PREMIUM: Private Reactive Multipath Communication Middleware

Isabel Cristina Monteiro da Costa
Instituto Superior Técnico, Universidade de Lisboa
Advisors: Professor Miguel Filipe Leitão Pardal and
Professor Miguel Nuno Dias Alves Pupo Correia

***Abstract*—Any communication over the Internet is a target for eavesdropping attacks from unauthorized parties, sometimes involving hijacking attacks due to Border Gateway Protocol (BGP) vulnerabilities. We want to protect confidentiality of communication between two entities that communicate over the Internet and cloud networks, i.e., data transfer between a client and a server. In order to provide secure communication over the Internet, against the most resourceful, powerful and motivated adversaries, we propose PREMIUM, a private reactive multipath communication middleware. It provides a mechanism to split network traffic among multiple paths, and is be able to react in near real-time to a hijacking attack, by shutting down the whole connection. The solution uses two services: MACHETE and Darshana. The first is a multipath communication mechanism that splits data, with Multipath TCP (MPTCP), among multiple physical paths on top of an overlay network, using when possible multiple Internet Service Providers (ISPs) through multihoming. The second is a route hijacking monitor, that uses a combination of detection mechanisms to alert the user that its data traffic is likely being intercepted. The end client uses this reactive middleware so that hijack alerts can trigger path changes, to protect the communication.**

Keywords: Cloud Security, Secure Channel, Communication Confidentiality, Multipath Routing, Route Monitoring

I. Introduction

We aim to protect communication confidentiality from eavesdropping attacks from the most resourceful, powerful and motivated adversaries. We want to protect this property even if the attacker has capabilities to break cryptographic protocols. To do so, this work will present a *PRivate rEactive MultI-path commUnication Middleware* (PREMIUM) library which provides an interface similar to secure sockets (SSL/TLS) and uses a combination of two existing services that perform multipath communication and route monitoring.

The Transmission Control Protocol (TCP) over the Internet Protocol (IP) provides only single-path communication, however multipath communication has become a topic of interest when studying new ways to communicate between two distant nodes within a network, both for performance [1] and security [2]. If an attacker gains control over a node in the path, then it has full access to the path where all the data stream circulates from a source to destination, all data can be compromised. Multipath communication is the concept of splitting traffic, from a source, across different routes over a network, and then sinking the data on the destination. This may be used to protect availability and add redundancy to communications. In this project it is used for security.

MACHETE [2] is an application-layer multipath communication mechanism that provides additional confidentiality by splitting data streams of an already secure channel in different physical paths. This mechanism makes use of a multiple techniques to enforce multipath communication, such as multihoming, multipath TCP and overlay routing.

The underlying routing protocol in the Internet is BGP, version 4, which is known to have vulnerabilities that can be exploited by attackers. The traffic between two nodes can be diverted and intercepted without the source or destination's knowledge, therefore suffering an hijack attack. Because these types of attacks represent a threat to the security of communications and the routing system over the Internet, it is important to identify and avoid them. These attacks can be detected by monitoring network routes. One recent hijacking attack, that occurred on April 24th of 2018, which exploited its vulnerabilities, was done to the Amazon's Domain Name System (DNS) traffic [1].

DARSHANA [3] is a monitoring solution that detects route hijacking based solely on data plane information, and has enough redundancy to prevent attacker countermeasures such as dropping of traceroute probes. By using active probing techniques it is able to detect attacks in near real time.

PREMIUM is our final solution which consists of a private reactive multipath communication middleware. This is able to react in near real-time to a possible hijacking attack. It uses the base of MACHETE, to split data across multiple paths, and DARSHANA, to monitor routes used for multipath communication. Upon detecting an attack it reacts by shutting down all communication.

This project was developed in the context of SafeCloud [4], an ongoing European project that aims at providing a complete solution to address privacy in Cloud Computing. This work is mainly focused on secure communications despite the existence of powerful, motivated and well-funded adversaries.

A. Structure of the document

The rest of the document is organized as follows: Section II presents relevant work in the context of this project. Section III presents our solution PREMIUM, and and overview of its

[1]For more information about this attack the reader can visit https://doublepulsar.com/hijack-of-amazons-internet-domain-service-used-to-reroute-web-traffic-for-two-hours-unnoticed-3a6f0dda6a6f (as consulted on May 10th, 2018)

components. Section V presents the qualitative and experimental evaluation of our private reactive multipath communication middleware. Finally Section VI presents the preliminary conclusions of this research.

II. Related Work

This section addresses Multipath Communication for security in section II-A and Route Monitoring to detect hijack attacks in sectionII-B.

A. Multipath Communication for Path Diversity

Most of multipath communication applications address problems as availability, reliability and resilience of the network. This technique is used to transmit data redundantly across multiple paths, and avoid packet losses in case routes are compromised.

Some systems, such as H-SPREAD [5] and INSENS [6], use multipath routing for security purposes. Both of these are concerned with Wireless Sensor Networks (WSN).

MACHETE [2] provides an extra layer of security, for communications that already use a secure channel mechanism, with multipath routing. To accomplish this, it makes use of a combination of techniques: multihoming, MPTCP and overlay routing, to split streams of data among disjoint physical paths, achieving path diversity, in order to provide confidentiality for communications over the Internet. The idea behind using multipath communication in MACHETE is making the attacker's task harder, since it has to spy all the multiple paths to have access to all the data, which requires much effort. The combined techniques used by MACHETE will be explored in the following sections.

1) Multi-homing: One of the techniques that can be used to achieve path diversity is multihoming. It refers to a single customer being connected to multiple Internet Service Providers (ISPs), instead of just one. Usually it is used to optimize performance, availability among other metrics [2], [7], [8].

MACHETE uses multihoming, when available, so that the first autonomous systems along the path are different, which would not happen using single-homing. The mechanism uses this technique to minimize the single points of interception, where an attacker can eavesdrop information sent through several channels.

2) Overlay Routing: Overlay routing is the underlying operation in overlay networks, that allows nodes to communicate with each other to route packets between a sender and a receiver. Overlay networks represent a virtual network on top of a physical one with its IP routing infrastructure and without its modification [2], [7].

As mentioned before, MACHETE uses overlay networks combined with multihoming, to guarantee that the paths stay diverse, achieving path diversity. This mechanism does not choose the overlay nodes randomly, it uses a topology-aware decision algorithm to choose them according to their location. This type of algorithm allows the mechanism to make wiser decisions about the routing paths. The overlay network is single-hoped, where there is only one overlay node between the source and destination nodes, and there is no advantages in contrast with multi-hoped [9].

These virtual networks are usually used with the intent to take advantage of Internet's redundancy to improve reliability and performance. Whenever a path is unavailable or another one is preferred the traffic can be routed through a specific overlay node [7].

3) Multipath TCP: TCP communication is restricted to a single path per connection. To leverage multipath communication, the Internet Engineering Task Force (IETF) proposed Multipath TCP (MPTCP). MPTCP extends TCP protocol by enabling hosts to send data from a single TCP connection over multiple paths, also referred as subflows. A single stream of data from the source is split into one or more subflows, and then it is reassembled and delivered, in order and reliably to the destination [1].

MACHETE uses MPTCP to split data and transfer it across multiple paths, over an overlay network. In case multihoming is available, and the multipath device has some physical interfaces connected to different providers (each one with an IP address), MPTCP uses them to spread the split data. This mechanism makes use of the fullmesh feature to create a number subflows in order to match the number of overlay nodes to be used [2].

4) Path diversity: Path diversity represents the existence of multiple paths to reach a destination. These paths can be either physical or virtual. This concept is important regarding systems that use multipath communication for security, like MACHETE. Although, this is only relevant for security if more than one physical path is used. The less these paths overlap, the more diverse they are. If the attacker has access to one of the multiple routes, and these are completely diverse, then this attacker will not have access to all data streams.

When evaluating MACHETE, multihoming proved to be a key component to achieve path diversity [2]. The effectiveness of this approach depends on the path diversity between two endpoints. Using multihoming will not have many advantage if the paths provided by different ISPs overlap too much [7].

MPTCP does not ensure the use of multiple paths nor their diversity, however, it provides the necessary means to do so. This can also be assured if combined with other techniques such as multihoming [1].

Ultimately, path diversity depends on diversity of physical links, routing infrastructure, administrative control and geographical distribution [7].

B. Route monitoring

For the main objective of protecting communication confidentiality, it would be interesting to be able to monitor the routing paths established for communications over the Internet. By monitoring the network, we can understand if the chosen paths are not being eavesdropped by an attacker and if the data is not being compromised.

The Internet is composed of independent Autonomous Systems (AS), which have border routers connected through BGP, which is the standard inter-domain routing protocol. BGP was

designed without taking security into consideration, which causes the Internet to be vulnerable to hijacking attacks [10].

In this project we are most concerned with interception attacks, where the main goal of an attacker is to compromise data, that we want to protect. We use DARSHANA [3] that focus on detecting these types of attacks against an Internet user.

1) Route hijacking: Attackers can exploit BGP security weaknesses and perform attacks, such as route hijacking or IP prefix hijacking. These attacks occur when an attacker takes control of a block of IP addresses without the legitimate owners consent [11], *i.e.*, an AS advertises an unauthorized prefix from an address space that is unassigned or belongs to another AS. Other neighbor ASes that receive this announcement can select this route and forward the traffic to the compromised AS [12].

Prefix hijacking can be used to perform *interception* attacks in which an AS analyzes or even changes the packets before forwarding them to the legitimate destination. In this case both source and destination host might not notice that its traffic was intercepted, and data was compromised [12]. These attacks, that allow an attacker to spy on the communication, is our main concern in this project.

2) Detection mechanisms: Detecting route hijacking can be done by measuring latency and tracking routes that packets take over the Internet. The following paragraphs show different metrics and techniques used by DARSHANA to detect hijack attacks.

Monitoring network latency (Lat): A common practice of the majority of tools that measure latency is to calculate the round trip time (RTT). RTT is the sum of forward delay from source to destination and the reverse delay from destination to the source.

DARSHANA uses round trip time (RTT) measured by *cryptographic ping*, to analyze latency. This is a new version of ping designed to avoid an attacker from answering to ping probes earlier and fool the system. It uses public-key cryptography, where both endpoints of communication share their public keys. The source sends a nonce to the destination. This nonce is sent back ciphered with the destination's private key. In this way, authenticity of the destination is guaranteed.

RTT is important because in the case of a hijacking event, this metric's value tends to change significantly, and has the advantage of having low overhead.

Estimating hop count (Hop): DARSHANA estimates the number of intermediates devices from a source to a destination, hop count, since this metric can also change drastically if an interception attack eventually occurs. This variation is due to the natural deviation from the source's traffic to the hijacker's AS. In this situation the hop count increases significantly if the attacker is far from the source. Unlike RTT, the hop count is not affected by congestion.

Calculating path similarity (Path): Traceroute is a widely used tool to trace the path that packets take from a source host to a destination. It does not need to control the destination. This tool is used by network operators to identify routing failures and performance problems. It is also used by researchers to study the Internet and detect route hijacking attacks.

In [13], Augustin et al. propose a new version of traceroute, *Paris Traceroute*, that helps mitigating some anomalies related with the standard traceroute, mostly deficiencies related to per-flow load balancing. By using this traceroute, DARSHANA tracks periodically the path that packets are taking. It compares a new path measurement from the previous one. If these paths present significant differences it can indicate the occurrence of an hijacking event. This is because the deviation from the source's traffic to the attacker's AS, before being forwarded to the legitimate destination, causes the traffic's path to be different if an attack does not occur.

Monitoring propagation delay (Prop): In this mechanism DARSHANA calculates the propagation delay from the RTT's value. This is composed of two phases. In a first phase it calculates the propagation delay as a ratio between the link length among two nodes and the propagation speed of the communication medium. Besides this, it also estimates transmission, propagation and processing delay. Then comes the second phase that only activates if the Path mechanism stops showing results, indicating that an attacker may be interfering with it. When activated, this mechanism estimates the upper bound of the propagation delay. If this value is greater than the one calculated in the first phase then route hijacking attack is detected.

C. Summary

So far multipath communication has been mostly used for availability, resilience and reliability of networks, and fewer times for security purposes. MACHETE uses this concept to protect communication confidentiality, i.e. protect communication from eavesdropping attacks. For that it uses a combination of techniques: multihoming, MPTCP and overlay routing. MACHETE leverages MPTCP to split data from the source into multiple communication channels. When available, it uses multihoming, combined with an overlay network to guarantee that the packets are sent to the destination over disjoint paths. The routing paths must provide a high degree of diversity so that if an attacker gets access to a specific route, it does not imply that another route being used, which may overlap at some point, is also compromised.

As seen before, the BGP protocol has many security weaknesses that can be exploited by attackers. By understanding the vulnerabilities of the protocol, they can perform route hijacking attacks. These can be in the form of interception attacks, allowing for an attacker to eavesdrop Internet communications, without the source and destination's knowledge. However, they can be detected by analyzing some metrics, such as latency, routes that packets take and hop count. DARSHANA uses a combination of techniques to analyze such metrics and detect hijack attacks, requiring only data-plane information.

III. PREMIUM

This section presents PREMIUM, the private reactive multipath communication middleware. Section III-A will discuss

in detail the multipath communication mechanism. Section IV will address the route hijacking monitor. PREMIUM will be explained in section IV-A.

A. Multipath communication mechanism

We use MACHETE [2] as the base mechanism to implement multipath communication. As stated before, MACHETE is a multipath communication mechanism that aims to mitigate the impact of network security vulnerabilities, that may be exploited by unauthorized third parties. MACHETE addresses this issue by splitting streams of data into different physical paths.

This mechanism is composed of three main components:

- **Multipath manager:** keeps track of the overlay nodes and multipath devices.
- **Multipath device:** computer that communicates using MACHETE. Is also responsible for dynamically establishing paths and splitting the packets among them.
- **Overlay nodes:** nodes that compose the overlay network that forward the messages sent by the multipath devices.

In the whole process of communication, through MACHETE, we assume that each of the two multipath devices represent a sender and a receiver, because the communication is one way, since the destination is only able to respond to the packets received, which come from the overlays nodes, not the sending device.

The multipath manager is a tuple space, that implements Linda's generative coordination model [14]. The implementation used is called DepSpace. Whenever a multipath device or an overlay node start to run, they register themselves by inserting tuples in DepSpace [2], with their primary IPs and number of available network interfaces.

After the multipath devices are registered and want to initiate communication they start the process of path setup. The sender requests the active overlay nodes, from the multipath manager. Then when the sender receives all the available nodes, it chooses N overlay nodes. The sender sends a command to the overlay nodes to set them up to forward traffic to the receiver. After setting up the overlay network, the data transfer between the sender and receiver begins, thorough multiple routes.

When the data transfer step is finished, the sender notifies the overlay nodes to remove their rules. After all of the overlay nodes confirm the rules removal, the sender removes its own NAT rules.

Figure 1 shows an overview of the communication between all components of MACHETE, starting from the multipath devices and overlay nodes to the multipath manager until the data transmission to the receiver. The steps presented are the following:

1) The sender, overlay nodes and the receiver register themselves in the Multi-path Manager;
2) The sender queries for overlay nodes;
3) The sender chooses N overlay nodes and sets its NAT rules;
4) The sender sends NAT rules of overlay nodes;
5) The overlay node changes their NAT rules;
6) The overlay node confirms the set of rules to the sender;
7) The sender initiates data transfer, sending split messages to the Overlay nodes;
8) The overlay node forwards the messages to the receiver.

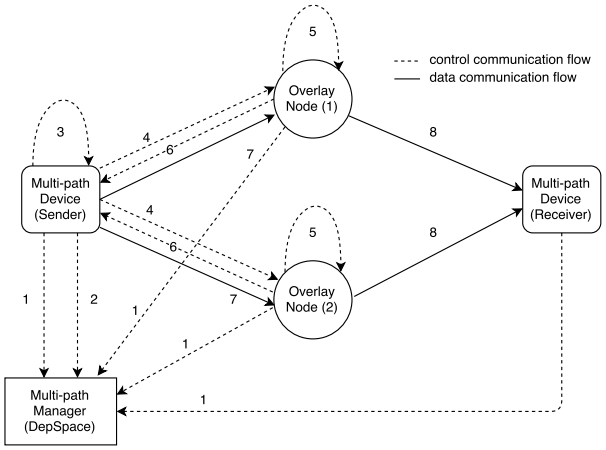

Fig. 1: Diagram for MACHETE overall communication (without the path tear down step).

IV. Route monitoring

Another component in this system is route monitoring. This is important because it is the components responsible for providing information to later trigger a reaction in the system. During route monitoring our solution can detect possible hijack attacks and act accordingly.

To monitor the overlay network we will use Darshana [3], a monitoring solution that detects route hijacking based solely on data-plane information. It works by continuously observing network information to detect route hijacking attacks.

The system applies active probing techniques, using the ones with lower overhead and reliability more often, and heavier and more reliable techniques when needed. It uses the Lat, Hop, Path and Prop mechanisms explained in section II-B2.

The architecture of Darshana is divided in four components that implement the aforementioned mechanisms:

- **Active Probing:** In this component the system uses Lat, Hop mechanisms and the first phase of Prop mechanism. It issues cryptographic pings and Paris traceroute [15]. RTT values are probed more often since it has the lowest overhead. Upon detecting an anomaly in this value the hop count is estimated to filter out small legitimate changes. Then it measures the propagation delay and the other RTT related latencies.
- **Path Similarity Detection:** If there is suspicion of an attack while running the previous component, Paris traceroute is executed. The path that contains most nodes is stored. Upon receiving enough results, the new path is compared with the last path stored. If any conclusive results are indicated in this phase then a route hijacking attack is declared.

- **Propagation Delay Validation:** If the previous component lacks conclusive results, this one starts. The maximum propagation delay is calculated, as well as the anomalous propagation delay, that uses the anomalous RTT value from the active probing component. If the ratio between these values is higher than a pre-established threshold, then a route hijacking is declared.
- **Hijacking declared:** this component notifies the user of this system that its traffic was hijacked.

The balanced combination of different techniques in terms of overhead and reliability, allows this system to be more resilient to network failures and attacker countermeasures. The evaluation results in [3] show that the combination of techniques manage to filter some false positives, proving to be able to accurately detect hijacking attacks. Also, the use of active probing techniques, allows the system to detect these attacks in near real-time, which is very important to our project.

Before using Darshana in the PREMIUM prototype, we had to do some improvements and add some features. One main feature that was added to the route hijacking monitor was the ability to send alerts to a remote server. These alerts can be sent many times within a certain time frame, because of that we do not have issues if we loose a certain alert. So for this we use the UDP protocol, that provides less overhead than TCP. This will be useful for the final solution explained in section IV-A. The hijack alerts are sent to the remote server indicating the type of alert, the source and destination IP, and the metric that accused the hijack attack.

A. PREMIUM

After understanding MACHETE and Darshana, we present PREMIUM, a reactive private multipath communication middleware. Our solution will leverage multipath communication to provide confidentiality, and react in near real time upon detecting possible hijack alerts, to compromise the least amount of data possible.

Our solution uses MACHETE to split data across multiple paths over an overlay network, so that in this way if an attacker has access to one of the paths it will be able to spy only the portion of data that traverses that path. PREMIUM will also have a path monitoring component, where it will use Darshana to monitor the sub paths used in the communication. Having this component that will monitor the overlay network, to look for possible hijack attacks, our solution will react upon detecting these events. We defined three main reactions that will be explained in section IV-A4.

In the next subsections we will go over other important components of PREMIUM.

1) Architecture and internal components: PREMIUM is based on MACHETE, so the architecture is similar to it. There is the Sender, the first source of traffic, and the Receiver which is the destination of this traffic. To forward data within network there are overlay nodes, that forward the traffic from the Sender to the Receiver.

The Multipath Manager is implemented by an improved version of DepSpace, which was the implementation used by MACHETE. This improved version is called DepSpacito.

To monitor the overlay network we wanted for Darshana to be topology-aware, where it is knows the location and position of the nodes within the network. This was essential so that it can monitor specific routes, namely, the ones between the Sender and the overlay Darshana as well as between these nodes and the Receiver. However, by running only on the Sender, DARSHANA cannot monitor the latter.

To solve this issue, PREMIUM runs an instance of Darshana in every node of the network presented in figure 1. Darshana runs client and server side. This defines the direction of path monitoring. The machine that runs Darshana client is the one that monitors the path from itself to the destination machine, that runs the server side of Darshana. In this architecture an overlay node will have another task, besides forwarding data stream from the source to the destination. It will also monitor the segment of path from itself to the Receiver.

To be aware of the hijack events, PREMIUM centralizes the alerts in a specific module, called Darshana Alert Receiver (DAR). This component is explored in the next subsection IV-A3.

Figure 2 shows the high level components of PREMIUM.

We use mostly TCP for communication, since this protocol are reliable and provide guaranties of delivery and ordering of packets. It is used for secure and reliable communication between the nodes of the system, except when sending hijack alerts to DAR.

2) Path monitoring: Path monitor is the component that is responsible for monitoring the overlay network. This means monitoring all paths used to split the data stream. These paths are divided into two sub paths. Since the overlay network is single hoped, as mentioned in section II-A2, we have to take into consideration that in between of each path we have one overlay node. So a full path consists of the segment between the source of traffic and overlay nodes plus the segment from the overlay nodes to destination.

PREMIUM runs the Darshana server side on the destination endpoints of each segment of the paths. This means that the server side will run on the overlay nodes and the Receiver machine. However the client side of Darshana must run on the source endpoints of each segment, which will be the Sender and the overlay nodes as well. Since Darshana client only handles monitoring one path at a time, we run N instances of this on the Sender, where N is the number of overlay nodes.

The path monitoring flow occurs in the Setup phase, previously explained in section III-A. This happens while setting up the Overlay Nodes. PREMIUM sends them instructions to run Darshana instances targeting the Receiver. Once the overlay nodes acknowledge the Sender, this device starts an instance of its own targeting the overlay node that was instructed at that moment.

To achieve this, the setup protocol from the Sender to the overlay nodes was improved. Before, the setup protocol consisted of just sending an expression with the data needed to

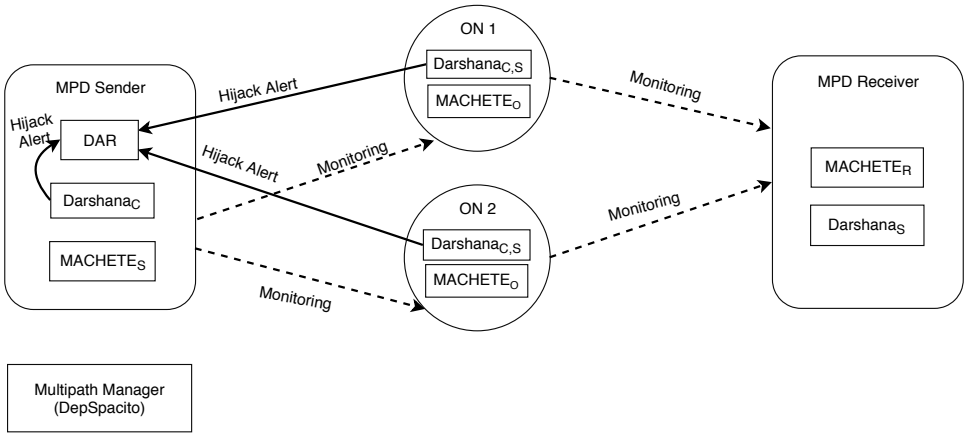

Fig. 2: The architecture of PREMIUM. The dashed lines represent the route monitoring performed by Darshana that runs in the sender multipath device and the overlay nodes. The solid lines represent the hijack alerts sent from Darshana to DAR.

build the `iptables` rules to forward traffic from the overlay nodes to the receiver. In addition to that expression, we are also sending the instruction to run Darshana.

At the moment, the setup protocol involves sending two expressions: the first for the forwarding rules, and the second is the necessary data to run an instance of Darshana aggregated. This data consists of the metric that is used by the Sender and its thresholds. It also has the IP and port of DAR, to send the hijack alerts.

All paths are being monitored by Darshana and send alerts to the DAR module running on the Sender.

3) Darshana Alert Receiver: Darshana was modified as explained in section IV to send alerts to a remote machine. This new feature is used for PREMIUM to receive this alerts and act accordingly. To receive these alerts we created a separate module called Darshana Alert Receiver (DAR). This component is responsible for receiving the alerts and saving them with its timestamp.

DAR uses UDP to receive these alerts. This protocol was used since we do not need reliability and guarantees to receive simple alerts. Once Darshana detects an alert sends it immediately to DAR, and continues to monitor the network, and thus sending alerts.

DAR has a threshold N, in which it sends a signal to after receiving more than N alerts. Regardless of the path that was hijacked, DAR sends a signal to the Sender, indicating that it should react.

4) Reactive component: The main component of PREMIUM is its reactor. By monitoring all the paths used for communication, in case of an attack, our solution can react upon it.

When DAR receives a certain amount hijack attack alerts, it notifies PREMIUM that a path currently being used was compromised. After this notification it has to decide what is going to be the reaction.

We defined three main reactions for PREMIUM in the case of an hijack attack, to one of its routes of interest, i.e., sub paths between the overlay nodes and the endpoints of the communication. These are the proposed reactions: 1. Close all connections; 2. Create another connection after shutting down communication through the compromised path; 3. Reuse current connections, after shutting down the compromised connection.

Currently PREMIUM is able to shutdown all communication when it receives a certain amount of alerts.

To trigger PREMIUM to react to an attack, DAR sends it a signal. DAR runs on a separate thread of the main program. After receiving a decent amount of alerts (this amount is defined as an argument given to the system), this thread send a signal to the parent process. After receiving this signal, the default action is to shutdown the connection and tear down the overlay network.

Figure 3 shows how the system proceeds after detecting an hijack attack within the overlay network until PREMIUM gets noticed and is able to make a decision.

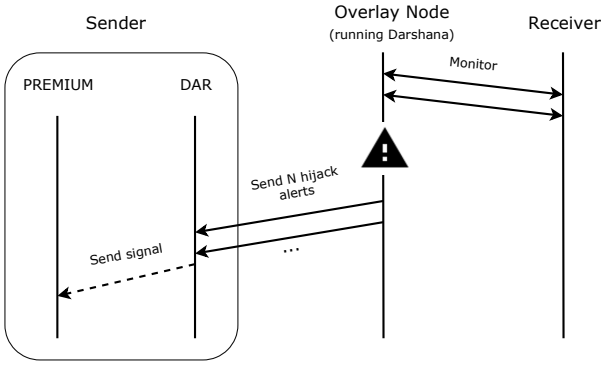

Fig. 3: Representation of sequence of events from the hijack detection from the monitoring nodes until PREMIUM is notified.

5) Prototype implementation details: PREMIUM uses mainly C programming language since its built on top of MACHETE [16] which uses it as well. To communicate with the Multipath Manager, it uses an adapter originally written

with Java programming language. This adapter interprets the requests made by the multipath devices and the overlay nodes, and communicates with the Multipath Manager.

Darshana [17] is kept as an isolated project, which PREMIUM uses to monitor multiple routes. The route monitoring system is developed with Java programming language, with the exception of cryptographic ping which is developed in Python, and kept as an isolated tool.

DepSpacito [18] is an independent project like Darshana. This version has some bug fixes and is adapted to the communication with the Multipath Manager adapter.

Figure 4 shows an overview of how PREMIUM is structured in the Multipath Device.

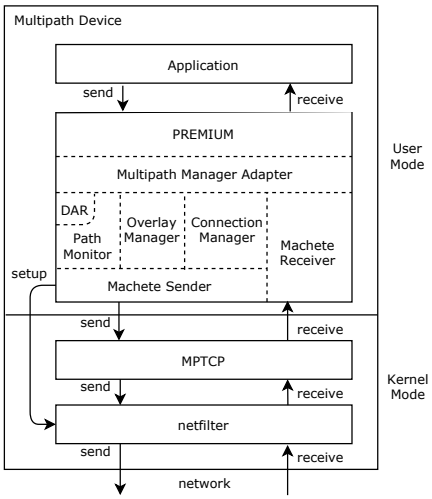

Fig. 4: PREMIUM multipath device architecture. Just as MACHETE, PREMIUM runs mainly in user mode, which then connects with MPTCP to send traffic to the Internet, in this case directly to the Overlay Nodes. Before the traffic leaves Sender's machine, the packets are filtered according to the rules established on the Sender.

One of the improvements made over MACHETE is the code structure, and separation of functions. As it is in figure 4, PREMIUM has multiple modules which have their own functions. The Overlay Manager is responsible for setting up the nodes of the Overlay network. The Connection Manager is responsible for establishing and shutting down connections. The Path Monitor is responsible for monitoring the multiple routes and interacting with DAR. As mentioned before the Multipath Manager Adapter is responsible to communicate with DepSpacito, the implementation of the manager.

The rules that are used to forward the traffic from the Sender to the Overlay Nodes and from these nodes to the Receiver, are set with `iptables` tool. That is why the netfilter, is involved in the communication sequence of the multipath device architecture.

The sequence of events of MACHETE, explained in section III-A: setup, teardown and data transfer are kept the same in PREMIUM, with the difference that they now involve the path monitoring setup and teardown.

V. Evaluation

This section presents the evaluation of PREMIUM. We present both a qualitative and an experimental evaluation.

A. Qualitative Evaluation

PREMIUM integrates improved versions of MACHETE (multipath communication) and Darshana (route monitoring).

In this Section we compare the original version of MACHETE to PREMIUM, and then the initial version of Darshana to the final version used in PREMIUM.

1) Comparison between MACHETE and PREMIUM: PREMIUM is an improvement over MACHETE, regarding communication confidentiality. The main improvement that distinguish PREMIUM from MACHETE is the ability to be aware of hijack attacks and react upon detecting them. Besides this, there are other improvements related to its usability and several bug fixes.

Unlike MACHETE, our new solution can run in parallel with other instances of itself. The overlay network settings are not corrupted by using multiple instances of our reactive mechanism, whereas with MACHETE only worked with one instance at a time, when using the same subset of overlay nodes.

Another advantage over MACHETE is that PREMIUM provides more liberty of configuration. It allows choosing parameters such as ports and IPs to use, as well if it needs to use and external service to get the external IPs of the machines running our solution. It also has the first version of a well defined API that lets create other types of applications for different use cases. PREMIUM can be used for more use cases other than the file transfer use case, such as in a case of an HTTP proxy.

Initially, MACHETE was only used within the file transfer application context, making it difficult for us to verify if it had the ability to have bidirectional communication, i.e., have communication in both ways within the same connection between the sender and the receiver. After we redesigned the code structure, that allowed us to define a simple API for MACHETE, we tested our solution with a message exchange application, which consisted of the sender sending a message to the receive, then the receiver would send back the message in upper case mode. We then verified that MACHETE allows bidirectional communication within a current connection. These capability was kept in PREMIUM.

Both MACHETE and PREMIUM rely on MPTCP for the multipath communication, to split data streams of data into multiple paths. The overlay network and multihoming continue to be very important techniques to provide diversity for the multiple routes, thus it is still used in PREMIUM.

Table I presents the comparison of features between MACHETE and our final solution, PREMIUM.

2) Comparison between original and improved version of Darshana: For route monitoring we used an improved version of the original Darshana. This improved version was crucial for *Experimental Evaluation of Route Monitoring*, Bachelor's Thesis by Markus Hinz [19]. Markus' work consisted of an

Mechanism/ Feature	MACHETE	PREMIUM
Use of MPTCP	✓	✓
Overlay network	✓	✓
Multihoming	✓	✓
Bidirectional communication	✓	✓
Reusable API	✗	✓
Parallel communication	✗	✓
Reaction to hijack attacks	✗	✓

TABLE I: Comparison of features between MACHETE and PREMIUM.

extensive analysis of Darshana, which led to deciding on configuration thresholds that result in high detection and low false positives. This study used realistic and historic traceroute data to perform measurement ans and statistics.

We modified the original version of Darshana to allows us to measure an isolated metric, such as a mechanism with heavier or lower overhead. An example of this is the ability to use Darshana to measure just the path similarity of the routes between the source and destination, which is considered a mechanism with a higher overhead than measuring hop count or network latency. However, Darshana still has the option to run in full monitoring mode in case we want to use the original sequence of measurements to detect hijack attacks.

Another improvement made was the ability to configure the metric thresholds as arguments. In this way we can easily test the prototype, with different thresholds, without changing the source code. Also, Darshana was also optimized to run in a local network environment.

The initial version of Darshana already had the ability to measure four metrics: latency, hop count, path similarity and propagation delay. These metrics were used in four detection mechanism presented in section II-B2: Lat, Hop, Path and Prop. The improved version still utilizes these mechanisms to detect an hijack attack.

Cryptographic ping was completely isolated from the main path monitor. It also became more configurable, providing flexibility to choose its arguments.

Table II shows the comparison between the initial version of Darshana and the improved one.

Mechanism/ Feature	Original Version	Improved Version
Measure Latency (Lat)	✓	✓
Measure Hop Count (Hop)	✓	✓
Measure Path Similarity (Path)	✓	✓
Measure Propagation Delay (Prop)	✓	✓
Full monitoring mode	✓	✓
Isolated Measurements	✗	✓
Cryptographic ping isolated	✗	✓
Configuration of thresholds	✗	✓
Work on local network environment	✗	✓

TABLE II: Comparison of features between the original and the improved version of Darshana.

3) Summary: In summary, we made significant improvements to MACHETE and Darshana prototypes, that were crucial for further development. Without some of these improvements made on the original versions of MACHETE and Darshana we could not move forward with PREMIUM, since we needed stable versions of these prototypes.

B. Experimental Evaluation

In this section we present an overview of the testbed of the evaluation and the results we got from testing PREMIUM when deployed in the cloud.

1) Testbed: During the development and testing of the prototypes, we developed a fully working environment using Virtual Box [20] to deploy PREMIUM and its prototypes in virtual machines. All the machines used during the experiments in local settings had the same configurations.

The virtual machines (VM) used Debian GNU/Linux 8.9 (jessie) operating system, with 2 GB of memory, 1 CPUs. All VMs only required a minimum of 1 network interface, with the exception of the one that ran the Sender which needed at least 2 network interfaces.

To test the hijack alerts, we used Darshana option to only monitor latency differences, to cause more false positives and focus on setup and teardown times. To evaluate the whole system, we used a simple application that sends a message and receives its echo in a loop. After ensuring that the prototype of PREMIUM worked properly, in a local network setting, we deployed it in a cloud provider.

2) Cloud Deployment: For a wide network deployment we used Google Cloud Platform (GCP) [21] to place the hosts used in the experiments.

The machines configuration used in GCP, were the same as to the ones mentioned except for the memory. The flavor we chose for all the machines provided 3.75 GB of memory.

Due to some constraints in network configurations of the Google Cloud Platform we were not able to deploy more overlay nodes in diverse geographical locations.

Figure 5 represents the placement of the nodes. To understand the impact of the route monitoring component, we measured the time for the main actions of the setup and teardown phase.

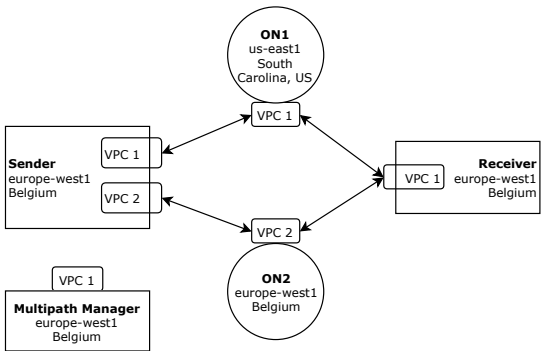

Fig. 5: Representation of the placement of the nodes in Google Cloud Platform. We had to use two VPCs, one for each network interface the Sender had. All the nodes belong to the VPC1 except the overlay node 2 that has to be connected with sender's second interface, VPC2.

For this evaluation, we wanted to have diverse paths for the overlay nodes, that forward data from the sender to the receiver. This diversity is important to preserve the security characteristics of having physical disjoint paths [2]. Having the limitations, already mentioned, we placed the overlay node 2 in Belgium, along with the sender, receiver, and the multipath manager, while the overlay node 1 was placed in South Carolina, US.

We ran PREMIUM in this deployment environment 25 times. In the next section we discuss the results regarding the time distribution of each step involved in the setup and teardown phases. We also study the impact of adding path monitoring to these phases.

3) Operation time breakdown: Before understanding the impact of the path monitoring in terms of time, we first want to understand at a higher level how the time is distributed in the multiple steps of PREMIUM's configuration.

The setup time includes all the time since PREMIUM starts its execution until the connection with the receiver actually starts. The sequence of events since the beginning of the execution can be defined by the following steps: 1. Fetch overlay nodes' data from multipath manager; 2. Check responsive overlay nodes; 3. Fetch receiver information from multipath manager; 4. Flows configuration (setup and teardown).

The teardown time consists of the time of sending instructions to the overlay nodes of the network to stop path monitoring and disable forwarding data to the receiver. Figure 6 presents a comparison between the duration of setting up and tearing down the overlay network.

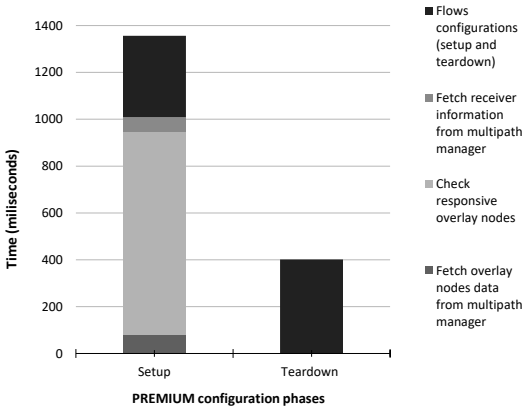

Fig. 6: Representation of time distribution for the setup and teardown phase.

We can see that most of the setup time is spent on pinging the overlay nodes for connectivity. Also that the setup and teardown of the routes are very close.

4) Time dedicated to Path monitoring within Setup: To understand the impact of the path monitoring, we should breakdown the time from setup the setup mentioned above, flows configuration (setup and teardown) to check how much time is spent on setting up path monitoring for each one of the routes used during the connection with the receiver.

Figure 7 presents the times spent in the configuration of the forwarding rules and path monitoring in the sender and overlay nodes machine.

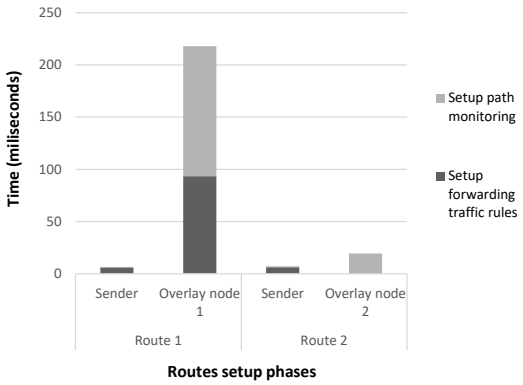

Fig. 7: Representation of time distribution for the setting up forward rules and path monitoring both on the sender and each overlay node side. Note that the overlay node 1 located in South Carolina, US and both the sender and the overlay node 2 are located in Belgium.

By observing figure 7, we can first observe significant difference between the overall setup time of the first route is significantly greater than the second route setup time. The differences between the times of sender setup and overlay node setup, for each route, can be explained by the fact that setting up the sender, does not require network connection establishment, since the setup is local to the sender. In the case of setting up the overlay nodes, network connections are required, since this phase consists of sending commands to the overlay node and waiting for the acknowledgment. The need for a network connection, causes this phase to be affected by network latencies.

Another observation we can see is that by comparison, the time to setup the overlay node 1 is significantly higher than the time to setup the overlay node 2. This might be because the overlay node 2 and the sender are in the same zone (*europe-west1*) in Belgium, whereas the overlay node 1 is far from the sender, in South Carolina, US.

Lastly, we can also observe that the time to setup the NAT rules and setup path monitoring in the overlay node 1, are very similar. This can lead us to think that the time to setup the overlay nodes might doubled because of this added step, path monitoring, to the setup of the sender.

5) Time spent on Path monitoring within Teardown: In this section we analyze the teardown time distribution, focusing on the overlay nodes.

The time to teardown an overlay nodes includes the time to send a command to remove NAT rules in the overlay node and receive the acknowledgment, plus the time to send the command to stop monitoring the sub path from the overlay node to the receiver.

Figure 8 presents the distribution of time for tearing down the overlay nodes.

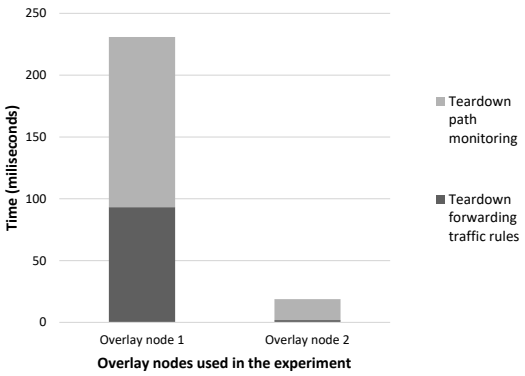

Fig. 8: Representation of time distribution for the teardown of the forwarding rules and path monitoring on each overlay node used for the connection with the receiver. Note that the overlay node 1 located in South Carolina, US and both the sender and the overlay node 2 are located in Belgium.

We can observe that the time between the overlay node 1 and 2, have significant differences. However, this distinction can be explained by the fact that the first overlay node is in another region, much more distant from the sender than the overlay node 2. We can also see that for the distant overlay node the time for forwarding rules teardown configuration is very similar to the time to teardown the path monitoring for the same node. This can lead us to think that the impact of path monitoring for the teardown may not be significant. In this situation it seems that it doubles the overall time of tearing down a flow.

6) Summary: These experiments were not enough to make strong conclusions about the prototype's performance. We needed more variation in the number of overlay nodes and its location to see the evolution of the time distribution to better estimate the impact of combining path monitoring with the multipath mechanism, in PREMIUM. Nonetheless, we could see some indications of its impact. By analyzing the whole experimental evaluation we got some indications that the addition path monitoring could double both the setup and teardown times of the nodes. Although, this may have a significant impact on these times, we could also see that the nodes setup time is just a fraction of the total setup time of the sender, since this multipath device spends a lot of time getting the information about the nodes of the overlay network and the receiver.

VI. Conclusion

Our goal is to protect communication by using multipath communication to avoid an attacker of having access to information by controlling the communication path. We also want to minimize the amount of data compromised in case of an hijack attack. So we presented PREMIUM, private reactive multipath middleware. PREMIUM is able to split data across multiple paths, on top of an overlay network, while monitoring the routes being used to be aware of possible hijacking attacks. Our solution is also able to stop the whole communication, when it detects an hijack attack.

References

[1] A. Ford, C. Raiciu, M. J. Handley, O. Bonaventure, and C. Paasch, "TCP Extensions for Multipath Operation with Multiple Addresses," Internet Engineering Task Force, Internet-Draft draft-ietf-mptcp-rfc6824bis-07, Oct. 2016, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-mptcp-rfc6824bis-07

[2] D. Raposo, M. L. Pardal, L. Rodrigues, and M. Correia, "MACHETE: Multi-path communication for security," *Proceedings of the 2016 IEEE 15th International Symposium on Network Computing and Applications (NCA)*, no. 3, pp. 60–67, 2016. [Online]. Available: http://ieeexplore.ieee.org/document/7778594/

[3] K. Balu, M. L. Pardal, and M. Correia, "DARSHANA: Detecting route hijacking for communication confidentiality," *Proceedings of the Network Computing and Applications (NCA), 2016 IEEE 15th International Symposium*, pp. 52–59, 2016.

[4] SafeCloud project, available from http://www.safecloud-project.eu. Last time accessed: May 16, 2017.

[5] W. Lou and Y. Kwon, "H-SPREAD: A hybrid multipath scheme for secure and reliable data collection in wireless sensor networks," *Proceedings of the IEEE Transactions on Vehicular Technology*, vol. 55, no. 4, pp. 1320–1330, 2006.

[6] J. Deng, R. Han, and S. Mishra, "INSENS: Intrusion-Tolerant Routing in Wireless Sensor Networks," University of Colorado, Department of Computer Science, Tech. Rep., 2006.

[7] J. Han and F. Jahanian, "Impact of path diversity on multi-homed and overlay networks," *Proceedings of the International Conference on Dependable Systems and Networks, 2004*, no. Dsn, pp. 29–38, 2004. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1311874

[8] A. Akella, J. Pang, B. Maggs, S. Seshan, and A. Shaikh, "A comparison of overlay routing and multihoming route control," *Proceedings of the ACM SIGCOMM Computer Communication Review*, vol. 34, p. 93, 2004.

[9] J. Han, D. Watson, and F. Jahanian, "Topology aware overlay networks," *Proceedings of the IEEE INFOCOM*, vol. 4, pp. 2554–2565, 2005.

[10] J. Schlamp, G. Carle, and E. W. Biersack, "A forensic case study on as hijacking," *Proceedings of the ACM SIGCOMM Computer Communication Review*, vol. 43, no. 1, p. 5, 2013. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2479957.2479959

[11] P. Vervier, O. Thonnard, and M. Dacier, "Mind Your Blocks: On the Stealthiness of Malicious BGP Hijacks," *Proceedings 2015 Network and Distributed System Security Symposium*, no. February, pp. 8–11, 2015. [Online]. Available: http://www.internetsociety.org/doc/mind-your-blocks-stealthiness-malicious-bgp-hijacks

[12] K. Butler, T. R. Farley, P. McDaniel, and J. Rexford, "A survey of BGP security issues and solutions," *Proceedings of the IEEE*, vol. 98, no. 1, pp. 100–122, 2010.

[13] B. Augustin, X. Cuvellier, B. Orgogozo, F. Viger, M. Latapy, R. Teixeira, and T. Friedman, "Avoiding traceroute anomalies with Paris traceroute," *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pp. 153–158, 2006.

[14] D. Gelernter, "Generative communication in Linda," *ACM Transactions on Programming Languages and Systems*, vol. 7, no. 1, pp. 80–112, 1985. [Online]. Available: http://portal.acm.org/citation.cfm?doid=2363.2433

[15] J. Aug, B. Augustin, et al., Paris Traceroute, available from https://paris-traceroute.net/. Last time accessed: April 15, 2018.

[16] MACHETE hosted on GitHub, available from https://github.com/inesc-id/MACHETE. Last time accessed: April 29, 2018.

[17] Darshana hosted on GitHub, available from https://github.com/inesc-id/darshana. Last time accessed: May 3, 2018.

[18] DepSpacito hosted on GitHub, available from https://github.com/inesc-id/DepSpacito. Last time accessed: April 29, 2018.

[19] Markus Hinz. Experimental Evaluation of Route Monitoring. Bachelor's Thesis. Technical University of Munich. May, 2018.

[20] Oracle VM VirtualBox, available from https://www.virtualbox.org/. Last time accessed: May 10, 2018.

[21] Google Cloud Computing, Hosting Services & APIs, available from https://cloud.google.com/. Last time accessed: April 21, 2018.