MULTIPATH TCP, PWNING TODAY'S NETWORKS WITH TOMORROW'S PROTOCOLS.

Catherine Pearce – Catherine.pearce@neohapsis.com, twitter: @secvalve

ABSTRACT:

MultiPath TCP (MPTCP) is an extension to TCP that enables sessions to use multiple network endpoints and multiple network paths at the same time, and to change addresses in the middle of a connection. MPTCP works transparently over most existing network infrastructure, yet very few security and network management tools can correctly interpret MPTCP streams. With MPTCP network security is changed: how do you secure traffic when you can't see it all and when the endpoint addresses change in the middle of a connection?

This paper briefly discusses how MPTCP breaks assumptions about how TCP works, and how this can be used to evade security controls. We will also discuss tools and strategies for understanding and mitigating the risk of MPTCP-capable devices on a network.

INTRODUCTION

Ubiquitous in modern networking, with much of the world's infrastructure depending upon it, but its single path nature is a significant shortcoming in many modern uses. Where there is a need to combine the bandwidth of multiple network connections, or to roam across different network locations while maintaining a connection, TCP falls short. This has motivated both research and proposed solutions. While many proposed solutions have been revolutionary (comprising complete replacements, notably Stream Control Transmission Control Protocol (SCTP)), recent proposals have been more evolutionary extensions of TCP itself. Multipath TCP (MPTCP) is one such expansion. It adds the ability to split TCP flows over multiple network endpoints, and to change network endpoints seamlessly without dropping the connection. It is well on its way to becoming a widely accepted standard and runs transparently on most existing infrastructure, but it fundamentally changes how TCP communication actually works. MPTCP appears the same as TCP on a network, and is backwards compatible, but differs from TCP in various aspects including traffic routing, network addressing, and connection direction.

Given the importance of TCP to networked devices, any changes or extensions will have far-reaching implications. The security of the protocol itself against attack is not sufficient, as no protocol is used in isolation. Security assessments must take the environment into account because a protocol may itself be secure from attack, but breaking assumptions made by network devices that have been valid until now brings significant implications -- both good and bad. With MPTCP there are two key sets of network security implications to consider: a) those during the transitional period when only some infrastructure is aware of MPTCP and b) those once MPTCP is fully deployed when all the infrastructure is aware of MPTCP. Both of these conditions bring their own challenges to network security and network security architectures.

BACKGROUND

TCP defines each connection as between two singular fixed endpoints (in a 1-1 cardinality), and defines these endpoints as a combination consisting of an IP address and a TCP "port number". This static 1-1 mapping is the key weakness of TCP which has motivated the development of MPTCP (Raiciu et al. 2012). The static 1-1 mapping of endpoints means that TCP cannot support increasingly more common scenarios where devices (mobile devices) need to shift their communication address or interface regularly, use multiple redundant links to their full capacity (data centers), or otherwise utilize multiple different network connections in the same TCP connection. Furthermore, mobile devices such as smartphones not only have diverse network connections with different transport characteristics, they also move around both physically (across space), and logically (across networks). Any single TCP connection cannot handle changing endpoint addresses in the middle of a communication and must break and re-establish a connection for each network address change.

MULTIPATH NETWORKING AND MULTIPATH TCP (MPTCP)

Multipath Networking

Multipath networking is not a new idea, and multipath communications are a core concept of IP's sending packets along the best route at any given time and place. The idea of Multipath TCP itself dates to at least 1995(Huitema 1995). The SCTP protocol was developed to achieve many of the same goals, but did not result in widespread adoption, primarily due to the requirement that software and hardware (where internal software cannot be changed) be rewritten to support it (Barré et al. 2011). Most research into the feasibility, design, standardization, and implementation of what has become MPTCP has occurred in the last decade.

Initially there were questions raised about the feasibility of multipath communications, particularly their stability and efficiency without complex management overhead. However, it was shown to be possible through investigations of the feasibility of stable and efficient multipath routing and congestion in pooled connections (Kelly & Voice 2005; Key et al. 2006). Later, multipath approaches were discussed as solutions to existing problems on the Internet (Han et al. 2006; Raiciu et al. 2009; Wischik et al. 2008). Regarding the actual mechanism for widely used multipath communications, the feasibility of extending TCP was thoroughly investigated (Honda et al. 2011; Langley 2008), which led to the design of MPTCP in its current state as a TCP extension.

Multipath TCP (MPTCP)

The IETF Multipath TCP working group was set up in 2010 and was responsible for the development of what has become the MPTCP standard. The first MPTCP draft of what would become the core MPTCP specification is RFC 6824(A. Ford 2013) which was introduced in mid-2010, and protocol design and implementation has progressed significantly since then. At the moment, MPTCP is an experimental status RFC. MPTCP was designed to run on top of today's Internet Protocol stack by utilizing a standards-compliant TCP with additional TCP options in the options field of the TCP header. Furthermore, it is designed with a worst-case fallback to standard TCP if errors occur that make continuing MPTCP communications impossible. Further details on the rationales behind MPTCP design are given in (Raiciu et al. 2012) and details on the real world feasibility of extensions using TCP options are presented in (Honda et al. 2011). As a result, MPTCP is architecturally a layer that sits above TCP in the TCP/IP stack.

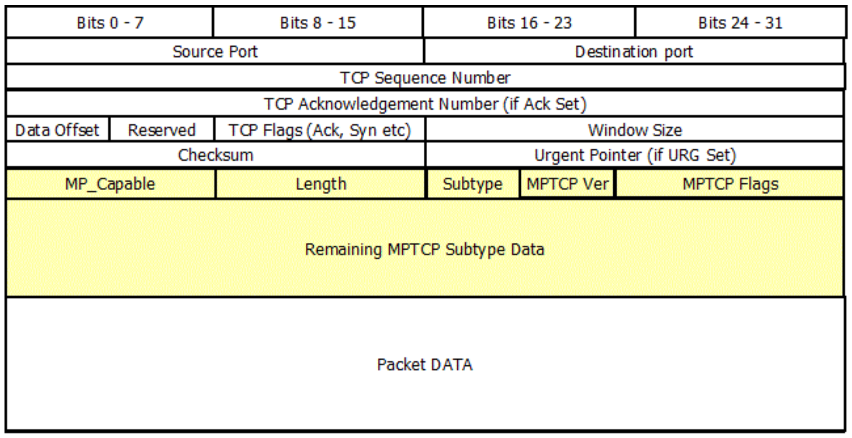

Figure 1: MPTCP Details added on to the end of a standard TCP packet

An MPTCP connection is made up of one or more subflows, with each subflow being a TCP flow with various TCP options set to communicate MPTCP information. As a result, there is no such thing as an MPTCP packet - MPTCP packets are simply standards compliant TCP packets with additional TCP options to handle MPTCP data. As a result, MPTCP is effectively a subtype of TCP traffic with additional capabilities needed for multipath communications.

MPTCP Functional Architecture

Although this work will not delve deeply into the MPTCP specification, some basic technical details should be understood. In particular, it is important to understand how MPTCP communicates flow control, how it identifies flows as related to MPTCP connections, and how it verifies integrity.

As MPTCP is implemented within TCP options, it adds extra control flow details as a TCP options field under the MP_CAPABLE option (0x30). In addition, there are several subtypes of MPTCP option defined by a 4-bit field, as well as some MPTCP handshake algorithms defined by 6 single bit flags. Current MPTCP Option Subtypes are briefly given in figure 2, while the only handshake algorithm presently defined is Hash Message Authentication Code (HMAC). Both subtypes and handshake algorithms are managed by IANA as a part of the TCP-Option Kind Registry(Anon 2014).

- **0x0 - MP_CAPABLE** - Multipath Capable [Declares MPTCP Support in connection setup]
- **0x1 - MP_JOIN -** Join Connection [Request to join the current MPTCP connection, may require a valid handshake entry to authenticate request]
- **0x2 - DSS -** Data Sequence Signal (Data ACK and data sequence Mapping) [Communicates MPTCP sequence mapping, Checksum, and ACKs receipt]
- **0x3 - ADD_ADDR -** Add Address [Tell a host that you have additional addresses available]
- **0x4 - REMOVE_ADDR -** Remove Address [Request the remote end remove a specific subflow]
- **0x5 - MP_PRIO -** Change Subflow Priority [Communicates streams priority preference]
- **0x6 - MP_FAIL -** Fallback [Fall back to regular TCP behavior]
- **0x7 - MP_FASTCLOSE** [Communicate that entire MPTCP subflow is closing, equivalent to RST in TCP but affects all subflows]
- 0x8-0xe - Currently Unassigned

Figure 2: MPTCP's Subtypes messages, from RFC 6824(A. Ford 2013)

We briefly discuss below the operation of MPTCP as it related to these subtypes.

MPTCP ADOPTION

Although MPTCP is not widely deployed, several open source implementations exist, and limited support has also been added into some major open source network tools. Operating systems with implementations available include (but are not limited to) Linux (Raiciu et al. 2012), BSD(Williams et al. 2013), and various Android ports. In addition, some MPTCP-aware commercial products exist, including vendors which have disclosed the existence of support (Jay Young & Wing 2013) and vendors who did not initially disclose MPTCP support but still offered it in their products. The most notable of the initially non-disclosing vendors is Apple which introduced MPTCP support into the voice command section of their iOS devices with iOS 7(Apple 2014). Currently, no support exists for Microsoft Windows platforms. Although the two stated use cases (i.e., datacenters and mobility) motivating MPTCP largely use platforms other than Windows, Microsoft's support is likely to be crucial for long-term adoption of MPTCP. This is because MPTCP largely operates at the socket layer, and therefore much software will not need to be rewritten to support it in some capacity. However, Software that performs raw reads or writes from the network interface will need to be modified to support MPTCP. In the long term, software which operates using the operating system's networking interface will need to have a more nuanced understanding of addressing, and to not to use network address alone to make decisions. In the interim, non-MPTCP aware software can be made to use MPTCP by allowing the software to listen on all interfaces it may need to use for communication, but this has security implications.

There are market barriers to MPTCP adoption, if not carefully deployed with aware customers and integrated business models (Kostopoulos et al. 2010), as it removes the ability for network providers to actively manage traffic content for reasons such as advertisement injection or malicious content removal. MPTCP may also result in confusion for metered customers of network providers as they may not understand why their data bill suddenly increased greatly while they were connected to their wireless network. Additionally, there is the potential for perverse incentives to arise leading to path arbitrage (Waldburger et al. 2011) where a network provider or network peering link somewhere along the path drops certain traffic (such as that from lower paying customers, or that travelling over more expensive paths), with the understanding that because they are using MPTCP they have other paths open and should not notice. Nevertheless, there is significant performance and usability to be gained if it becomes widely deployed.

Detal et al(Detal et al. 2013) developed techniques for determining whether MPTCP is actively used, otherwise supported, and they also did preliminary scans of key websites.

As a part of our research we built our own scanner using these techniques to scan systems for MPTCP support. We describe this later.

BASIC CONCEPTS AND OPERATION OF MULTIPATH TCP (MPTCP)

In this work we will endeavor to use definitions aligning with those given in RFC 6824, listed in Table 2.

Term	**Description**
Path	A sequence of links between a sender and a receiver, defined by a 4-tuple of source and destination address/port pairs.
Subflow	A single TCP flow operating over an individual path, which forms part of a larger MPTCP connection.
MPTCP Connection	A set of one or more subflows, over which an application can communicate between two hosts. There is a one-to-one mapping between a connection and an application socket.
Data-level	The payload data is nominally transferred over a connection, which in turn is transported over subflows.
Token	A locally unique identifier given to a multipath connection by a host. May also be referred to as a “Connection ID”.
Host	An end host operating an MPTCP implementation, and either initiating or accepting an MPTCP connection.

Table 2: MPTCP definitions in brief, derived from RFC 6824(A. Ford 2013)

Unlike TCP, which identifies connections by a tuple of port and address combinations, MPTCP identifies connections through the use of an address agnostic connection ID. The keys used to generate this connection ID are set up and communicated (in the clear) in the initial handshake, sent in a protected form thereafter, and only sent over the wire in the clear again if the connection is *fast closed*.

Overall MPTCP Initial Connection Handshake

MPTCP performs connection establishment over the standard TCP 3-way handshake using the TCP options. When a client (A) initiates a connection to a server (B), the MPTCP handshake proceeds as follows:

Step 1. (A->B): The initial handshake SYN packet indicates MPTCP support, sends the sender’s message signing key, and indicates if the MPTCP checksum is required.

Step 2. (A<-B): If MPTCP is supported, B replies with the MPTCP Options set, and includes its own signing key. (If B replies back without indicating MPTCP support, the connection proceeds as a standard TCP connection.)

Step 3. (A->B) The client responds back, including both its own key and the server's key (to allow B to remain stateless until a connection is established – similar to TCP SYN cookies in effect)

The exchanged keys are used later to validate connection JOIN (MP_JOIN) requests.

Overall MPTCP Connection Closing

The entire MPTCP connection closes either once all subflows are closed, or upon receipt of a MP_FASTCLOSE (which is equivalent to a TCP reset, but affects the entire MPTCP connection). The MP_FASTCLOSE is accompanied by *full* 64-bit connection identifiers, rather than the 32-bit version normally used. As a result, it may be possible that any ability to intercept an MP_FASTCLOSE request and prevent it from reaching the destination may enable a connection to be hijacked without the knowledge of either party.

MPTCP Subflow changes

Subflow changes can occur automatically on failure or at the request of either party. These add subflows via address/port tuples to the connection which is identified by the Token (Connection ID).

- MP_ADD_ADDR, MP_REMOVE_ADDR: these requests add or remove an address that occurs within an already established connection. Given the widespread usage of Network Address Translation (NAT) devices it is likely that many hosts will request to add a non-routable, private or non-public address (such as 192.168.1.2). In this case the subflow should be added by the MP_JOIN method.
- MP_JOIN: this is an MPTCP message subtype on inbound traffic indicating that it should be added to an existing MPTCP connection. The stream is identified by a connection ID and the join request is mutually authenticated. The authentication occurs by calculating the Hash Message Authentication Code (HMAC) values of random single-use nonces. Thus, to succeed in adding a connection the party sending the MP_JOIN request must have both keys, nonces that have not been previously used– making it both authenticated and resilient to replay attacks.

MPTCP Data Sequence Flow and Acknowledgement

MPTCP runs above standard TCP flows, and adds additional sequence mapping for the MPTCP connection. Therefore, MPTCP has separate transmission windows for each subflow (component TCP connection), as well as an overall mapping for the entire connection. The way data is routed among subflows depends on the configuration and implementation specifics and is beyond the scope of this work. The DSS subtype maintains the sequence mapping of data to the MPTCP connection, and also manages an additional level of MPTCP checksums.

Handling of Error Conditions

MPTCP checksums are maintained to detect if packet content is altered by a middlebox (such as a Network Address Translation (NAT) device or firewall) which is not MPTCP aware. If the checksums are consistently altered on a path, MPTCP will drop that subflow *as long as* other MPTCP subflows are available. If there is no available alternate path, MPTCP falls back to the traditional TCP protocol. This ability to detect and avoid paths with non-MPTCP-aware middleboxes has the security side effect that man-in-the-middle attacks which alter packet content will not function against MPTCP *unless* the tool is MPTCP aware or only one subflow is available. If the tool is *not* MPTCP aware and alters packet content, the MPTCP checksum will fail and other paths will be preferred if available. This is a new form of intrusion prevention evasion; we elaborate more on this attack later.

In the event that a valid MPTCP connection fails in a way that cannot be routed around, and TCP communications proceed normally, the connection reverts back to standard TCP via the MP_FAIL option. A simple example of this would be where all paths available return checksum errors (indicating content modification). If a single MPTCP path remains, MPTCP will drop to that path but remain MPTCP capable. However, once the connection has fallen back to TCP, the connection cannot change back to MPTCP.

A more in-depth description of MPTCP is beyond the scope of this paper but can be found in numerous tutorials and introductions to MPTCP have been published recently (Bonaventure 2012; Tcp 2013).

SCANNING FOR MPTCP SUPPORT

General Idea

Detal et al(Detal et al. 2013) developed techniques for determining whether MPTCP is actively used, otherwise supported, and they also did preliminary scans of key websites. Their scans indicated very low uptake in the HTTP space, and our own investigations of public-facing service to agree with this. Their scan method was designed to test whether paths would permit MPTCP traffic across them, and in general it appears successful for this in our tests.

However, one problem mentioned in their paper, the problem of false positives due to TCP option ECHO, was causing problems for us so we developed a new technique to scan for MPTCP support. After examining the MPTCP protocol specification, and looking for ways to distinguish a true MPTCP host from a pretender we settled on the technique of MP_JOIN spoofing. We detail this below

MPTCP Support Checks via MP_JOIN Spoofing

Our Technique for distinguishing hosts that support MPTCP from those that merely reply with MP_capable without establishing and sending data over an MPTCP connection relies upon the way MPTCP handles invalid join requests. If a join requests is received for a stream which does not exist then the host must reply with a TCP reset. Conversely, a host which does not understand MPTCP will treat the packet as a standard TCP SYN that happens to have some unknown option on it. Therefore, if a TCP connection establishes with MP_CAPABLE but does not succeed with randomized MP_JOIN then the host does support MPTCP. We illustrate this in figure 2.

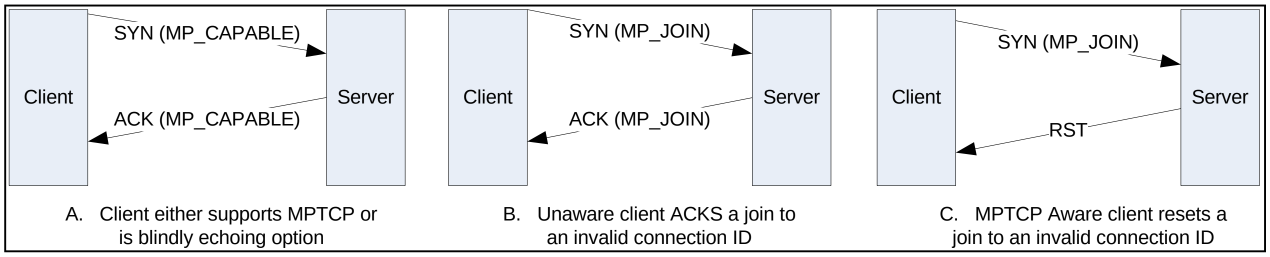
Client
SYN (MP_CAPABLE)
ACK (MP_CAPABLE)
Server
A. Client either supports MPTCP or is blindly echoing option
Client
SYN (MP_JOIN)
ACK (MP_JOIN)
Server
B. Unaware client ACKS a join to an invalid connection ID
Client
SYN (MP_JOIN)
RST
Server
C. MPTCP Aware client resets a join to an invalid connection ID

Fig 2 - MPTCP support cannot be verified by looking for an MP_CAPABLE reply alone.
It can be verified in a two step process by next sending a request to join an invalid MPTCP connection.

While we have found this to be a very effective test, and have implemented it as a second level of verification in our network scanner, it is possible that in future MPTCP-aware network devices which block MP_JOIN subtypes will lead to false positives.

ASSUMPTIONS ABOUT TCP BROKEN BY MPTCP

MPTCP breaks some key assumptions about how connection-oriented TCP works. These assumptions have usually been reliable when analyzing network traffic, but MPTCP breaks each of these in some situations. The key ones are below, and we discuss some threats arising from these in the next section.

1. TCP connections connect two, and only two IP addresses and require the addressees to remain constant for the life of the connection.
2. Related to 1 above, TCP connections have a single TCP port they use at each endpoint and it must stay constant throughout the life of the connection.
3. TCP Connections can't split traffic over multiple links using layer 3 alone
4. TCP Connections aggregate traffic from multiple links using layer 3 alone

And some security angles on the above:

5. Correlating flows from different network egress points doesn't require correlating exact packet content to make useful decisions.
6. Traffic fragmentation only happens along one path (and can be defeated by implementing reassembly logic).
7. A packet with TCP SYN set and ACK not set indicates the direction of connection establishment.
8. To reset a TCP connection and associated data flow you only need the correct TCP endpoints and sequence number.
9. To alter the content in a TCP connection you only need to manage the TCP connection state and checksum
10. Connections will only use one of IPv4 or IPv6.

EXAMPLES OF SPECIFIC THREATS FROM MPTCP TRAFFIC

Fragmentation

MPTCP relies upon the sending party to provide a map of the overall data sequence to the specific subflow. The sending party has complete control over which paths the data is sent over. Although the other party may state a preference which is not binding, as there is not currently any implementation that actually enforces a preference.[1]

However, because the party can choose how to divide up data it can do fragmentation attacks very easily, for instance by sending every alternate byte down alternate paths vi different interfaces. Because of technical details of how the mapping occurs, this wouldn't be very efficient, however if both ends are doing this then there may be no single point on the network which can observe the traffic from both flows. This is the unique danger of a multipath fragmentation attack – reassembly capability alone is not sufficient to observe all traffic.

Aggregation, Persistence & Resilience

Additionally, MPTCP adds an additional degree of resilience and redundancy not previously seen in TCP connections. Where TCP handles reliability across a single path, MPTCP handles it across many paths. Where previously a single compromised path could be used to manipulate traffic now the other party must be able to manipulate all paths in use.

This has interesting takeaways for things like botnet Command and control. We suspect that if this technology becomes mainstream we will see multiple path fast flux routing networks, where traffic is sent via an ever changing set of intermediate hosts, fragmented across them in a redundant fashion.

In the immediate term however, MPTCP is designed to provide reliable delivery over multiple paths and to detect where interference happens and avoid that path. This may cause issues for organizations who use active or transparent security measures. If a client can find a path that supports MPTCP it will prefer it over the recommended path, leading to both avoidance of security measures and the potential for new forms of phishing attacks.

Address Hopping

MPTCP provides a degree address mobility in-connection which hasn't been seen in TCP before. An endpoint can add and remove addresses at will throughout the connection, with no requirement that any connection remain available, just that the chain of connections remains so. This adds additional complexity for monitoring and state handling, for now, we need to watch every connection, every possible connection, and track every advertised address.

[1] Requiring the other party to send you data via a specific interface is currently contrary to the philosophy of packet switching, where a packet is SENT down the preferred path – not received down it! Although it is feasible for an implementation to refuse to acknowledge data on the wrong interface, this is both against the spec nd counterintuitive (why refuse data you received?).

Combining the multiple link nature with the address hopping means we can immediately ad complexities to flow monitoring systems. For example in figure 3 below, this is one host using several addresses talking to a host with several addresses. If we do not know both that this is the case, and which address is which host then the web is impossible to make sense of. In figure 4, where we incorporate these details we can see that it is simply a host spreading traffic over many links to another host with many links.

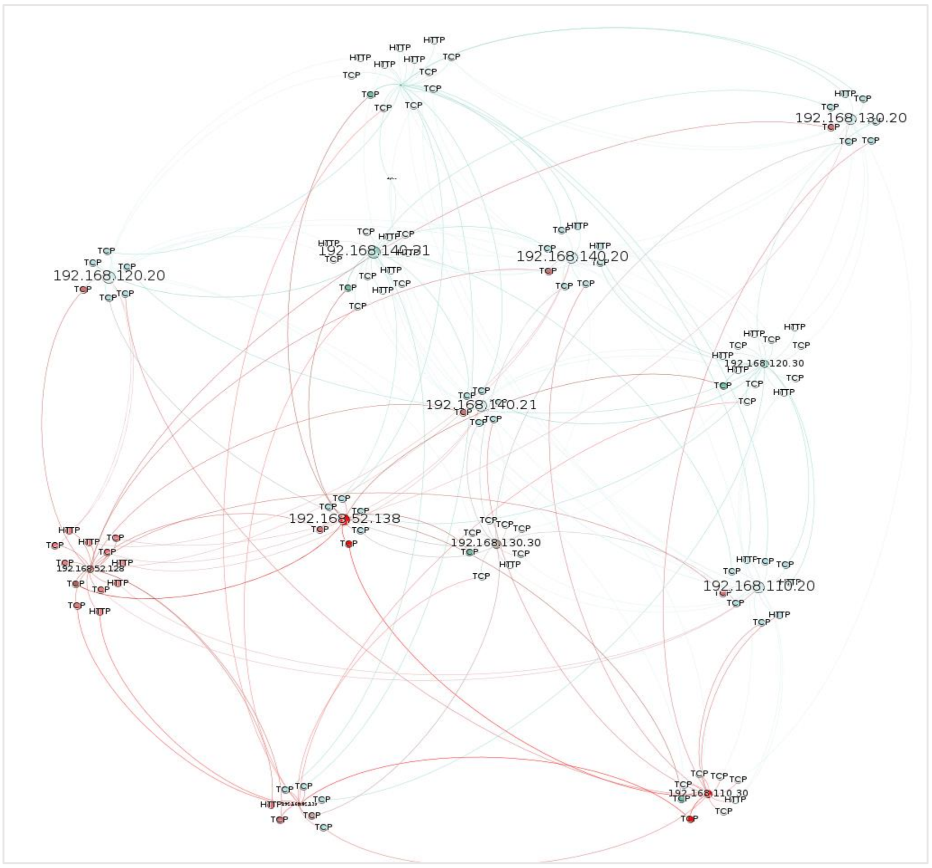

Figure 3 - MPTCP Flows can be impossible to make sense of if you cannot correlate addresses to hosts, as this picture indicates

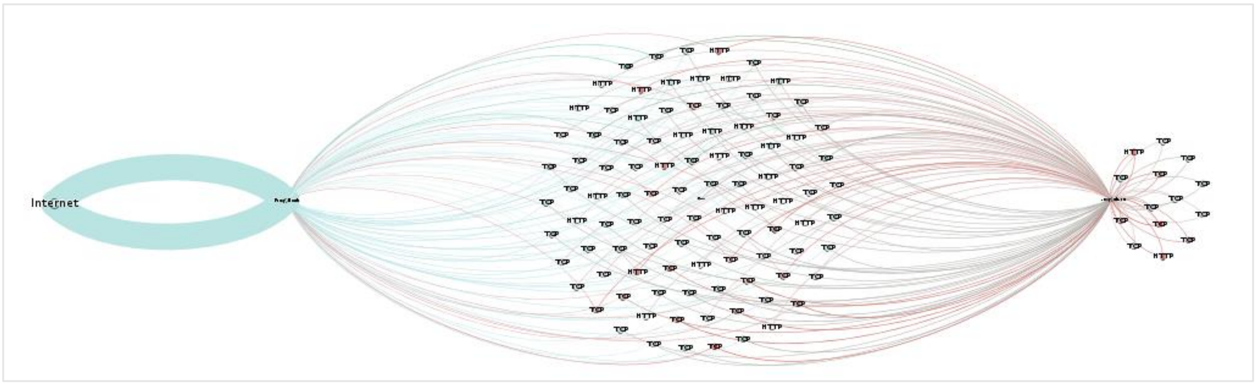

Figure 4 - When you are able to correlate flows and addresses to hosts a clear picture emerges, which was not there before.

MPTCP DEFENSES

MPTCP is a really promising technology, however many organizations use network security which is highly dependent on tight perimeters, flow monitoring and data inspection. Where network security technology is not ready, it may be significantly less effective against MPTCP traffic. While we recommend moving towards multipath capable technologies, this will be quite some time, but in the interim security remains very a real concern!

Awareness

First of all, since there is already technology out there that supports MPTCP it is something that should be known to exist. There are two key aspects of this, people and technology. People should be educated what it is if they see it, although actual procedures may not be necessary for some years yet. MPTCP itself adds a lot of complexity to TCP, and the cheatsheet we have released with this work should aid a bit with this education, simply by providing a reference guide for some of the intricacies which doesn't require reading an RFC.

Technology is a more difficult problem, as the network security technologies for a multipath world simply don't exist at present. When it does become available, core items such as firewalls and intrusion detection systems should be upgraded where possible. Where they cannot be upgraded to full support there may be ways to ad partial support via existing rules for inspection or filtering based on TCP options.

Observation

Determining the actual potential threat of MPTCP requires checking, or watching, for three specific things:

1. Hosts that support MPTCP
2. Network paths that do/do not support MPTCP
3. MPTCP traffic on the network

Host support can be verified through the use of the scanner tool released with this research. This tool checks not only for professed support but for actual protocol understanding.

Network support can be tested by this same system, however it cannot currently tell the difference between a host which does not support MPTCP and a path which strips out the MPTCP option. The Tracebox tool (Detal et al) provides a way to check a path for support, however it probably isn't an effective solution for testing many thousands of paths in a scalable way. We plan to release a better solution in future, but at present tracebox is the recommended solution for checking paths.

We have provided two additional tools which may assist in testing network equipment's response to MPTCP: a Linux route randomizer, and a simple IDS tester which sends a fragmented payload over a MPTCP connection.

The route randomizer is modified version of MultiNetworkManager (Evansen, 2013) that, when run on a Linux system using the Linux-mptcp reference implementation (Raichu et al 2012) randomizes route metrics many thousands of times a second and as a result ensures packets are routed through different interfaces between MPTCP hosts. For some types of testing it may be desirable to set the TCP Maximum Sequence Size small, but this is beyond the scope of this work to detail.

Response

Response to undesired MPTCP on a network can occur in two main forms, which we have labeled active and passive. Active, where the MPTCP protocol is actively used, and passive, where MPTCP is dealt with but not explicitly understood by the technology doing the manipulation.

Passive response

Passive response and MPTCP prevention has the goal of preventing MPTCP traffic from flowing past a point on a network. The conceptually simplest method is to strip out the MPTCP TCP option (30) at firewalls, rewriting traffic without it. The alternative is transparent proxies which force connection termination.

Be aware that while this can work, it relies upon the assumption that no MPTCP-supported paths are available to the endpoints, for if these exist they will be preferred. This will achieve the security goal of preventing MPTCP traffic flowing past you, but it won’t necessarily mean the client doesn’t communicate around your measures!

Active Response

Active response is more likely to succeed, however it requires technology the is both more powerful (to track MPTCP states) and more mature. At present we do not know of ANY network equipment which is capable of actively intervening in MPTCP traffic. Nevertheless we discuss the approaches we have identified which will be useful in some cases during the transitional period.

The options are: strip only initial MPTCP packets (to prevent it ever being established as MPTCP), to force an existing session to downgrade to MPTCP, or sending MPTCP fastclose requests to all MPTCP streams identified. These last two are more complex, as they require tracking the full connection id or initial negotiated encryption keys.

THE MULTIPATH FUTURE

Multipath communications can allow connection-oriented networking to move closer to the ideal nature of a packet switching network, where data is sent along the best path at any point, and not simply along the path it began the journey on. It provides an improved level of degree of mobility, and by enabling link aggregation the user experience will improve while datacenter networking will probably become slightly less proprietary and complex.

This same shift of power from the network to the host however is what causes the network security challenges to organizations. The amount of computing power required to track connections is significantly higher, with significantly more intra-node communication needed for successful traffic reassembly, and some existing architectures do not have enough reserve capacity to manage this. Firewalls and Intrusion detection systems also need to evolve to make better decisions on partial data content.

Additional opportunities arise for those who wish to protect connection privacy too, as split-path cryptography may significantly increase the resources needed for traffic interception and manipulation. We believe that this area has been barely explored, but that technologies will arise to integrate MPTCP into TOR and related technologies.

FUTURE WORK

We are also looking into the following areas: MPTCP attack proxies, MPTCP reverse connections and NAT traversal, Side channel cryptography approaches, redundant stream hopping.

CONCLUSION

While this work may have come across as antagonistic to multipath-tcp this is not the intention. MPTCP is a technology that has the potential to improve many things about connection-oriented networking, and the internet at large, for the better.

In the end, our goal was proof of concept and awareness, and we hope others will take the things we have mentioned and use them to make the impending multipath rollout both successful and secure. Some of the things we have investigated, and techniques mentioned in this work, will likely be better incorporated into existing tools. For instance, organization would likely be better served by the MPTCP support scanning technique in existing tools than using out custom tool. Similarly, it is likely that as MPTCP development proceeds and API development matures we will end up with fine-grained control of MPTCP connections in implementations which does not presently exist.

TOOLS RELEASES

We released the following tools or aids with this talk. Updated links will be provided during the presentation, and can be found at the Neohapsis website: http://www.neohapsis.com

1. MPTC Scanner
2. MPTCP Cheat sheet
3. MPTCP Route Randomizer (modified version of MNM)
4. MPTCP IDS tester

REFERENCES & BIBIIOGRAPHY

Anon, 2014. *Transmission Control Protocol (TCP) Parameters*, IANA. Available at: https://www.iana.org/assignments/tcp-parameters/tcp-parameters.xhtml.

Apple, I., 2014. *iOS: Multipath TCP Support in iOS 7*, Apple, Inc. Available at: http://support.apple.com/kb/HT5977.

Bagnulo, M., 2011. Threat Analysis for TCP Extensions for Multipath Operation with Multiple Addresses. *Internet Eng. Task Force*, RFC 6181, pp.1–17. Available at: http://tools.ietf.org/html/rfc6181.

Barré, S., Paasch, C. & Bonaventure, O., 2011. Multipath TCP: from theory to practice. *Netw. 2011*, pp.1–42. Available at: http://link.springer.com/chapter/10.1007/978-3-642-20757-0_35 http://datatracker.ietf.org/doc/draft-barre-mptcp-impl/.

Bonaventure, O., 2012. An overview of Multipath TCP. *; login Mag. ...*, pp.17–23. Available at: http://dial.academielouvain.be/handle/boreal:114081.

Detal, G. et al., 2013. Revealing middlebox interference with tracebox. *Proc. 2013 Conf. Internet Meas. Conf. - IMC '13*, pp.1–8. Available at: http://dl.acm.org/citation.cfm?doid=2504730.2504757.

Evensen, K.R. MULTI Network Manager (MNM), 2013. <http://github.com/kristrev/multi>.

A. Ford, O.B. C. Raiciu M. Handley, 2013. TCP Extensions for Multipath Operation with Multiple Addresses. RFC 6824,. *Internet Engineering Task Force (IETF) Experimental RFC, January 2013.* Available at: http://medcontent.metapress.com/index/A65RM03P4874243N.pdf http://www.hjp.at/doc/rfc/rfc6824.html.

Han, H. et al., 2006. Multi-Path TCP: A Joint Congestion Control and Routing Scheme to Exploit Path Diversity in the Internet. *IEEE/ACM Trans. Netw.*, 14(6), pp.1260–1271. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4032726.

Honda, M., Nishida, Y. & Raiciu, C., 2011. Is it still possible to extend TCP? *Proc. ...*, p.181. Available at: http://dl.acm.org/citation.cfm?doid=2068816.2068834 http://dl.acm.org/citation.cfm?id=2068834.

Huitema, C., 1995. *Multi-homed TCP*, http://tools.ietf.org/html/draft-huitema-multi-homed-01: IETF.

Jay Young & Wing, D., 2013. *MPTCP and Product Support Overview*, Cisco. Available at: http://www.cisco.com/c/en/us/support/docs/ip/transmission-control-protocol-tcp/116519-technote-mptcp-00.pdf.

Kelly, F. & Voice, T., 2005. Stability of end-to-end algorithms for joint routing and rate control. *ACM SIGCOMM Computer Communication Review*, 35(2), pp.5–12.

Key, P., Massoulié, L. & Towsley, D., 2006. Combining multipath routing and congestion control for robustness. In *Information Sciences and Systems, 2006 40th Annual Conference on*. IEEE, pp. 345–350.

Kostopoulos, A. et al., 2010. Towards Multipath TCP Adoption: Challenges and opportunities. *Next Gener. Internet (NGI), 2010 6th EURO-NF Conf.*, pp.1–8. Available at: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5534465.

Langley, A., 2008. Probing the viability of TCP extensions. *URL http//www. Imp. org/binary/ecntest. pdf*, pp.1–3. Available at: http://www.imperialviolet.org/binary/ecntest.pdf.

Raiciu, C. et al., 2012. How hard can it be? designing and implementing a deployable multipath TCP. *NSDI*, (1). Available at: https://www.usenix.org/system/files/conference/nsdi12/nsdi12-final125.pdf.

Raiciu, C., Wischik, D. & Handley, M., 2009. Practical congestion control for multipath transport protocols. *Univ. Coll. London …*. Available at: http://nrg.cs.ucl.ac.uk/mptcp/mptcp-techreport.pdf.

Tcp, M., 2013. Decoupling TCP from IP with Multipath TCP. , (March).

Waldburger, M. et al., 2011. Socio-Economic Services for European Research Projects (SESERV) First Report on Economic Future Internet Coordination Activities Public. , (258138).

Williams, N., Stewart, L. & Armitage, G., 2013. Design Overview of Multipath TCP version 0.3 for FreeBSD-10. *CAIA Technical Report*, 130424A. Available at: http://caia.swin.edu.au/reports/130424A/CAIA-TR-130424A.pdf, http://caia.swin.edu.au/newtcp/mptcp/.

Wischik, D., Handley, M. & Braun, M.B., 2008. The resource pooling principle. *ACM SIGCOMM Comput. Commun. Rev.*, 38(5), p.47. Available at: http://portal.acm.org/citation.cfm?doid=1452335.1452342.