Cross-Path Inference Attacks on Multipath TCP

M. Zubair Shafiq‡, Franck Le†, Mudhakar Srivatsa†, Alex X. Liu‡
‡ Michigan State University, East Lansing, MI, USA.
† IBM T. J. Watson Research Center, Yorktown Heights, NY, USA.
Email: {shafiqmu,alexliu}@cse.msu.edu, {fle,msrivats}@us.ibm.com

ABSTRACT

Multipath TCP (MPTCP) allows the concurrent use of multiple paths between two end points, and as such holds great promise for improving application performance. However, in this paper, we report a newly discovered class of attacks on MPTCP that may jeopardize and hamper its wide-scale adoption. The attacks stem from the interdependence between the multiple subflows in an MPTCP connection. MPTCP congestion control algorithms are designed to achieve resource pooling and fairness with single-path TCP users at shared bottlenecks. Therefore, multiple MPTCP subflows are inherently coupled with each other, resulting in potential side-channels that can be exploited to infer cross-path properties. In particular, an ISP monitoring one or more paths used by an MPTCP connection can infer sensitive and proprietary information (e.g., level of network congestion, end-to-end TCP throughput, packet loss, network delay) about its competitors. Since the side-channel information enabled by the coupling among the subflows in an MPTCP connection results directly from the design goals of MPTCP congestion control algorithms, it is not obvious how to circumvent this attack easily. We believe our findings provide insights that can be used to guide future security-related research on MPTCP and other similar multipath extensions.

Categories and Subject Descriptors

C.2.5 [**Computer-Communication Networks**]: Local and Wide-Area Networks—*Internet (e.g., TCP/IP)*

General Terms

Security, Experimentation

Keywords

Multipath TCP, Congestion Control

Hotnets '13, November 21–22, 2013, College Park, MD, USA.

1. INTRODUCTION

Multipath TCP (MPTCP) enables two endpoints to simultaneously use multiple paths available between them. This new capability can improve application performance, especially considering mobile devices are increasingly becoming multihomed (e.g., cellular, Wi-Fi). This feature has consequently sparked a lot of interest and excitement both in industry [6, 13] and academia [14, 12, 9, 10]. Recently, researchers have proposed numerous theoretical frameworks and solutions to better understand and take advantage of this new capability while satisfying MPTCP's requirements. MPTCP has three main design objectives [6]:

1. *Improve throughput:* It should give a throughput that is at least as high as that of a single-path TCP connection on the best available path. The goal of this objective is to incentivize MPTCP deployment.

2. *Do no harm:* It should not take up more capacity on its different paths than if it were a single-path TCP connection using only one of these paths. This is to ensure that MPTCP will not degrade the performance of applications that use single-path TCP at shared bottlenecks.

3. *Balance congestion:* It should move traffic off from the congested paths, subject to satisfying the first two objectives. The purpose of this design objective is to achieve resource pooling, i.e., MPCTP connections should send more traffic on lesser congested paths to alleviate overall congestion in the network.

Recent research efforts [14, 9, 10] have focused on designing and improving congestion controllers to effectively satisfy these design goals. However, these design goals create a tight coupling between multiple MPTCP subflows that can be exploited as side-channels to launch a *new* class of attacks.

In this paper, we present a new class of attacks on MPTCP, that we call *cross-path* attacks. These attacks enable a party monitoring one (or more) paths used by

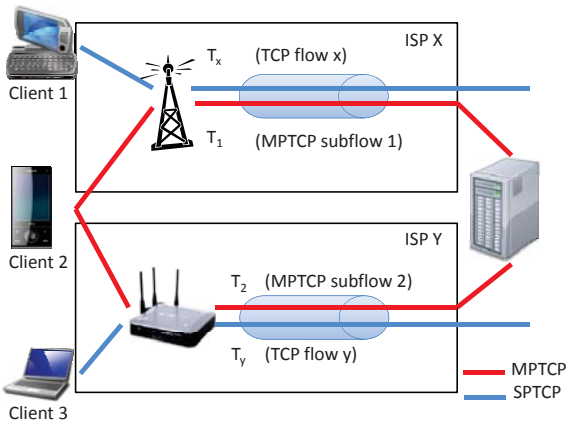


Figure 1: A scenario illustrating competing single-path TCP (SPTCP) and MPTCP connections

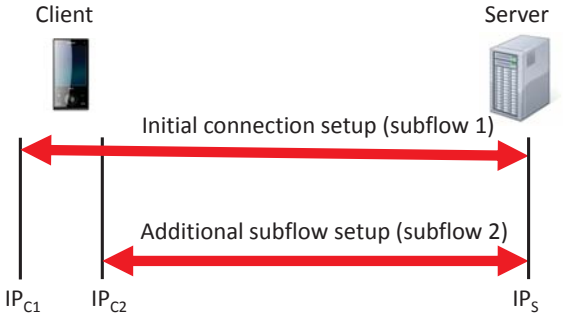


Figure 2: An example MPTCP usage scenario

MPTCP subflow(s) to infer properties of other paths. In particular, such attacks can allow ISPs to infer sensitive and proprietary information (e.g., level of congestion, throughput, packet loss, round trip time) about their competitors [4]. For the scenario illustrated in Figure 1, the coupling among MPTCP subflows 1 and 2 can enable ISP X to infer cross-path network performance metrics for ISP Y (i.e., properties of single-path TCP flows traversing ISP Y), and vice-versa. To the best of the authors' knowledge, we are the first to describe this new type of attack. Cross-path attacks were not possible in regular TCP: although an *on-path* attacker could eavesdrop and launch session hijacking or man-in-the-middle attacks, an *off-path* attacker could not directly infer any property about the connection in regular TCP.

Using a testbed of Linux nodes supporting the MPTCP Linked-Increases Algorithm (LIA) congestion controller [11], standardized by the IETF, we empirically demonstrate the feasibility of such attacks. Our experiments show that an attacker can infer the throughput that a single path TCP will obtain in other networks, with up to 90% accuracy in a measurement interval of less than 8 minutes, 3 minutes, and 8 minutes respectively for links with the characteristics of wired, Wi-Fi, and cellular networks.

We highlight that this class of attacks is not specific to a particular congestion control algorithm (e.g., LIA [11]), but derives from the design goals of MPTCP, and as such applies to all congestion control algorithms designed to satisfy them, including more recent proposals [9, 10]. Also, interestingly, recent congestion control proposals developed to more effectively satisfy the MPTCP design objectives work in favor of attackers, allowing them to more quickly and accurately infer the throughput of other paths. For example, with the new Opportunistic Linked-Increases Algorithm (OLIA) congestion control algorithm [9], an attacker can conduct cross-path throughput inference up to 50% faster than with the initial LIA congestion control [11]. Finally, although our preliminary empirical validations focus on inferring the throughput of a single path TCP flow in other networks, we believe that the same principle can be used to infer other key metrics including connection delay and packet loss.

The rest of this paper is organized as follows. We provide a brief overview of MPTCP in Section 2. In Section 3, we introduce the cross-path inference attack on MPTCP. Next, we present some preliminary results in Section 4 before discussing various technical challenges and potential countermeasures in Section 5. Section 6 provides a summary of the prior work on MPTCP security. Finally, Section 7 concludes the paper.

2. BACKGROUND

We provide a brief overview of MPTCP. More details about MPTCP operation can be found in RFC 6824 [6]. To facilitate its deployment and compatibility with middleboxes, MPTCP is defined as a set of extensions to TCP. Different MPTCP parameters are signaled through TCP options. An MPTCP connection consists of one or multiple TCP subflows.

Figure 2 illustrates the MPTCP connection setup steps for the scenario of Figure 1. The MPTCP connection (subflow 1) is first established between IP_{C1} and IP_S on the client and server, respectively. MPTCP capability is negotiated between the client and server using the `MP_CAPABLE` TCP option during the three-way handshake in the initial connection setup. During the initial MPTCP connection setup phase, the end points may also advertise the presence of multiple addresses at either hosts. This allows the establishment of additional MPTCP subflows at later times. In this scenario, the client advertises the address IP_{C2} to the server using the `ADD_ADDR` TCP option on the existing subflow

1 and then initiates a new subflow 2 between IP_{C2} and IP_S using the `MP_JOIN` TCP option.

MPTCP runs on top of TCP subflows, and uses a 64-bit data sequence number (DSN) for all the data sent across multiple subflows. Although each TCP subflow has its own 32-bit sequence number, the DSN enables data to be retransmitted on different subflows in the event of failure. MPTCP also signals connection-level acknowledgements (i.e., Data ACK) to implement flow control. Both the MPTCP DSN and Data ACK are sent as Data Sequence Signal (DSS) TCP option.

3. MPTCP INFERENCE ATTACK

We present the main idea behind the MPTCP inference attack. To simplify the presentation, we assume the scenario of Figure 1 where the MPTCP connection simply comprises of two subflows. Let T, T_1, and T_2 respectively denote the total throughput of the MPTCP connection, the throughput of MPTCP subflow passing through ISP X, the throughput of MPTCP subflow passing through ISP Y. Also let T_x and T_y denote the throughput of single-path TCP connections passing through ISPs A and B, respectively. We address the cases of MPTCP connections with more than two subflows later in Section 5.

We go back to the first two MPTCP design objectives (see Section 1), and look at their implications on the MPTCP connection and its subflows:

1. *Improve throughput:* MPTCP throughput should be at least as high as that of a single-path TCP connection on the best available path.

$$T \geq max(T_x, T_y) \tag{1}$$

2. *Do no harm:* MPTCP should not take up more capacity on its different paths than if it were a single-path TCP connection using only one of these paths. This is to ensure that MPTCP will not degrade the performance of applications that use single-path TCP at shared bottleneck links.

$$T \leq max(T_x, T_y) \tag{2a}$$
$$T_1 \leq T_x \tag{2b}$$
$$T_2 \leq T_y \tag{2c}$$

From equations (1) and (2a), we derive

$$T = max(T_x, T_y) \tag{3}$$

In addition, an attacker in ISP X can directly observe T and T_x. T can be derived through the MPTCP DSN and MPTCP Data ACKs, carried in subflow 1. T_x can be observed through the TCP sequence numbers and TCP ACKs of the single-path TCP flow X. As such, we distinguish two cases:

1. $T > T_x$*:* From equation (3), we have

$$T_y^* = T \tag{4}$$

(T_y^* represents an estimator for T_y. More generally, we note T_i^* an estimator for T_i). From the MPTCP throughput T, one can infer T_y. In other words, ISP X can passively infer the throughput of a single-path TCP in a different ISP, e.g., ISP Y.

2. $T \leq T_x$*:* From (2c) and (3), $T_2 \leq T_y \leq T_x$.

The attacker can estimate T_2 from the fact that $T = T_1 + T_2$. By observing T and T_1, the throughputs of the MPTCP connection and the MPTCP subflow 1, $T_2^* = T - T_1$. The attacker can therefore infer an upper bound T_x and a lower bound T_2^* for T_y^*. By throttling T_x, the attacker can gradually tighten the bounds. When $T > T_x$, revert to the former case, where the attacker can directly infer T_y using (4).

We observe that the attack still applies for the more general case of $T = f(T_x, T_y)$. For equation (3), we have $f() = max()$. However, other congestion control algorithms may have different design objectives [14], resulting in different definitions of $f()$, but still allow an attacker to derive key information about other networks.

4. PRELIMINARY RESULTS

We present preliminary results that demonstrate the feasibility of the MPTCP inference attack. Our experimental testbed aims at replicating a topology similar to that of Figure 1. It consists of four nodes, two of them (node 1, node 4) running the MPTCP Linux kernel implementation [12], and configured such that node 1 has two paths (path A: 1–2–4, path B: 1–3–4) to node 4. We experimented with the Linked-Increases Algorithm (LIA) congestion controller [11, 14], standardized by the IETF, and a more recent Opportunistic Linked-Increases Algorithm (OLIA) congestion controller [9]. In order to modify the characteristics (delay and loss rate) of the two paths, we apply the traffic control (a.k.a. *tc*) tool in the Linux kernel, at nodes 2 and 3.

The goal of the attack is for a node (e.g., node 2) to infer the throughput that a single-path TCP flow would achieve on the unmonitored path (e.g., 1–3–4). Consequently, we have nodes 1 and 4 running MPTCP with two subflows over the paths A and B, and competing single TCP flows on these two paths. Path A is throttled to meet the conditions of Section 3. In particular, we have $T > T_x$. Figure 3 confirms that T can indeed be used to estimate T_y when $T > T_x$.

To systematically quantify the accuracy of T in estimating T_y, we analyze the estimation error as a function of measurement intervals (denoted by τ) for paths with

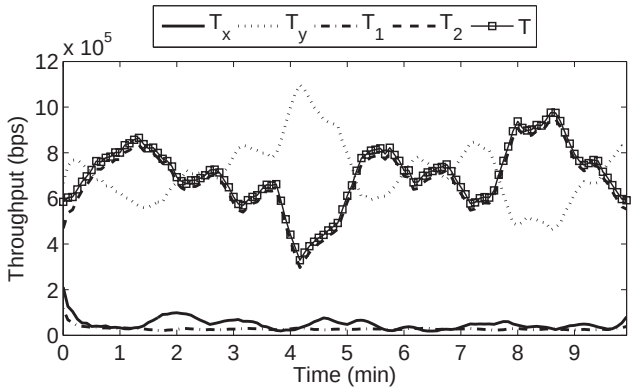


Figure 3: Single-path TCP and MPTCP (OLIA congestion controller) connections compete with each other on paths A and B, which are rate-limited to 1.544 Mbps. Path B is configured with 10 ms delay and 0.01% loss rate, while path A is configured with 250 ms delay and 5% loss rate.

the characteristics of wired, Wi-Fi, and cellular networks. Path B is configured with 10 millisecond delay and 0.01% loss rate to simulate wired networks, 10 millisecond delay and 3% loss rate for Wi-Fi networks, and 100 millisecond delay and 1% loss rate for cellular networks. Path A is configured with 250 millisecond delay and 5% loss rate. Figures 4 and 5 plot the estimation error curves for LIA and OLIA congestion controllers. The error bars show mean and standard deviation of the estimation error for 1000 independent experimental runs. For all link types, we observe that both mean and standard deviation of the estimation error decrease as τ (size of measurement interval) increases. Specifically, the proposed inference attack achieves up to 90% accuracy in a measurement interval of less than 8 minutes, 3 minutes, and 8 minutes respectively for links with the characteristics of wired, Wi-Fi, and cellular links with the LIA congestion controller. Furthermore, it achieves up to 90% accuracy in a measurement interval of less than 7 minutes, 2 minutes, and 4 minutes respectively for links with the characteristics of wired, Wi-Fi, and cellular links with the OLIA congestion controller. Comparing the inference results for LIA and OLIA, we observe that better estimation accuracy is achieved for OLIA as compared to LIA. This difference can be attributed to the fact that OLIA provides at least similar or better responsiveness than LIA. Consequently, T adjusts to changing path conditions (and hence changing T_y) quicker for OLIA as compared to LIA. Comparing the inference time for different paths, the faster inference for Wi-Fi as compared to wired and cellular paths is likely due to Wi-Fi's higher loss rate, which causes connections to reach their "steady state" sooner than when losses are more sporadic.

These results demonstrate that an attacker can exploit coupling among MPTCP subflows to infer proprietary information such as end-to-end throughput of unmonitored paths. It is noteworthy that the attack is successful for different MPTCP congestion control algorithms and paths with diverse characteristics. Next, we discuss how to further build on the promise of these preliminary results to not only scale this particular attack but also construct other similar cross-path inference attacks on MPTCP.

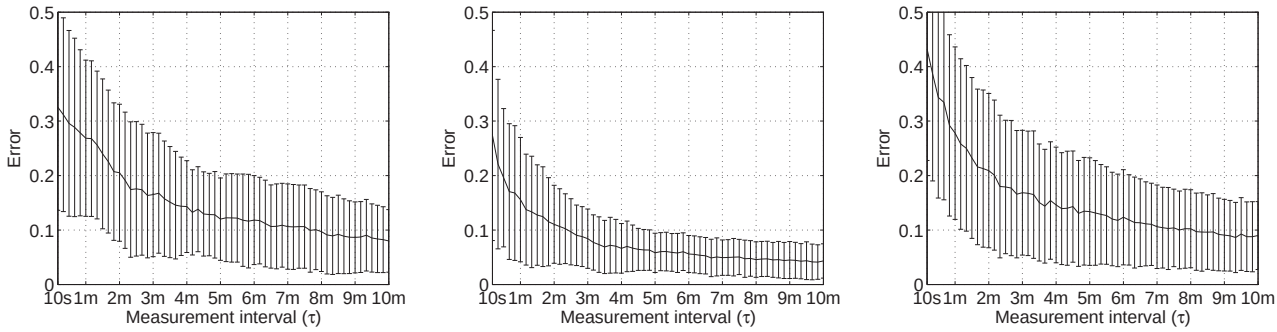
5. DISCUSSION AND FUTURE WORK

We discuss the technical challenges for implementing the proposed attack at scale in real-life settings and potential countermeasures.

Time to inference. Our results showed that the estimation error significantly decreases as the size of measurement interval is increased. For instance, achieving more than 90% estimation accuracy for links with the characteristics of Wi-Fi networks required measurement up to 2-3 minutes. Thus, an attacker would need to monitor an MPTCP subflow that lasts up to 2-3 minutes to achieve good accuracy for such links. The size of measurement interval to achieve 90% estimation accuracy further increases up to 8 minutes for links with the characteristics of wired and cellular networks for LIA congestion controller. In real-life settings, an attacker may not always be able to observe such long-lived MPTCP flows. We will explore ways for an attacker to possibly more quickly and effectively carry cross-path attacks: for example, by simultaneously monitoring multiple MPTCP flows, can an attacker reduce the inference time and improve the estimation accuracy?

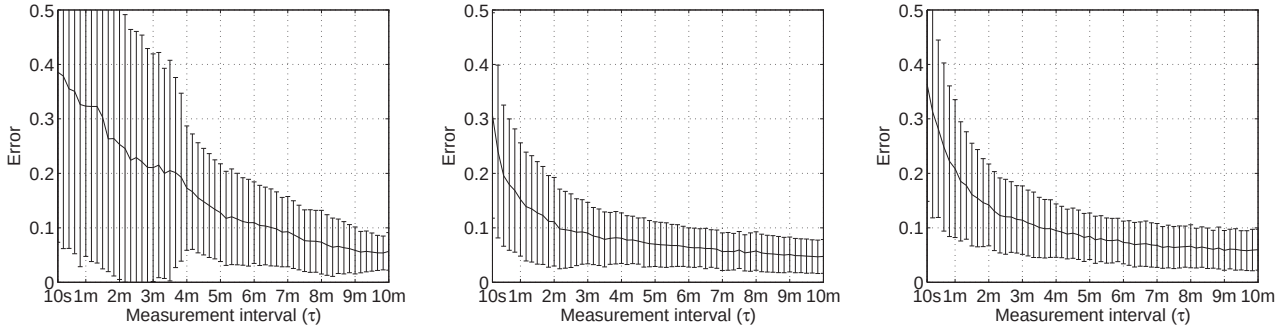
Level of throttling. For our experiments, we imposed a fixed 250 millisecond delay and 5% loss rate on the throttled link. These delay and loss rate values were chosen such that the throttled link's performance is significantly worse than the normal link operation. However, we plan to conduct a systematic analysis to precisely quantify the level of throttling an attacker must enforce to infer properties of the other paths with good estimation accuracy. Ideally, an attacker should use throttling as minimally as possible to avoid degrading the performance of MPTCP connections.

Means to throttle. In our experiments, we throttled a link on one of the two paths in our testbed to force MPTCP connections to the case where we could infer the cross-path throughput. However, if the link that an attacker is monitoring is serving hundreds or even thousands of other flows (e.g., for edge, enterprise, or backbone links), throttling the link will also adversely impact the performance of these flows. Thus, the at-



(a) Wired (b) Wi-Fi (c) Cellular

Figure 4: Throughput estimation error for LIA congestion controller



(a) Wired (b) Wi-Fi (c) Cellular

Figure 5: Throughput estimation error for OLIA congestion controller

tacker may not desire to degrade the performance of all flows passing through the monitored link because it can leave a large scale footprint of the attacker's operation that can detected and even traced back to the attacker. To address this challenge, an attacker can instead throttle individual MPTCP subflows, using standard QoS mechanisms already implemented in routers [3, 5], so that other flows on the same path are unaffected.

Handling MPTCP connections with more than two subflows. Throughout this paper, our analysis and experiments were conducted for MPTCP connections with two subflows. However, MPTCP supports more than two subflows when more paths are available; although, we envision the most common MPTCP scenarios will employ two subflows, at least in the near future (e.g., mobile devices with cellular and Wi-Fi links connected to regular web servers). It is noteworthy that the proposed cross-path throughput inference attack can be directly extended to MPTCP connections with an arbitrary number of subflows (say N) by monitoring $N-1$ links on distinct subflows. However, is it unlikely for an attacker to access links on $N-1$ distinct paths, especially when N is large. Towards this end, an alternative approach is to supplement an MPTCP connection that has more than two subflows with MPTCP connections that have two subflows such that they cover at least one of the multiple subflows of the former MPTCP connection. Let $X = \{p_1, p_2, ..., p_N\}$ denote an arbitrary MPTCP connection X with N subflows on paths $p_1, p_2, ..., p_N$. Suppose $Y = \{p_1, p_2, p_3\}$ and $Z = \{p_1, p_2\}$, and an attacker is monitoring the subflow on path p_1. In this case, it is possible to conduct joint throughput inference for both connections to infer single-path TCP throughput on path p_3 because path p_2 is shared by Y and Z. Our preliminary results give us a reason to be hopeful that we can use this approach in our ongoing work to handle MPTCP connections with more than two subflows without monitoring as many as $N-1$ distinct paths.

Congestion control algorithm identification. As demonstrated in Section 4, the proposed inference attack successfully works for different congestion control algorithms. Recall that our main requirement for this attack to work was that the underlying congestion control algorithm should aim to satisfy the three MPTCP design goals, i.e., improve throughput, do no harm, and balance congestion. Thus, if an MPTCP congestion control algorithm does not satisfy one or more of these goals, the proposed attack may be less effective. First, we plan to evaluate the compliance of existing congestion control proposals with those design objectives, and study the impact of their deviation on the effectiveness

of the cross-path inference attack. Second, to detect and eliminate MPTCP connections whose congestion control algorithms may not satisfy the design objectives, we intend to investigate methods to fingerprint the congestion control algorithms employed by MPTCP connections. Towards this end, the active congestion control algorithm identification tool proposed by Yang et al. [15] can be extended or novel passive fingerprinting models can be developed for MPTCP.

Inferring other end-to-end metrics. As part of our ongoing and future work, we are investigating whether an attacker can infer additional metrics for other paths. Two key metrics associated with TCP connections are round trip time (RTT) and the sender's congestion window (cwnd), which can be measured passively [8]. Since each subflow in an MPTCP connection may have different values of RTT and cwnd, we plan to further exploit coupling among different MPTCP subflows to infer these metrics for single-path TCP flows on unobserved paths. It is noteworthy that the steady-state throughput is the ratio of equilibrium cwnd to RTT [14]. Consequently, given the successful cross-path throughput attack demonstrated in this paper, inferring either one of RTT or cwnd will automatically lead to the inference of the other.

Potential countermeasures. As mentioned earlier, the proposed cross-path throughput inference attack directly results from the design goals of MPTCP congestion control algorithms [6]. Therefore, congestion control algorithms that deviate from these design goals (e.g., cost- or energy-aware congestion controllers) may circumvent this attack. However, violations of these goals (e.g., do no harm) may degrade the performance of single-path TCP flows. Besides, encryption based solutions such as IPsec [7] or other end-to-end encryption schemes can be employed to evade this attack.

6. RELATED WORK

To the best of our knowledge, only RFC 6181 [1] and a follow-up Internet-Draft [2] in prior literature provide the discussion of MPTCP-specific flooding, hijacking, and DoS attacks. However, the flooding and hijacking attacks discussed in [1, 2] are either infeasible because their prerequisites are hard to satisfy or they have a fairly limited security impact.

In the flooding attack, an attacker first establishes an MPTCP connection to a server that can generate a significant amount of traffic and then falsely adds the victim's address as one of the available addresses to direct as much traffic as possible to the victim. However, the effectiveness of this attack is limited because the victim would issue RST packets addressed to the server immediately upon receiving the data and the server would terminate the subflow upon receiving the RST packets.

In the hijacking attack, an attacker adds her/his address to the MPTCP connection between two other peers and then participates by injecting data into or receiving data from the existing MPTCP connection. However, the feasibility of this attack is limited because the attacker would need to guess the 4-tuple (source IP, destination IP, source port, destination port), out of which ports are hard to know, and data/ACK sequence numbers.

In the DoS attack, an attacker sends a large number of `SYN+MP_JOIN` messages to a host with already established MPTCP connection, which triggers the creation of half-open connection states on the server that eventually overwhelm it. However, the feasibility of this attack is limited because the attacker would need to blindly guess the 32-bit MPTCP session identifier token.

7. CONCLUSIONS

In this paper, we present and demonstrate the feasibility of a new class of attacks on multipath transport protocols: an attacker can successfully infer properties of unmonitored paths within small measurement intervals. Specifically, an attacker can infer the throughput of unmonitored paths with up to 90% accuracy and within measurements interval of less than two minutes by exploiting coupling among MPTCP subflows. We emphasize that these attacks are not specific to a particular implementation, but directly result from the design goals of MPTCP. As a result, we argue that these attacks are fundamental to MPTCP and may constitute a barrier to its wide-scale adoption. We draw attention to this important security problem, and urge the research community to develop adequate countermeasures, so the benefits of MPTCP can ultimately be fully realized.

Acknowledgements

The authors would like to thank Don Towsley, Olivier Bonaventure, and the anonymous reviewers for their useful feedback on this paper.

This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

8. REFERENCES

[1] M. Bagnulo. Threat analysis for TCP extensions for multipath operation with multiple addresses. RFC 6181, Internet Engineering Task Force (IETF), March 2011.

[2] M. Bagnulo, C. Paasch, F. Gont, O. Bonaventure, and C. Raiciu. Analysis of MPTCP residual threats and possible fixes. draft-bagnulo-mptcp-attacks-00, Internet Engineering Task Force (IETF), Work in progress, July 2013.

[3] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated services. RFC 2475, Internet Engineering Task Force (IETF), December 1998.

[4] R. Castro, M. Coates, G. Liang, R. Nowak, and B. Yu. Network Tomography: Recent Developments. *Statistical Science*, 19(3):499–517, 2004.

[5] Cisco. *Implementing Quality of Service Policies with DSCP*, February 2008. Document ID: 10103.

[6] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure. TCP Extensions for Multipath Operation with Multiple Addresses. RFC 6824, Internet Engineering Task Force (IETF), January 2013.

[7] S. Frankel and S. Krishnan. IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap. RFC 6071, Internet Engineering Task Force (IETF), February 2011.

[8] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, and D. Towsley. Inferring TCP Connection Characteristics Through Passive Measurements. In *IEEE INFOCOM*, 2004.

[9] R. Khalili, N. Gast, M. Popovic, U. Upadhyay, and J.-Y. L. Boudec. MPTCP is not pareto-optimal: performance issues and a possible solution. In *ACM CoNEXT*, 2012.

[10] Q. Peng, A. Walid, and S. H. Low. Multipath tcp algorithms: theory and design. In *ACM SIGMETRICS*, pages 305–316, 2013.

[11] C. Raiciu, M. Handley, and D. Wischik. Coupled congestion control for multipath transport protocols. RFC 6356, Internet Engineering Task Force (IETF), October 2011.

[12] C. Raiciu, C. Paasch, S. Barre, A. Ford, M. Honda, F. Duchene, O. Bonaventure, and M. Handley. How hard can it be? Designing and implementing a deployable Multipath TCP. In *USENIX NSDI*, 2012.

[13] M. Scharf and A. Ford. Multipath TCP (MPTCP) Application Interface Considerations. RFC 6897, Internet Engineering Task Force (IETF), March 2013.

[14] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley. Design, implementation and evaluation of congestion control for multipath TCP. In *USENIX NSDI*, 2011.

[15] P. Yang, W. Luo, L. Xu, J. Deogun, and Y. Lu. TCP Congestion Avoidance Algorithm Identification. In *IEEE ICDCS*, 2011.