL value once the attack is discovered. To overcome this, LDP generates HL for fake segments according to a specified probability distribution function. The goal is to generate enough datagrams that are received close to the other communicating party. LDP also randomize the HL value used in datagrams with correct TCP segments to prevent identification of these segments. The HL is generated randomly between R_{min} and the original input HL by default. See section 5.1 for further discussion on this topic.

3.3 Forcing datagram drops in IPv6

The sole purpose of network layer modifications is to ensure that fake packets are dropped before they are processed by the other communicating party. Besides HL modifications, IPv6 provides other possibilities to ensure the drop. However, the attack has to be modified for the specific network topology (see Figure 3). The goal for the attacker is to find such values of required or optional IPv6 header fields that a device between the eavesdropping point and the other communicating party drops the whole packet. Apart from a router or a security device on the path (Figure 3a), it can also be the operating system of the other communicating host (Figure 3b).

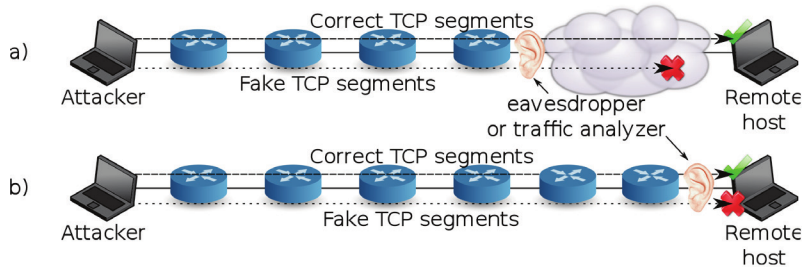

Figure 3: The knowledge of network topology might enable an attacker to use other techniques to force that fake segments are dropped a) before reaching the remote host or b) by the remote host operating system.

As Raiciu et al. studied [13], there are several types of middle-boxes in the network that change TCP segments. Firewall, IDS/IPS and other security system may even drop some packets. The exact behaviour depends on the type and configuration of the middle-boxes. In case of a security device configured to drop IPv6 datagrams marked with a specific Traffic Class or Flow Label, these fields might be used for fake segments as the only one in the basic IPv6 header as other fields in the header are important for the attack.

Additionally, IPv6 introduced extension headers, which can also be misused to ensure that fake datagrams are dropped before reaching their destination. The drop can be a result of 1) an incorrect order of extension headers, 2) a middle-box configured to drop datagrams containing a specific extension header, 3) an incorrect signature of IPsec headers (AH or unencrypted ESP), or 4) a middle-box not supporting fragments that are not split to different datagrams. In addition, the options of Hop-by-hop header contain a bit that specifies that the whole datagram should be dropped if the option is not known. Another possibility is to use routing header when the first destination does not route the datagram to the next destination. Of course, the attack might combine several of the above mentioned possibilities, including HL values.

Hence, HL modifications are not the only method that enables the attacker to force that a datagram is dropped but it is the only one that can work automatically, without prior knowledge of the network topology and position of the IAP.

4 Experimental evaluation of the attack

This section summarizes the experiments performed in the shoes of an attacker. We communicated with Internet hosts that were not under our control. That confirmed our expectations that the attack is deployable to the real world. Yet, as explained in this Section, the advantages of the attack for an attacker are questionable. Since the attack has a big overhead we performed a set of measurements to study its impact on computer networks.

4.1 Communication with servers on the Internet

First, we configured the testbed depicted in Figure 4 to evaluate that the attack works on the Internet. We connected a NAT64 translator to enable the attacker to communicate with IPv4-only hosts. It was possible to communicate with other Internet hosts, even if they were not under our control. However, the attacker needs to consider the possibility that the opposite direction of the TCP stream is being eavesdropped too. This may not be a big problem if the attacker was interested in confusing a security device to bypass some firewall or IDS rules. However, in case of a lawful interception or illegal eavesdropping, the interceptor could deduce some of the missing data from the replies of the other host.

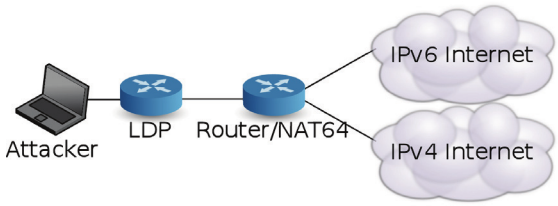

Figure 4: The testbed for the attack evaluation on the Internet.

For example, when the attacker connected to an IRC server, it was possible to communicate with other people connected to the IRC network. However, a lot of information leaked from the server responses. For example, names of the joined channels and their participants or the topics of the joined channels. Additionally, all messages sent by other participants were clearly visible in the TCP stream from the IRC server. Nevertheless, the attacker can join an IRC channel and give reports or instructions to the other channel members while the content of the messages can stay hidden to naïve analysers.

Experiments with web servers revealed similar leaks. It was possible to mask the content of the outgoing communication but the responses could often be used for identification of the content that was originally sent. For example the responses contained a preview of the message that the attacker tried to post to a discussion forum, a thread identification number after the attacker opened a new one on other forum etc. Nevertheless, some web of the servers reply with messages that does not identify the content that was just uploaded to the server and in such case, the attack could spoil the investigation.

Therefore, the attack might be misused by an attacker but only in a specific scenario when he or she knows in advance that the data in the other direction of the TCP stream does not leak substantial information. Another use case for the attack is an attacker that does not care that the content of the messages that he or she sent might be deduced from the other TCP direction, e.g. when the attack is performed only to confuse a firewall and monitoring device and the discovery after the data from the other communicating party are received does not matter.

We also tested the scenario where the attack is performed in the direction from a web server to its clients. This might be done by an illegal site to mask its content for innocent one. In such case, the intercepted data contains the HTTP GET or POST requests but the replies are hidden in the noise. In addition, the client might also run the attack to confuse the client to server direction of the TCP stream. During the testing, we discovered that one-byte segments have a very big overhead and for larger chunks of data (e.g. images and long pages), the download time is very long. We investigated the overhead further; the results are summarized in subsection 4.2. To counter this limitation, we implemented support for larger than one-byte segments to LDP, which lowered the download time considerably.

4.2 The overhead of the attack

As the overhead of the attack slows data transfers considerably, we investigated the attack overhead in theory. Every application data sent through the Internet has to be encapsulated in IP (version 6 in our case). As the attack is based on TCP, TCP header is another part of the overhead. Let us consider standard TCP connection opened by three-way handshake without any options to lower the overhead as much as possible. The SYN packet, in our case, does not carry any data.

Since the standard IPv6 header is 40 bytes long and minimal TCP header is 20 bytes long, the total header overhead for each segment is 60 bytes (the overhead is higher when extension headers are present, see subsection 3.3). The overhead and relative increase in comparison to the standard TCP is computed in Table 1. The attack overhead is in the order of magnitude or more higher than the size of a regular TCP. For 8 bytes of data and the attack with two cover messages and four noise segments for each byte of the

original message, 18.8-times higher amount of data has to be sent. For longer messages the relative increase gets even higher and for 16 KB of data, 407.6 more bytes has to be sent compared to regular TCP. Therefore, the practical use of the one-byte-per-segment configuration of the attack for bigger amount of data is questionable. Moreover, the experiments with LDP show even bigger overhead due to retransmission of several TCP segments during the TCP connections. It seems that application protocols with short messages such as IRC, XMPP etc. are most suitable for this attack configuration. Often, the size of HTTP GET requests is also short. However, POST requests or GET requests with forms might overwhelm the network and hinder the browsing performance.

L7 data length	Total length of standard TCP [B]	2 cover messages + 4 noise segments			2 cover messages		
		Total length [B]	Attack overhead	Relative increase	Total length [B]	Attack overhead	Relative increase
8 B	188	3536	94.68%	18.8	1464	87.16%	7.8
64 B	244	27448	99.11%	112.5	11712	97.92%	48.0
512 B	692	218744	99.68%	316.1	93696	99.26%	135.4
2 KB	2288	874616	99.74%	382.3	374784	99.39%	163.8
16 KB	17164	6996088	99.75%	407.6	2998272	99.43%	174.7

Table 1: The theoretical attack overhead computed for one-byte-long TCP segments.
Each flow consists of 60-bytes-long SYN and FIN segment.
Maximal size of IPv6 datagrams is 1500 bytes.

Increased TCP segment size makes the attack more appealing. As can be seen in Figure 5, the overhead lowers substantially. For longer segments (about 512 bytes), the overhead for *n* fake segments is similar to sending *n*-times longer TCP stream. Our personal experience with bigger data transfers confirmed that the performance is much better. However, once the attack with longer segments is revealed, if the original text is in plain text, human analyser can differentiate fake and correct segments. The only defence against such revelation would be a very good generator of the noise or manually prepared noise.

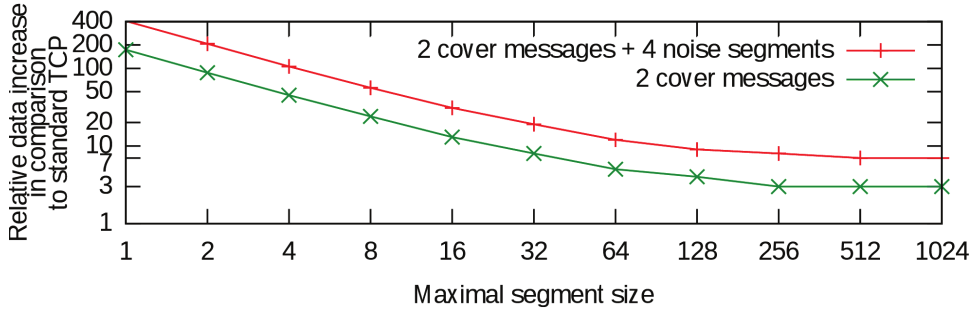

Figure 5: Relative data increase of a TCP stream with the attack for 16 KB data transfers in comparison with regular TCP.

5 Defence against the attack

For some tasks (e.g. real-time IDS/IPS) it is crucial to detect all attack vectors, whereas for other tasks (e.g. manual or computer aided data decoding) it is sufficient to report that something unusual is happening in the network or was captured in the packet trace. This section elaborates on both possibilities. Additionally, we re-evaluated software that is often used for network traffic decoding. This was originally done by Cronin et al. [1]. we were interested in possible changes in the software, support of IPv6 and the mystery of random data that Cronin et al. decoded with some of the tools. We discovered that some of these tools do not decode random data but data from the last received segments.

5.1 Attack detection

For attack detection, either packet headers or metadata analysis (e.g. NetFlow or IPFIX) can be used. Figure 6 shows an example of histogram HL values in one of the traces of a TCP session (confused by the attack) with an IRC server. Probability distribution function $f(x) = 2 \cdot x / D_{max}$ was used to generate HL values of fake TCP segments. While asymmetric load balancing may cause that more than one HL value is present in a legitimate TCP connection, more than 20 HL values are highly unlikely under normal network conditions. Therefore, HL tracking of TCP flows could reveal the attack. On the other hand, a loop in the network could also result to a similar HL distribution. Nevertheless, as Figure 6 reveals, if the attacker generates HL with different distribution for correct and fake segments, the histogram of the values seen in the packet segments may indicate R_{min} because it is obvious that datagrams with HL of 15 or lower have different distribution than datagrams with HL of 16 and higher.

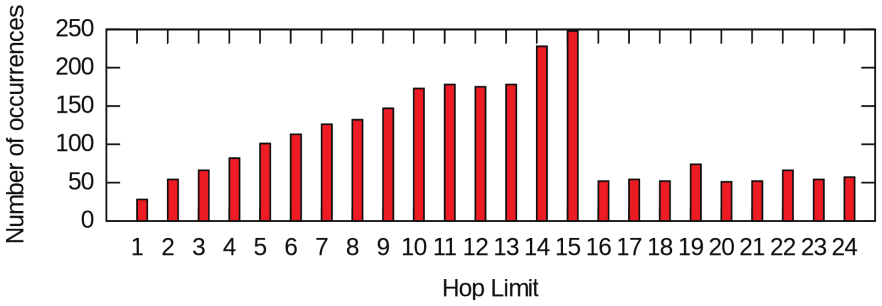

Figure 6: Histogram of HL occurrences in a TCP stream to an IRC server (R_{min} = 16).

However, should the attacker use another distribution of HL values (e.g. similar to the one depicted in Figure 7, which has the same number of fake and correct packets as is in Figure 6), the estimation of correct number of hops to the destination from the histogram alone would be impossible.

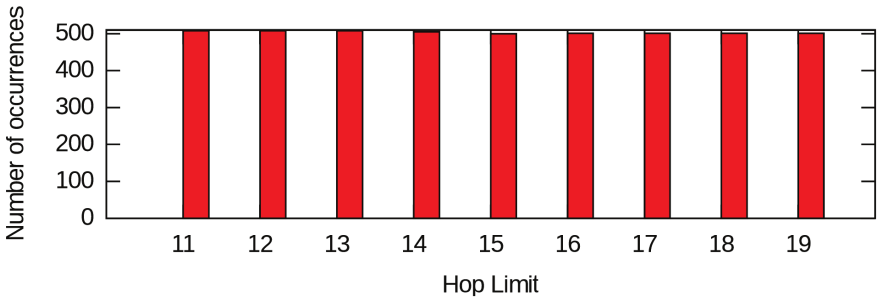

Figure 7: Alternative histogram for the TCP connection shown in Figure 6. The estimation of the pair (R_{min}, D_{max}) is impossible to make from this histogram alone.

Therefore, the attack may be detected by unusually high number of very low HL values in one TCP stream. When an attacker is not careful enough, such analysis could even suggest the expected distance to the other communicating party. However, HL monitoring is not enough. As the summary in subsection 3.3 indicates, there are more possibilities to force packet drops on the route.

Another possibility to detect the attack provides Netflow or IPFIX data analysis. Consider the following flow statistics of one of our attack attempts (Duration is shown in seconds, Bpp = bytes per packet, which means the average number of bytes in a packet of the flow):

```
Duration Direction                Packets  Bytes  Bpp
3.502    <Attacker> -> <Server>   8467    521056   61
3.502    <Server> -> <Attacker>   1016     79352   78
```

This bi-directional flow has a short duration of 3.5 seconds, yet at least 1000 of packets were transferred in both directions. Moreover, both directions have low average packet size. The low size of packets sent to the server is caused by the attack while mostly acknowledgement packets were sent from the server. When the server replies contain data, the Bpp gets higher. However, since the receiver should acknowledge at least every second segment (RFC 1122), the higher number of segments in one direction of a TCP stream causes that more TCP acknowledgements have to be sent. As TCP acknowledgments are short, the Bpp of the server to attacker direction of the TCP connection is lower in comparison to the Bpp of TCP streams of the same server that are not affected by the attack. This property distinguishes the attack from regular bi-directional TCP flows in which data are transferred mostly in one direction (e.g. file download or upload via HTTP). For example, consider the following HTTP download detected in BUT campus network:

```
Duration Direction                  Packets  Bytes    Bpp
137.117  <Client> -> <Web server>   10311    639122    61
137.117  <Web server> -> <Client>   34604    51.9 M 1498
```

Web file transfers tend to have Bpp of one direction near the maximum allowed on the path while Bpp in the opposite direction is similar to the attack statistics due to TCP acknowledgements. Note, that the packet number sent to the server in the HTTP example is about 20% higher while the flow duration is almost 40-times longer. Therefore, NetFlow data might be used to indicate the attack. However, other attack configurations that employed TCP segments as big as possible had properties very similar to regular data transfers:

```
Duration Direction                  Packets  Bytes    Bpp
0.607    <Attacker> -> <Server>     100     143819  1438
0.607    <Server> -> <Attacker>      14       1084    77
```

As the above mentioned examples show, if the attacker configures the attack with short-sized TCP it may be visible in NetFlow data. However, the last example revealed that the attacker may configure the attack in a way that the NetFlow statistics are similar to some of the regular flows.

5.2 Fake data removal

Whenever the exact scheme of the attack is known, the interceptor can remove the noise from the traffic traces. For example, if the pair (D_{max}, R_{min}) was known, every packet with HL lower than R_{min} is noise and can be dropped. Another example is a topology, where a known router on the path from the IAP to the remote destination drops each packet marked by a hop-by-hop option. Then, every packet containing that option would have been dropped and consequently could be considered to be noise by the analyser.

Sometimes a network trace might be available to an investigator without additional knowledge about the network topology where the trace was captured. We investigated the possibilities of the recovery of the original traffic from the trace in the case of the attack being based on dropping fake segments due to insufficient HL. The fake data removal is based on analysis of HLs of the packets in a trace. We implemented a tool called LNC [5] that filters out fake TCP segments from pcap files. For each sequential number in the trace, LNC searches for a datagram with the highest HL in the trace (see Figure 8). All other TCP segments are filtered out by LNC. The original message can be decoded even by the software that is not able to decode the original message from a trace with the attack.

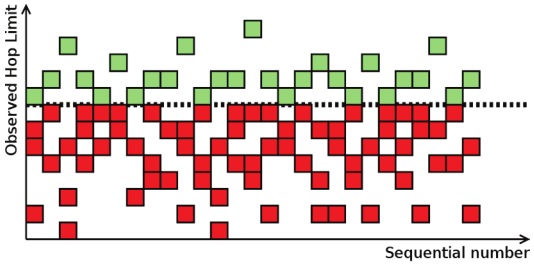

Figure 8: IPv6 datagrams with the highest Hop Limit number carries the original message.

Even this method can be evaded, though. As Shankar and Paxson revealed [15], in case of receiving overlapping TCP segments with distinct content, the behaviour of operating systems differs. The attacker can misuse the knowledge of the operating system running by the other communicating party and send a fake segment with higher HL than the HL of the correct TCP segment. The only condition is that the operating system of the other communicating party decodes the original message from the received sequence of segments. In this case, the analysis of the trace has to be done with the knowledge of the operating systems of all hosts affected by the attack.

In addition, the attacker could confuse the decoding method even without the knowledge of the type of the overlapping segment processing of the other operating system,. If the attacker sends another segment when he or she is certain that the operating system of the other communicating party has already passed the received data to the application reading the corresponding socket, such packet would be intercepted but not processed by the other communicating party. Therefore, the decoding software should also filter out segments that are clearly sent too late regardless on the HL or another property used to drop fake packets.

LNC was successfully tested on pcap files gathered during the experiments with LDP discussed in section 4. The cleaned pcap files were tested with the software that failed to decode the original message from the original pcap files (see subsection 5.3). The repeated tests were successful and the original message was decoded from the pcap files cleaned by LNC.

5.3 Evaluation of decoding software

Decoding software	IPv6 support	Interpretation	Detected anomalies
Wireshark	Yes	First cover message	High number of TCP retransmissions and out-of-order packets in the packet list.
Chaosreader	Yes	Random noise	None
tcpflow	Yes	Last cover message	None
tcptrace	Yes	Last cover message	High number of segments with the same sequential number.

Table 2: Summary of properties of the tested decoding software.

Evaluation of current tools for TCP decoding was another part of the experiments with attack detection. We re-evaluated a few of the software originally tested by Cronin et al. [1]. Additionally to their research, we tested support for the attack in both IPv4 and IPv6. The main result of the study is the disclosure that some software decodes the message from the content of last segments observed with each sequential number. The results of the study are summarised in Table 2.

The decoding software was tested with packet traces that were gathered during the experimental phase with LDP covered in Section 4. None of the software was able to detect the original message. However, an experienced investigator can reveal the attack by noticing the unusually high number of TCP retransmissions in the packet list of Wireshark. Similarly, tcptrace provides a textual output in which it is possible to notice the high number of retransmissions.

5.4 Discussion

There are several methods that provides defence against the attack. As there are dozens of possible configurations of the examined attack, some of them work only for a specific attack configurations whereas others are generic.

In contrast to Cronin et al. [1] who discussed the attack with one-byte-long TCP segments, we are more concerned of attackers using longer TCP segments with specifically crafted cover messages. Subsection 5.1 suggests that such flows have similar NetFlow characteristics to ordinary flows. Moreover, the overhead (see subsection 4.2) of this attack configuration is relatively small. The method based on tracking of number of HL values presented in subsection 5.1 would fail if the fake data had HL of D_{max} and the correct data had HL of R_{min} as load balancing over paths with different number of hops could have caused that two values of HL are visible. However, such attack can still be detected by the method presented in subsection 5.2 as it would detect duplicate segments in the packet trace. Moreover, as pointed out in Section 3, as soon as the attack is revealed, (D_{max}, R_{min}) can be deduced and consequently the original message would be disclosed.

6 Conclusions

In recent years, there has been a vast interest in cover channels in computer networks. This paper elaborated on an already known threat [1, 11] that allows an attacker to confuse security and monitoring software into decoding a cover message instead of the original message that is decoded by the other communicating party even if it is not aware of the attack. The first purpose of the study was to determine

the benefits of the attack for an attacker; however, the ultimate goal was a proposal of detection methods tailored for the attack. This paper has pointed out that the attack has an excessive overhead (over 400-times more data has to be sent in comparison with regular TCP to transfer private message of just 16 KB in the attack configuration discussed by Cronin et al. [1]). Moreover, the study unveiled that the responses of the other communicating party are often enough to leak substantial clues about the attacker's activity. The research of counter-measures indicates that there are several methods that might be useful for the attack detection. Additionally, the paper presented a method for fake segments filtering of pcap files with TCP streams affected by the attack.

7 Acknowledgements

This work is a part of the project VG20102015022 supported by Ministry of the Interior of the Czech Republic. This work was also supported by the research plan MSM0021630528 and BUT project FIT-S-11-1.

References

[1] Cronin, E.; Sherr, M.; Blaze, M.: On the (un)reliability of eavesdropping. In Int. J. Secur. Netw., vol. 3, pp. 103–113, 2008, ISSN 1747-8405.

[2] Department of Defense: Trusted Computer System Evaluation Criteria, TR DoD 5200.28-STD, 1985.

[3] Girling, C. G.: Covert channels in LAN's. In IEEE Transactions on Software Engineering, vol. 13, no. 2, pp. 292–296, 1987.

[4] Kundur, D.; Ahsan, K.: Practical Internet Steganography: Data Hiding in IP. In Texas Wksp. Security of Information Systems, p. 5, 2003.

[5] Hranický, R., Polčák, L.: LDP/LNC Download page. http://www.fit.vutbr.cz/~ipolcak/prods.php?id=255, http://www.fit.vutbr.cz/~ipolcak/prods.php?id=256, 2013.

[6] Lucena, N.; Lewandowski, G.; Chapin, S.: Covert Channels in IPv6. In Privacy Enhancing Technologies, LNCS, vol. 3856, Springer Berlin / Heidelberg, pp. 147–166, 2006, ISBN 978-3-540-34745-3.

[7] Mátray, P., Hága, P., Laki, S., Vattay, G., Csabai, I: On the spatial properties of internet routes, In Computer Networks, Volume 56, Issue 9, pp. 2237-2248, 2012, ISSN 1389-1286.

[8] Mazurczyk, W., Szaga, P., Szczypiorski, K.: Using Transcoding for Hidden Communication in IP Telephony. In CoRR, vol. abs/1111.1250, p. 17, 2011.

[9] Miller, T.: Passive OS Fingerprinting: Details and Techniques. http://www.ouah.org/incosfingerp.htm

[10] Murdoch, S.; Lewis, S.: Embedding Covert Channels into TCP/IP. In Information Hiding, Lecture Notes in Computer Science, vol. 3727, Springer Berlin / Heidelberg, pp. 247–261, 2005, ISBN 978-3-540-29039-1.

[11] Paxson, V.: Bro: a system for detecting network intruders in real-time. In Computer Networks, volume 31, no. 23-24, pp. 2435 – 2463, 1999.

[12] Perkins, M. C.: Hiding out in Plaintext: Covert Messaging with Bitwise Summations. Diploma thesis, Iowa State University, 2005.

[13] Raiciu, C., Paasch, C., Barre, S., Ford, A., Honda, M., Duchene, F., Bonaventure, O, and

Handley, M:. How hard can it be? designing and implementing a deployable multipath TCP. In Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation, USENIX Association, Berkeley, CA, USA, 2012.

[14] Servetto, S.; Vetterli, M.: Communication using phantoms: covert channels in the Internet. In IEEE International Symposium on Information Theory, p. 229, 2001.

[15] Shankar, U.; Paxson, V.: Active mapping: resisting NIDS evasion without altering traffic. In Symposium on Security and Privacy, pp. 44–61, 2003, ISSN 1081-6011.

[16] Zander, S., Armitage, G.: Covert Channels in the IP Time to Live Field. In Australian Telecommunication Networks and Applications Conf., 2006, ISBN 0-97-758610-3.

[17] Zander, S., Armitage, G., and Branch, P.: A survey of covert channels and countermeasures in computer network protocols. In Communications Surveys Tutorials, IEEE, vol. 9, no. 3, pp. 44–57, 2007.

[18] Zander, S., Branch, P., Armitage, G.: Error probability analysis of IP Time To Live covert channels, In International Symposium on Communications and Information Technologies, pp. 562-567, 2007.