Traffic Flow Confidentiality in IPsec: Protocol and Implementation

Csaba Kiraly[1], Simone Teofili[2], Giuseppe Bianchi[2], Renato Lo Cigno[1],
Matteo Nardelli[1], Emanuele Delzeri[2]
1 University of Trento,
{kiraly,locigno,matteo.nardelli}@dit.unitn.it
2 University of Rome Tor Vergata,
{ giuseppe.bianchi,simone.teofili,emanuele.delzeri}@uniroma2.it

Traffic Flow Confidentiality (TFC) mechanisms are techniques devised to hide/masquerade the traffic pattern to prevent statistical traffic analysis attacks.. Their inclusion in widespread security protocols, in conjunction with the ability for deployers to flexibly control their operation, might boost their adoption and improve privacy of future networks. This paper describes a TFC protocol integrated, as a sub-layer, in the IPsec Encapsulated Security Payload (ESP) protocol. A Linux-based implementation has been developed, supporting a variety of per-packet treatments (padding, fragmentation, dummy packet generation, and artificial alteration of the packet forwarding delay), in an easily combinable manner. Experimental results are reported to demonstrate the flexibility and the effectiveness of the TFC implementation.

Introduction

Extensive literature work demonstrates that the traffic pattern generated in a communication carries plenty of information, which can be gathered through specially devised "statistical traffic analysis attacks". These attacks operate irrespective of the deployed encryption means, and allow to extract, from the statistical analysis of the generated packet sizes and of their inter-arrival times, valuable confidential information such as the employed applications [1], the application layer protocols [2], the physical devices used [3], or the web page accessed [4,5]. To perform these attacks, a signature for the protocol or the web site to be recognized is typically precomputed as a set of statistical parameters describing packet size and/or packet inter-

This work was supported by the EU IST 6th Framework Program project Discreet, IST No. 027679

arrival time distributions. Flow classification can be performed by matching the actual statistics with the pre-stored signatures. Quite interestingly, accurate flow classification may be obtained even by looking only at its very first packets [2,6].

Statistical traffic analysis attacks have been also employed for the purpose of breaching security (such as for gathering password transmitted over encrypted sessions [7,8]), and for performing passive [9] or active [10] attacks to anonymization (Mix) networks, aimed at uncovering the identity of the communicating parties. To duly protect the privacy of the users, "Traffic Flow Confidentiality" (TFC) mechanisms devised to alter or mask the statistical characteristics of the traffic patterns are necessary.

The contribution of this paper is twofold. First, we propose a TFC approach embedded as a sub-layer (and a separate, self-contained, TFC protocol) of IPsec. We believe that the inclusion of TFC mechanisms in existing and widely deployed standards may significantly improve their adoption. Second, our approach is not bound to provide a “specific” traffic masking pattern, but rather aims at providing a flexible platform, endowed with a set of packet treatment primitives (including packet padding, fragmentation[2], dummy packet generation, and artificial alteration of the packet forwarding delay, upon which the system deployers may easily configure the traffic masking patterns they deem more appropriate for achieving a given privacy/performance trade-off.

To the best of our knowledge, our approach differs from most of the existing literature in this field as i) it is deployed as a separate module, rather than integrated in a specific Mix-like solution [11,12,13], and ii) it is developed as a flexible suite of easily composable tools rather than as a pre-programmed specific traffic masking technique (for instance, the frequently employed traffic CBR-ization, i.e. transforming traffic into continuous bit rate pattern composed of packets with maximum size). The TFC protocol and the related packet treatment tools are implemented in Linux as part of the packet transformation framework introduced in the 2.6 kernel.

TFC sub-layer design and implementation

To overcome the drawbacks of the (limited) TFC mechanisms specified in the latest version of the IPsec Encapsulated Security Payload specification [14], we have designed TFC as a separate sub-layer, thus maintaining backward compatibility with traditional IPsec implementations. The TFC sub-layer is implemented through a neat separation between i) the TFC control logic, namely the algorithms devised to transform a traffic pattern into another one, ii) their protocol support, accomplished through the specification of a TFC header, and iii) the set of basic mechanisms

[2] Indeed fragmentation is a technique traditionally neglected as a tool for traffic masking, as most of the approaches proposed in the past in fact are based on traffic padding and/or dummy packet generation. Conversely, we believe that fragmentation is a highly effective tool as i) it has a very low overhead if compared with padding or dummy packet generation, and ii) it may be selectively employed on the packets, such as the very first in the flow [1,5,6], which are found to provide most of the information useful for the classification algorithms.

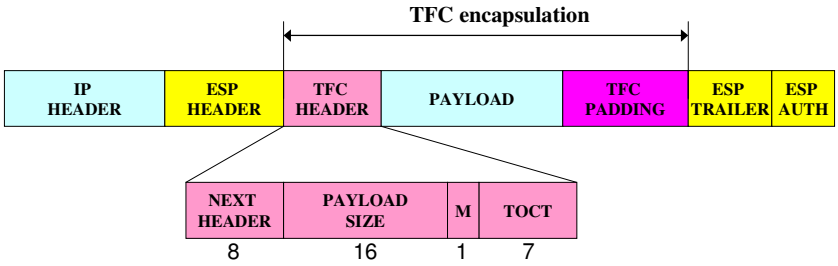

Figure 1: TFC header format

employed by this control logic to modify characteristics at the packet level. Basic TFC mechanisms can be conveniently categorized as follows:

a) Packet forming: devised to alter the packet size; they include packet padding, packet fragmentation, and packet aggregation (multiplexing);
b) Dummy packet management: devised to generate and discard dummy packets, in order to alter the traffic pattern;
c) Packet timing: devised to alter the forwarding latency of packets adding extra per-packet delay.

Our TFC implementation is developed inside the Linux Kernel 2.6, and leverages the XFRM framework [15] deployed for integrating the TFC processing in the IP/IPsec networking stack[3]. Being developed as a sub-layer, the TFC protocol takes advantage of all the existing ESP functionalities (confidentiality, data integrity and authentication, as well as Security Association and policy management).

Packet Forming

To support the three packet forming mechanisms, we have designed an IPsec Header Extension, the TFC Header (Figure 1), for the encapsulation of the datagram. The TFC header is internal to the IPsec ESP payload, and it conveys the necessary information to restore the original packet (padding removal, reassembly, demultiplexing). A next header code should be reserved in the ESP trailer to indicate to the receiver that the protocol contained in the ESP payload is TFC: in our experimentation we used the value 253, reserved by IANA for experimentation and testing. The next header field in the TFC header identifies the protocol carried in the payload.

Packet padding is the traditional (albeit naïve[4]) approach to alter the packet size statistics. The TFC header manages padding simply by explicitly carrying the packet

[3] In additional details, TFC, just like other IPsec security services, is managed through Security Associations, i.e. we have developed a new Security Association type for TFC similarly as what done for the ESP and AH SAa. This allows controlling TFC through standard security policies included in the IPsec Security Policy Database (SPD). We use the Netlink interface and XFRM SA and SP databases to implement these features. For additional implementation details, please refer to the Discreet Project Deliverable D3102 – available upon request from the authors.

[4] Indeed, statistics taken on real IP flows show a high variance in the packet size, and thus padding to the maximum possible size introduces a massive overhead. Moreover, studies such as [1,5,6] seem to imply that most of the traffic classification mechanisms use the size information contained in the first few

payload size information in a dedicated field in the TPC header. We point out that this allows overcoming the significant drawback of the "implicit" padding function proposed in the current version of IPsec, which impedes its usage for inner protocols which do not provide an explicit indication of the payload size (e.g. TCP).

Packet fragmentation allows splitting a large packet into smaller packets, and hence avoids the need to add a very large amount of per-packet overhead in the presence of many small packets and a few large ones. We support fragmentation by reusing the IPv6 fragmentation header inside the TFC header. Fragments are reassembled at the end of the overlay link, before the packet is handed to upper protocols.

Packet aggregation allows multiplexing packets into a bigger datagram, thus increasing the size of the packet through useful information and not through wasted padding bytes. Packet multiplexing is supported by introducing a flag in the TFC header. If this flag is set, after the defined length of the payload, another TFC header and payload follows instead of padding. With this mechanism, several payloads (even fragments) can be transferred in one datagram.

Finally, the TFC header is also exploited to deliver a field, called TOCT (Type of Confidentiality Treatment) which enables to carry information about the type of treatment the packet may be subjected to, when multiple IPsec links are used in a multi-hop fashion, and especially for building IPsec-based Mix Networks. For reasons of space this operation is not described in this paper.

Dummy Packets

The use of dummy (artificially generated) packets is frequently referred to as "traffic padding". It allows filling traffic gaps and avoids disclosing inactivity periods (i.e., provide unobservability). Moreover, dummy packets are a powerful instrument to alter the traffic pattern statistics, especially when real packets, due to quality of service constraints, cannot be delayed to an extent that allows proper reshaping of the traffic profile. Finally, dummy packet generation is a technique extensively employed in Mix Networks to counteract correlation and several types of active attacks (e.g., the "n-1" attack).

Protocol support for dummy packet management is straightforward, setting the next header code to "dummy". In our implementation the value 59 is employed, as this value has been standardized in the IPsec ESP specification. For a more homogeneous implementation of the TFC tools we use the dummy packet value 59 inside the inner TFC header, rather than inside the ESP trailer.

Packet Timing

Information extracted from packet inter-arrival times is a usual source for statistical traffic analysis methods [1,5,10]. To counter these attacks, scheduling algorithms (externally programmed in the "control logic" module discussed next - see Figure 2) should alter the forwarding time of packets.

---

packets of a session which are those that convey the protocol fingerprint, making it less important to `heavily' pad subsequent traffic.

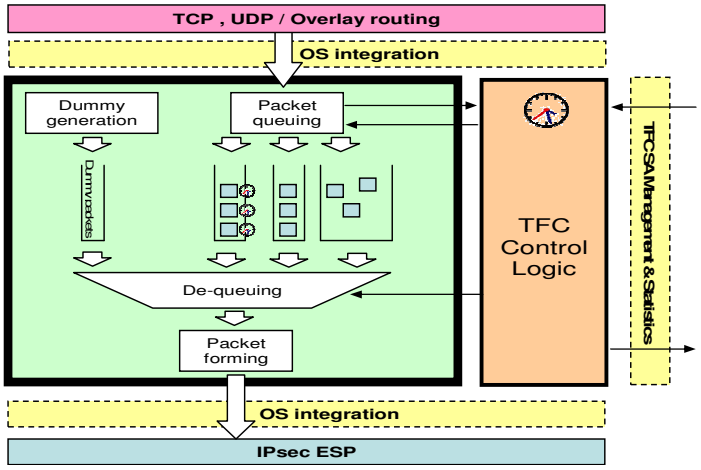

Figure 2: TFC module architecture

Our implementation supports two methods to alter the packet delays. A first "event-driven" method allows a packet, upon its arrival (top arrow in Figure 2), to be associated with a specific, possibly packet-dependent, delay. The packet is then delivered inside the TFC module queues[5] only when the associated timer elapses. A second "timer-driven" method allows de-queuing packets stored inside the TFC module queues at scheduling instants computed according to an algorithm controlled by the Control Logic module. Note that in the case queues are empty, a dummy packet will be de-queued from the dummy packet buffer and delivered.

Implementation complexity is delegated to the control logic implementation, as packet delivery is internally accomplished through appropriate setting of standard Linux timers which drive the invocation of a de-queuing primitive. Trivial methods, such as fixed or random packet clocking, may be easily replaced by adaptive clocking algorithms which explicitly take into account the status of the queues and the related congestion level (although, to the date of writing, the effectiveness of such adaptive approaches in terms of performance/privacy gains and trade-offs is still to be assessed).

Control Logic

The "intelligence" of the system is implemented in a separate "control logic" module, which can combine the TFC basic mechanisms arbitrarily. For the time being, in order to provide flexibility, we have implemented batching, CBR (Continuous Bit rate), random padding, and random delay algorithms. The ease of such implementation shows the flexibility of the proposed framework, as well as its amenability to implement new algorithms.

We believe that a significant asset of our work is the accomplished decoupling between the algorithmic logic devised to masquerade/shape the traffic pattern, and its

[5] Multiple queues may be internally deployed to differentiate packets incoming from different streams, where a stream may be defined either as a classification rule on the incoming packets, and/or in dependence of the different TFC Security Association mapped over the same IPsec ESP Security Association (for reasons of space, details are omitted).

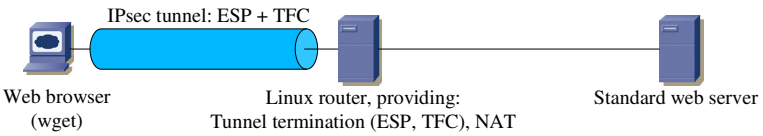

Figure 3: Test setup

underlying implementation as a set of basic tools. This decoupling allows the network deployer to configure (or eventually directly program in the control logic) the most appropriate TFC algorithm suited for its purposes. For instance, in an anonymization network such as Tarzan [13] (which may be built on top of IPsec tunnels extended with the proposed TFC functionalities) the user activity is "well known" but it is necessary to modify the flow fingerprint in order to avoid correlation attack. Conversely, in a point to point connection (e.g., user to proxy) the main goal is to avoid protocol or web site fingerprinting, which may be more adequately countered with different masking algorithms.

Demonstration

To demonstrate the flexibility of the TFC protocol, we set up a test environment (Figure 3) similar to the one used in [5] and [16]. We analyze how the information content of fingerprints can be reduced, while performance degradation remains limited.

A client downloads web pages from a normal, unmodified web server (we use http://www.ist-discreet.org and https://www.prime-project.eu/ for our experiments). To protect against traffic classification attacks, the client creates an IPsec protected tunnel to an exit node. Two ESP SAs and two TFC SAs are set up to cover the traffic of the bi-directional communication. Throughout our tests, we were using symmetric configuration for the SAs, but of course parameters (as well as the control logic) used in the two directions may differ.

The client downloads a full web page, with all of its inline images, using the *wget* program. We record traces of the packets traveling on the tunnel. We do not consider crypto analysis, therefore, according to our model, the only useful information for the attacker are: the packet length, packet time (and packet inter-arrival time) and the packet direction.

Figure 4 shows typical unprotected and protected packet size and packet inter-arrival time fingerprints, based on packet cardinality. It can easily be seen how information is removed from such fingerprint: padding and fragmentation makes packet size useless, while timing removes large part of the information contained in packet inter-arrival times.

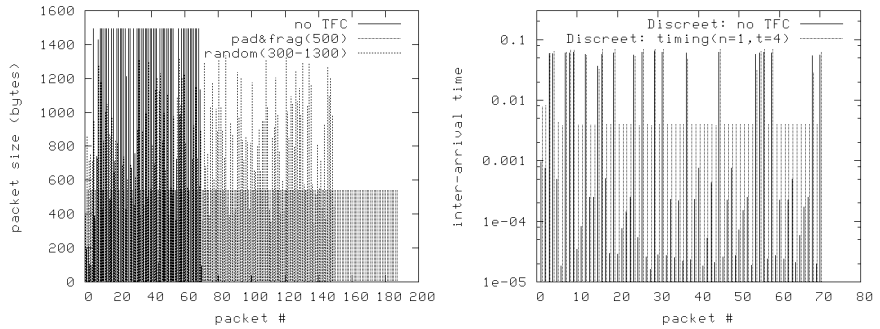

Figure 4: Packet size and inter-arrival time fingerprints, obtained by downloading the Discreet page with different TFC control logics. Control logics used are: padding or fragmentation to a fixed size; padding or fragmentation to a random size; timing of packet transmission to release at most one packet in timeslots of 4ms

How much protection this gives? Figure 5 shows the answer: packet counting and packet number based algorithms (such as [2,5,16]) can be fooled without randomization and without dummy traffic. However, looking at the figure, it can easily be seen that (using the same browser application) the traffic pattern mapped into the "cumulative length" - "elapsed time" space remains characteristic of the site.

The figure also shows results achieved with dummy packets using a CBR control logic. We should emphasize that here we show only the simplest use of dummy packets. We are investigating other dummy strategies as well as the effect of randomization.

Figure 5 shows the performance drawback of each control logic as well. The endpoint of each line shows the overhead in user perceived page download time and in traffic amount. For example, without TFC, the Discreet page downloads in 1.3 seconds and generates 88 Kbytes of traffic. The same download with CBR TFC takes 4.7 seconds and 130 KBytes (to improve readably, we did not report the whole curve for CBR TFC for the Prime site: it took 15.5 seconds and 422 Kbytes). It can be seen that the overhead remains in reasonable limits for each of the algorithms.

Conclusions

As shown by many recent papers, statistical traffic analysis techniques provide good results based on packet size and packet inter-arrival time statistics. We have designed the TFC IPsec security service to protect against such attack, discussed its implementation and demonstrated its effectiveness.

Our approach gives a further level of protection, as masking performed at the IPsec layer impedes reconstruction of application-layer message size. It also introduces fragmentation, aggregation and packet inter-arrival time variation to balance the protection-performance tradeoff.

Our future work includes the evaluation of different (deterministic and stochastic, traffic independent and adaptive) control logics.

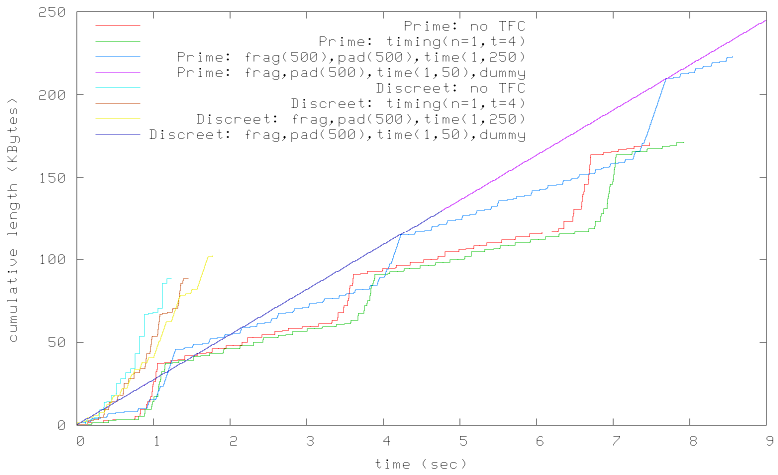

Figure 5: Download of different web pages using different control logics

References

[1] L. Bernaille, R. Teixeira, and K. Salamatian, "Early Application Identification", Proceedings of The 2nd ADETTI/ISCTE CoNEXT Conference, Portugal, 2006.

[2] M. Crotti, F. Gringoli, P. Pelosato, L. Salgarelli, "A statistical approach to IP-level classification of network traffic", IEEE ICC 2006, 11-15 Jun. 2006.

[3] T. Kohno, A. Broido, K. C. Claffy. "Remote physical device fingerprinting", in IEEE Symposium on Security and Privacy, pp. 211–225. IEEE Computer Society, 2005.

[4] A. Hintz, "Fingerprinting Websites Using Traffic Analysis", Privacy Enhancing Technologies, PET 2002, S. Francisco, USA, April 2002

[5] G. D. Bissias, M. Liberatore, D. Jensen, B. N. Levine, "Privacy Vulnerabilities in Encrypted HTTP Streams", PET 2005, Cavtat, Croatia, May 30-June 1, 2005.

[6] L. Bernaille, R. Teixeira, "Early Recognition of Encrypted Application" Proc. PAM, April 2007.

[7] D. X. Song, D. Wagner, X. Tian, "Timing analysis of keystrokes and timing attacks on SSH", 10th USENIX Security Symposium, 2001.

[8] B. Canvel, A. Hiltgen, S.Vaudenay, M. Vuagnoux, "Password Interception in a SSL/TLS Channel", CRYPTO2003, Aug 2003, Santa Barbara, USA

[9] Y. Zhu, X. Fu, B. Graham, R. Bettati, W. Zhao "On Flow Correlation Attacks and Countermeasures in Mix Networks", PET 2004, May 2004

[10] X.Wang, S. Chen, S. Jajodia, "Tracking anonymous peer-to-peer VoIP calls on the internet", ACM Conf. on Computer and Communications Security, November 2005.

[11] G. Danezis, R. Dingledine, N. Mathewson, "Mixminion: Design of a Type III Anonymous Remailer Protocol", 2003 IEEE Symp. on Security and Privacy, May 2003.

[12] R. Dingledine N. Mathewson, P. Syverson, "Tor: The Second-Generation Onion Router", 13th USENIX Security Symp. Aug 2004.

[13] M. J. Freedman, R. Morris, "Tarzan: a Peer-to-Peer Anonymizing Network Layer", ACM Conf. on Computer and Communications Security, Washington, DC, November 2002.

[14] S. Kent, "IP Encapsulating Security Payload (ESP)", RFC 4303, December 2005.

[15] M. Kanda, K. Miyazawa, H. Esaki, "USAGI IPv6 IPsec development for Linux", Int. Symp. Applications and the Internet Workshops (SAINT) 2004. pp. 159-163, Jan. 2004.

[16] M. Liberatore, B. N. Levine, "Inferring the Source of Encrypted HTTP Connections", CCS2006, October 2006