# DARSHANA: Detecting Route Hijacking for Communication Confidentiality

Karan Balu  Miguel L. Pardal  Miguel Correia
INESC-ID, Instituto Superior Técnico, Universidade de Lisboa – Lisbon, Portugal
{karan.balu, miguel.pardal, miguel.p.correia}@tecnico.ulisboa.pt

***Abstract*—The Border Gateway Protocol (BGP) plays a critical role in the Internet providing connectivity to hosts across the world. Unfortunately, due to its limited security, attackers can hijack traffic by generating invalid routes. Some detection systems for route hijacking have been presented, but they require non-public information, high resources, or can easily be circumvented by attackers. We propose DARSHANA, 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. DARSHANA uses active probing techniques that enable detection in near real-time. By using diverse methods, DARSHANA can still detect attacks even if the adversary manages to counter some techniques. We show that our solution allows effective detection of many hijacking attacks by emulating them using PlanetLab and Amazon AWS.**

## I. Introduction

The Internet is a network composed by many interconnected networks. Administrative network domains are called *Autonomous Systems (AS)*, and the routing between these autonomous systems is handled by the *Border Gateway Protocol (BGPv4)* [1]. Each AS contains one or more *Internet Protocol (IP) prefixes*, whereas each prefix is an identifier for a subnetwork. If some AS wants to provide connectivity between its IP prefixes and other ASes, it will announce those prefixes to those ASes. Each AS contains one or more routers configured with BGP, known as BGP speakers. Each speaker contains forwarding tables that provide the information necessary to forward a packet based on the destination and the prefix available in the table. BGP speakers send UPDATE messages to other BGP speakers in order to announce or withdraw routes. Upon receiving these update messages, an AS selects the best route to a certain prefix based on its internal policy.

Although BGP plays an essential role in the Internet, it has considerable limitations in terms of security. One example of its lack of security happened on August 2013 when a company called Hacking Team helped the Italian police regain control over computers that were being monitored by them. Hacking Team worked with an Italian Web host called Aruba announcing to the global routing system 256 IP addresses that it did not own. This caused all the traffic directed to the 256 IP addresses to be redirected to the Hacking Team. This was the first known case of an ISP performing a *route hijacking attack* intentionally [2].

978-1-5090-3216-7/16/$31.00 ©2016 IEEE

These security problems mainly come from the potential to interfere with route announcements in order to corrupt BGP routing. Attackers can exploit this vulnerability to claim ownership of victim prefixes and announce them to their upstream providers. Providers that do not verify the origin of the announcements may end up injecting these in to the global routing system, which leads network packets to reach incorrect destinations. In some cases, attackers may intercept traffic and forward it to its destination, compromising *confidentiality* without being noticed.

The vulnerability of the BGP protocol has been well-known for over two decades. Several solutions have been proposed, but none is widely adopted and deployed. These solutions mainly fall into filtering and cryptography methods [3]–[6], which require changes in routers configurations, router software and a public key infrastructure. Others proposals [7]–[10] rely on passive monitoring of BGP data, so they are easier to deploy; however, they suffer from high false positive rate, since they access public registries that are frequently outdated. Finally, there are systems that use only data-plane information, by executing active probing, but can easily be bypassed [11], or require vantage points [12].

We present DARSHANA (or DaRsHANa, from Detecting Route HijAckiNg – in Sanskrit, Darshana means to see, vision or glimpse) that works by continuously observing network information to detect route hijacking attacks. Our main goal is to detect if Internet traffic is diverted to be eavesdropped in arbitrary places around the world, when the adversary has no access to the path normally taken by the traffic. Therefore, the security property we are most interested in is communication *confidentiality*. DARSHANA has the advantage of being implemented in the data-plane, above the OSI network layer, therefore it can be implemented in terminals connected to the Internet, instead of being specific to Internet Service Providers (ISP) and large Internet companies. DARSHANA uses a set of monitoring techniques like: traceroute, latency measurements and IP traceback mechanisms that can effectively monitor the routes that packets are taking. Ultimately, this allows detecting route hijacking that could be used to eavesdrop on a communication to break confidentiality. We do not intend to substitute the use of best practices to configure BGP, or several prevention mechanisms that have already been proposed like [3]–[6], [8], [13].

The system applies active probing techniques which enables the detection in near real-time. The order of execution of these techniques is defined in terms of overhead and reliability: techniques with lower overhead and reliability are executed more often; when needed, heavier, more reliable techniques are used. The system does not depend solely on a specific technique to be able to accurately detect attacks.

We performed an experimental evaluation by deploying nodes in PlanetLab and Amazon AWS. We show that nodes can identify when traffic is being hijacked, although this is more difficult if the hijacker is close to the source.

The contributions of this paper are three-fold. First, we propose the design and implementation of DARSHANA, a route hijacking detection system that is accurate, does not need access to privileged information, does not require changes in routers software, is redundant enough to deal with attacker countermeasures, and does not need vantage points. Second, we present a new mechanism that uses the propagation delay in order to detect route hijacking. Third, we analyze and conduct experiments in large scale environments to evaluate DARSHANA.

## II. Background

BGP does not ensure that BGP routers use the AS number they have been allocated, or that the ASes holds the prefixes they originate. Therefore, a router can be configured to advertise a prefix from an address space belonging to another AS in an action known as route hijacking or IP prefix hijacking [13]. This action can happen in the following forms:

*Hijack the entire prefix.* The hijacker announces the exact prefix of the victim, meaning that the same prefix has two different origins.

*Hijack only a sub-prefix.* The offender announces a more specific prefix from an already announced prefix (e.g., the victim announces 200.200.0.0/16, the attacker 200.200.200.0/24). Due to the longest prefix matching rule, ASes that receive these announcements may direct traffic towards the wrong AS.

These forms of attacks can impact routing, leading to:

*Blackhole.* An AS drops all the packets received. The Pakistan Telecom / YouTube incident originated a blackhole where all the traffic sent to YouTube was redirected to Pakistan Telecom. Since there was no working path back to YouTube, Pakistan Telecom was forced to drop all packets [14].

*Interception.* The attacker announces a fake route to an AS, that forwards traffic of the victim to the original server. The contents of the intercepted traffic can be analyzed/changed, before sending it to the legitimate destination [15]. This type of attack requires an untampered working path that will route the traffic back to the legitimate destination.

BGP security procedures today consist mainly on filtering suspicious BGP announcements, e.g., announcements that contain loopback addresses or addresses that are not owned by the AS that announced it. The problem of this approach is that detecting invalid route announcements is more challenging when the offending AS is several hops away.

An accurate routing registry would have prefix ownership, AS-level connectivity and routing policies enabled in each AS, helping ASes to verify the legitimacy of the advertisements that they receive. The drawbacks of this model mainly include, the lack of desire of ISPs to share their proprietary routing policies.

In this work, we focus on interception attacks and propose a solution that does not rely completely on Internet registries.

## III. DARSHANA

We introduce DARSHANA a route hijacking detection system that uses only data-plane information.

### A. Mechanisms used in the system

This section presents the mechanisms used in DARSHANA. We indicate the short names we use for each between parentheses (e.g., Lat for the first mechanism). Table I shows a summary of the mechanisms.

*1) Monitoring network latency (Lat):* One of the metrics used in our system is the RTT (round trip time). Each node that is monitoring another (node) keeps information about the total time that each packet takes from source to destination and from destination to source. In a hijacking event the end-to-end latency between a certain source and a destination tends to change significantly. Measuring the RTT has some benefits like low overhead and the fact that time is a factor that is hard for an attacker to evade. On the other hand, an increase in RTT is hard to distinguish from network congestion.

We designed a new version of ping that we denote as *cryptographic ping*. The objective is to avoid having an adversary respond to a ping request earlier, before the request reaches the destination, leading to readings of RTT that are lower than the real value. The new mechanism works as follows. The machine that is monitoring *A* marks time and sends a nonce to a machine that is being monitored *B*. *B* will cipher the nonce with its private key and send it back. *A* marks the time again and will verify the received signed nonce by applying the public key of *B*. If the nonce matches, *A* calculates the round trip time by subtracting the first marked time from the last marked time. Without this ping, the hijacker, since he has hijacked the traffic, could answer to the ping probes sooner, ultimately fooling the system. This way we can guarantee authenticity and uniqueness. This requires the server to run code and share its public key.

*2) Estimating hop count (Hop):* We propose adding the hop count, the number of intermediate devices between a source and a destination, as one more criteria to detect a route hijacking attack. According to [12] the hop count to a certain destination generally remains unchanged over time. When a prefix is hijacked, the hop count tends to change. In an interception attack, the traffic takes a detour to the AS of the hijacker, then it is forwarded to the legitimate destination. This deviation can change significantly the hop count if the hijacker is far from the source, which is likely due to the size of the Internet. In contrast to the RTT, the hop count is not affected by congestion. However, other less frequent events

TABLE I
THE METHODS, BENEFITS AND DRAWBACKS OF THE MECHANISMS USED BY DARSHANA PRESENTED IN SECTION III-A

| Mechanism | Detection | Benefits | Drawbacks |
|---|---|---|---|
| Monitoring network latency (Lat) | High latency could mean traffic hijack. | Easy to measure. | Latency is also affected by congestion so it does not indicate hijacking with certainty. |
| Estimating hop count (Hop) | The hop count is usually stable, so a high increase in hop count could be induced by traffic hijack. | Usually stable. | Link failures and legitimate route changes may trigger alteration in the network topology. |
| Calculating path similarity (Path) | Paths may end up showing significant disagreement when traffic is hijacked. | Filters small legitimate route changes. | Not all route changes are the result of traffic hijack. |
| Monitoring propagation delay (Prop) | Propagation delay gives the time that a bit takes in the wire, meaning that in a hijacking event this value may show an anomolous value. | Provides insights about the attack even when traceroute does not give results. | Requires a period of initialization, to estimate all the other latencies. |

like link failures and operational route changes may cause it naturally.

*3) Calculating path similarity (Path):* The system tracks the path that packets are following. It periodically stores the path obtained using traceroute and translates the IP addresses found to autonomous systems numbers (ASN). This mapping increases accuracy, because we only need one router from a autonomous system to correctly obtain a path that packets are taking. The correlation between the new path measurement and the previous path measurement may provide insights about the occurrence of the attack. In a hijacking event, since the traffic has taken a detour, the paths measured may end up showing significant differences. The level of this difference sets apart legitimate route changes and hijacking situations. On the contrary, legitimate changes are not expected to result in a dramatic route change.

*4) Monitoring propagation delay (Prop):* We propose a new technique that isolates the propagation delay from the RTT and uses this metric to declare a route hijacking. This technique is divided into two phases. The second phase is activated only if the system stops obtaining results from the Path mechanism, indicating an attacker is interfering with this mechanism.

*Phase one.* Consider that the RTT can be decomposed in the following delays: transmission delay ($\sigma_{trans}$), propagation delay ($\sigma_{prop}$), queuing delay ($\sigma_{queue}$) and processing delay ($\sigma_{proc}$) as depicted in $RTT = \sigma_{trans} + \sigma_{prop} + \sigma_{queue} + \sigma_{proc}$. The propagation delay is the time that a bit takes in the communication medium from a node to another node. This delay can be calculated as the ratio between the link length and the propagation speed on that medium.

The system uses the IP addresses of the origin and the destination to obtain their approximate geographical coordinates. The link length is calculated by computing the shortest distance between both. For the propagation speed, we make a conservative approximation by considering that all nodes are connected with fiber-optics, which has higher propagation speed than alternative media (copper, air). We use the usual approximation that fiber optics operates at 2/3 the speed of light [16]. The minimum possible propagation delay is given by formula 1, where $o$ represents the origin, $d$ is the destination and $c$ is the speed of light:

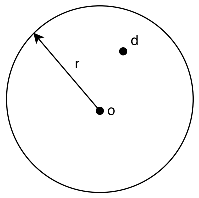


Fig. 1. The maximum propagation delay is represented as a circle defined by the source $o$ with radius $r$ where the destination $d$ is inside of the circle.

$$\sigma_{prop} = \frac{3}{c} \times ShortestDistance(o, d) \tag{1}$$

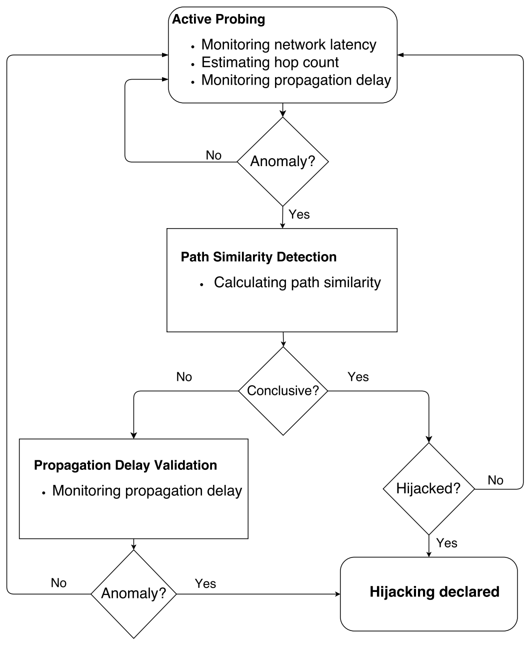
Besides the propagation delay, the system estimates the sum of the others latencies ($\sigma_{trans,queue,proc}$) by $\sigma_{trans,queue,proc} = RTT - \sigma_{prop}$.

*Phase two.* When the system obtains an anomalous RTT and stops receiving results from Path, it selects the minimum value of $\sigma_{trans,queue,proc}$ and the maximum value of the RTT estimated. By $max(\sigma_{prop}) = max(RTT) - min(\sigma_{trans,queue,proc})$, we obtain an upper bound on the value of the propagation delay. This allows drawing a circle around the source with a radius $r$ that represents the maximum propagation delay (represented in Figure 1). If the distance between $d$ and $o$ is greater than $r$ we detect a route hijacking. This mechanism allows detecting route hijacking even if the Path measurements cease to exist. However, it requires a period of initialization, to estimate the different latencies.

## B. System operation

This section presents the operation of DARSHANA. Figure 2 divides the mechanisms presented in the previous section in components and presents their relations. DARSHANA has the following components:

*Active Probing.* In this component three mechanisms come into play: Lat, Hop, and Prop (first phase). The system constantly takes values for RTT, hop count and the path that packets are taking. The system probes the RTT more often because this is the mechanism with the lowest overhead. Upon detecting an anomaly in the RTT the system passes to more reliable mechanisms, as this anomaly could be caused by temporary congestion in the network. The next mechanism is estimating the hop count, for the reasons explained in Section III-A2. This metric is more reliable than Lat so it is used to filter out small legitimate changes. This component also executes the first phase of monitoring propagation delay, calculates this delay with the shortest distance in a straight line between the source and the destination and estimates the other latencies belonging to the RTT.



Active Probing
Monitoring network latency
Estimating hop count
Monitoring propagation delay
Anomaly?
No
Yes
Path Similarity Detection
Calculating path similarity
Conclusive?
No
Yes
Propagation Delay Validation
Monitoring propagation delay
Hijacked?
No
Yes
Anomaly?
No
Yes
Hijacking declared

Fig. 2. Fluxogram of DARSHANA, with the mechanisms organized in different components

*Path Similarity Detection (Path).* Traceroutes with different protocols (ICMP, UDP, TCP) are issued. The system uses different protocols because routers may be configured to block certain protocols [17]. The path that contains the most nodes is chosen and stored. If enough results were received, then the new path will be compared with the last path obtained by the Active Probing. Disagreement above a certain threshold may indicate the existence of the attack.

*Propagation Delay Validation (Prop, second phase).* In case no conclusive results are received from path similarity detection, the $max(\sigma_{prop})$ and the $anomalous(\sigma_{prop})$ are calculated. The maximum propagation delay is computed by $max(\sigma_{prop}) = max(RTT) - min(\sigma_{trans,queue,proc})$. The anomalous propagation delay is calculated with $anomalous(\sigma_{prop}) = anomalous(RTT) - max(\sigma_{trans,queue,proc})$. This calculated propagation delay is compared with $max(\sigma_{prop})$.

*Hijacking declared.* Upon conclusion of the method chosen, an analysis is made and presented to the sender of the traffic.

## C. The system in detail

In this section, we present the implementation decisions of DARSHANA.

*Active probing.* DARSHANA issues cryptographic pings and Paris traceroute [18] probes with different periods. Paris traceroute is known to evade anomalies like loops, cycles and diamonds. These anomalies stem from the fact that a load balancer sends probes of traceroute to different interfaces based on the header of the probes. By not varying the fields used by a load balancer, Paris traceroute enables probes to be forwarded in the same interface even if load balancers exist.

Three values are obtained by executing traceroute: hop count, traffic path and propagation delay. For calculating the hop count we use traceroute. We only need to execute a partial traceroute with a TTL that is close to the destination in the majority of times. TTL = 1 is only used when we do not know about the destination.

We characterize the traffic path in terms of a set of autonomous systems, so each node of the result of the traceroute is mapped to the corresponding autonomous system using the CYMRU database [19].

Finally, the propagation delay is calculated by first, translating the IPs of the source and destination to geographical coordinates using MaxMind database [20], then the shortest distance is calculated between them and passed to the propagation delay by using the formula presented in Section III-A4.

Each iteration of the cryptographic ping gives a new sample of RTT and by subtracting the RTT with the propagation delay, we estimate the other latencies of the RTT.

*Path Similarity Detection.* New samples of RTT and hop count are compared with the exponential weighted moving average of past samples, the formula for the average is the following: $sample = (1 - \alpha) \times sample + \alpha \times sample_{new}$. The moving average allows DARSHANA to adapt to the normal changes in the network. If the quotients between the new samples of both RTT and hop count with the exponential weighted moving average passes certain defined thresholds $T_{Lat}$ and $T_{Hop}$, Paris traceroutes are issued to the destination in an attempt to reveal the cause of the anomalies. If there are enough elements in the resulting path, then this path is compared to the last path stored. The comparison of these two paths can be computed from $path$ and $path'$ using the Sorensen-Dice coefficient: $sim = 2|path \cap path'|/(|path| + |path'|)$. This gives the similarity in a number that ranges from [0,1]. 0 means that there is no similarity at all and 1 means that the items of the two paths are the same. If the similarity is below a threshold $T_{Path}$, then a route hijacking is declared.

*Propagation Delay Validation.* In case the traceroutes executed in the previous module do not produce any results, DARSHANA calculates the $\sigma_{prop}$ with the RTT and $\sigma_{trans,queue,proc}$ that were estimated. More precisely, the system will compute the $max(\sigma_{prop})$, by subtracting the $max(RTT)$, found before the anomaly, and the $min(\sigma_{trans,queue,proc})$. This computed delay will be compared with $anomalous(\sigma_{prop})$ resulted from the subtraction of the $anomalous(RTT)$ with the $max(\sigma_{trans,queue,proc})$. If $\frac{anomalous(\sigma_{prop})}{max(\sigma_{prop})}$ is higher than a defined threshold $T_{Prop}$, then a route hijacking is declared.

## IV. Evaluation

Simulations of prefix hijackings were conducted to validate our proposed implementation in terms of performance and cost. The objective of the experimental evaluation is to answer two important questions: (1) How effective is DARSHANA in detecting attacks? (Section IV-B) (2) How many times is DARSHANA forced to execute techniques with higher overhead in normal conditions when there is no attack? (Section IV-C)

The experiment was done in PlanetLab Europe [21] and AWS EC2 [22]. PlanetLab offers a geographically diverse set of nodes which provides more choice to build scenarios. However, the restriction of only being able to access nodes from Europe limits the testing in larger scale scenarios. AWS permits access to instances in different continents but does not provide much geographical diversity. With PlanetLab we use nodes from Portugal (POR), Ireland (IRE), France (FRA), Germany (GER) and Poland (POL). From AWS we used instances from N. Virginia (VA), N. California (NA) and South Korea (S. Korea). Figure 3 shows a world map with all the nodes used from PlanetLab and AWS marked in black circles and squares, respectively.

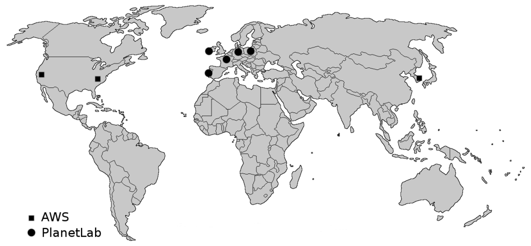


Fig. 3. Nodes used from PlanetLab and AWS

### A. Simulating route hijacking attacks

Before we present the tests done, it is important to explain how the simulation of the attacks is made. We simulate only the interception attack, because the blackhole attack ends up being just an interruption of communication, therefore it is easy to detect. In order to simulate the attack, we need three nodes: one node that is the source of the Internet traffic; a node that will serve as the destination; and another node that is trying to hijack traffic by receiving it and then sending it to the legitimate destination.

### B. Performance of the system

In order to evaluate the performance of DARSHANA, we measured the percentage of times that DARSHANA detects existing attacks and assess the false positives in different scenarios (i.e., false route hijacks reported). We compared DARSHANA with the individual mechanism: Lat, Hop, Path, Prop. Each scenario of the experiment was repeated 30 times.

*1) Small scale scenarios:* We tested small scale scenarios with nodes from PlanetLab. Each scenario is composed by three nodes. Two of them have a source-destination relation and the third one serves as the hijacker. Throughout the scenarios the source and the destination are fixed and the

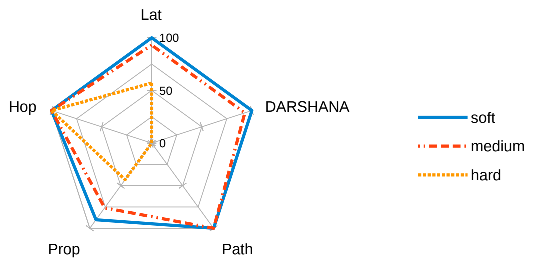


Fig. 4. The percentage of times that each mechanism detects a simulated route hijacking attack. The scenario involves Portugal as the source, Ireland as the destination and France as the hijacker. The labels soft, medium and hard represent different sets of thresholds for each mechanism.

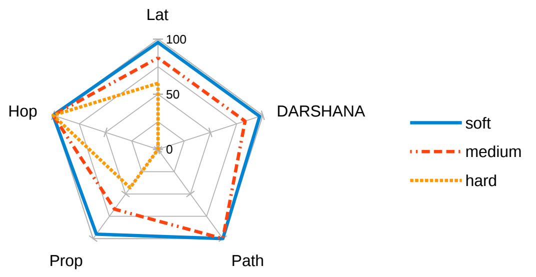


Fig. 5. The percentage of times that each mechanism detects a simulated route hijacking attack. The scenario involves Portugal as the source, Ireland as the destination and Poland as the hijacker.

hijacker varies its distance to the source. We selected a node from Portugal as the source, a node from Ireland as the destination and the hijackers are nodes from France and Poland. The distances from source to hijacker were chosen in a way that would enable us to determine the cases when DARSHANA has more difficulty in detecting the attack. The results are presented in Figures 4 and 5.

The figures show the percentage of times that each mechanism detects a simulated route hijacking attack under different scenarios. Each figure contains three labels: soft, medium and hard. They specify qualitatively the thresholds that were used for each mechanism. There are four thresholds, $T_{Lat}$, $T_{Hop}$, $T_{Path}$ and $T_{Prop}$, that indicate how much measurements of RTT, hop count, path and propagation delay have to deviate in order to declare a route hijacking. The values of the thresholds used in the experiments were defined based on many experiments done before the evaluation here reported. These values are presented in Table II.

Observing the results, we can conclude that soft thresholds lead Lat to detect all hijacks. However, this also leads to false positives and, in the case of DARSHANA, prevents the other mechanisms from actuating and removing such false positives, while keeping a high detection rate. The Hop and the Path mechanism present 0 or 100% values. This is due to the fact that these mechanisms provide constant results through time. Therefore for certain values of $T_{Hop}$ and $T_{Path}$, these mechanisms will detect or not the simulated attack. In regard

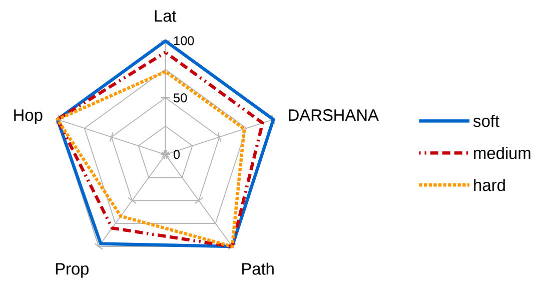


Fig. 6. The percentage of times that each mechanism detects a simulated route hijacking attack. The scenario involves N. Virginia as the source, N. California as the destination and Germany as the hijacker.

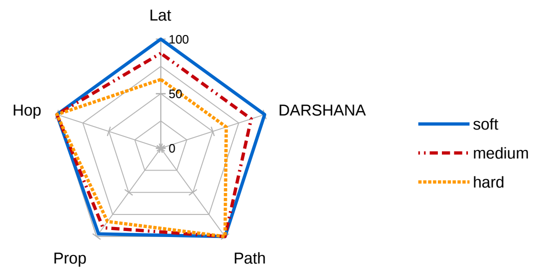


Fig. 7. The percentage of times that each mechanism detects a simulated route hijacking attack. The scenario involves Ireland as the source, Germany as the destination and South Korea as the hijacker.

to the propagation delay mechanism and observing Figures 4, 5 and Table II, the hard label in Figure 4 corresponds to $T_{Prop}$ = 4.6 and the soft label in Figure 5 is equal to $T_{Prop}$ = 7.4. In Figure 5 with the soft label, the detection of the attack is very close to 100%, but by observing Figure 4 we can see that for the hard label, this mechanism can only identify the attack less than 50% of the times. This means that changes in propagation delay are much more observable as the hijacker increases its distance to the source of the traffic.

When the source and the hijacker are close, the packets do not traverse many different autonomous systems and so DARSHANA is not able to detect the hijack with the hard thresholds.

*2) Real scenarios:* While our analysis in the previous section provided some insights about the capacity of detection of our system, we wanted to test our detection mechanism using historical prefix hijacking events and confirm that our mechanism behaves better by having the hijacker farther away. We simulated two scenarios. It was not possible to choose nodes from the exact locations in which these scenarios took place so we chose nearby nodes. The first scenario corresponds to the Belarusian Traffic Diversion [23], where traffic from New York was diverted to Belarusian ISP GlobalOneBel before arriving to the intended destination, Los Angeles. To simulate this, we deployed two nodes (micro-instances) in two different Amazon AWS regions: N. Virginia and N. California. The node from N. Virginia is the source, the node from N. California is the destination. We used a node from PlanetLab in Poland to serve as a hijacker and to represent the Belarusian ISP GlobalOneBel. The second scenario emulates the China 18-Minute Mystery [24], in which, allegedly, traffic between London and Germany took a detour through China. We simulate this by selecting a node from PlanetLab in Ireland as the source, a node from Germany as the destination and a micro-instance of Amazon AWS in Seoul as the hijacker. The results can be found in Figures 6 and 7. In these scenarios there is substantially more change, between the samples after attack and the samples prior to the simulated attack, than in the small scenarios experiments. The values for thresholds are shown in Table II.

To better understand why DARSHANA presents superior detection values in relation to the experiments done in Section IV-B1, we need to have an idea of the paths that packets take from source to destination, before the hijacking and after the hijacking. Table III shows the number of the ASes the traffic traverses, before the attack and after. It is possible to observe that the normal path and the hijacked path from the small scale scenarios share more numbers than the paths from real case scenarios. Furthermore, detecting the attack between two instances of Amazon AWS is easy, because there is not a lot path diversity as we can see from the normal path between N. Virginia and N. California.

*3) False positives:* There is a false positive when a scheme claims to have detected an attack that did not exist. We evaluated the false positives for each individual mechanism of our system during a run of 1h15m. The false positives were calculated by executing each detection mechanism with scenarios without running the attack (i.e., without hijacking). By capturing the amount of alerts given by a mechanism we get the false positive rate $\#alerts/\#samples$, where $\#alerts$ is the number of alerts and $\#samples$ the number of samples taken. We tested for three different scenarios and each scenario contains a source and a destination. For the first scenario, we chose a node from POR as the source and a node from IRE as the destination; in the second, the source is a node from IRE and the destination is a node from GER; finally for the last scenario the source is a node from VA and the destination is a node from CA.

For Path and Hop the number of false positives observed was 0, because there would have to be legitimate route changes to cause them, but these are not frequent and none was observed. For Prop the number of false positives was also 0, as the mechanism always searches for the maximum RTT stored to compute the maximum propagation delay ever observed. Unless a great anomaly in RTT is found, the mechanism will not raise an alarm. For Lat, we received a new sample from 30 to 30 seconds getting a total of 150 samples per scenario. The results are presented in Figure 8. The values for the thresholds were chosen with the objective to reveal variation in the false positive rate. As we can see in all sets of columns the false positive rate is bigger for softer thresholds. This makes sense since small thresholds mean that a small variance of RTT is considered an attack. For the soft label the value used was 1.2,

TABLE II
VALUES OF THRESHOLDS USED FOR EACH SCENARIO. S, D AND H ARE THE SOURCE, DESTINATION AND HIJACKER, RESPECTIVELY.

| | S:POR \| D:IRE \| H:FRA | | | S:POR \| D:IRE \| H:POL | | | S:VA \| D:CA \| H:POL | | | S:IRE \| D:GER \| H:S.Korea | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Mechanisms | Soft | Medium | Hard | Soft | Medium | Hard | Soft | Medium | Hard | Soft | Medium | Hard |
| Lat | 1.2 | 1.3 | 1.4 | 2.1 | 2.2 | 2.3 | 3.6 | 3.65 | 3.7 | 13 | 13.5 | 14 |
| Hop | 1.05 | 1.1 | 1.15 | 1.05 | 1.1 | 1.15 | 2.05 | 2.1 | 2.15 | 2.2 | 2.25 | 2.3 |
| Path | 0.9 | 0.8 | 0.7 | 0.9 | 0.8 | 0.7 | 0.9 | 0.8 | 0.7 | 0.9 | 0.8 | 0.7 |
| Prop | 3.6 | 4.1 | 4.6 | 7.4 | 7.9 | 8.4 | 12.5 | 13 | 13.5 | 70 | 75 | 80 |

TABLE III
NUMBERS OF THE ASES THAT PACKETS TRAVERSE

| Hijacker | From - To | | | | | |
|---|---|---|---|---|---|---|
| | POR - IRE | | IRE - GER | | VA - CA | |
| | Normal | Hijacked | Normal | Hijacked | Normal | Hijacked |
| FRA | 1930,21320,1213 | 1930,21320,2200,15557,1213 | - | - | - | - |
| POL | 1930,21320,1213 | 1930,21320,8501,8890,1213 | - | - | 16509 | 16509,2603,8501,8890,16509 |
| S.Korea | - | - | 1213,21320,680 | 1213,3356,2516, 16509,4766,174,680 | - | - |

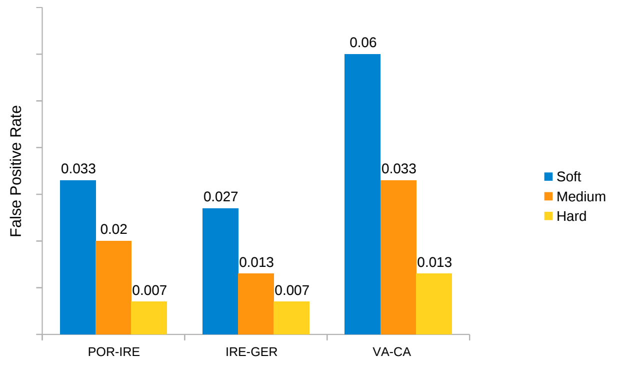


Fig. 8. False positive rate of RTT in different scenarios. The $Y$ axis refers to the false positive rate and the $X$ axis represents the different scenarios tested.

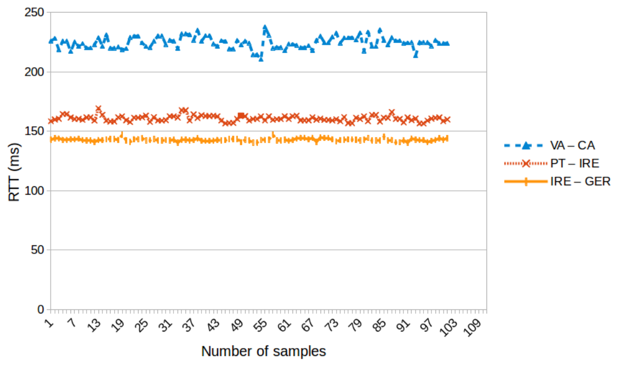


Fig. 9. RTT values in different scenarios. The $Y$ axis refers to the RTT values in milliseconds and the $X$ axis represents the number of samples.

for medium the value was 1.3 and for the hard label the value chosen was 1.4.

The results for DARSHANA were obtained with the soft thresholds for Lat. However, on the contrary of Lat, the false positive rate for DARSHANA was 0 in all scenarios, as the other mechanisms (Path, Hop, and Prop) filtered the false positives of Lat, leading to 0 false positives as obtained with each of the 3 individually.

## C. Cost

DARSHANA keeps probing for RTT with period $k$. Unless anomalies in RTT are verified, leading the system to use techniques with bigger overhead, like Hop and Path. We evaluate the cost as how many times DARSHANA is forced to execute heavier techniques in normal conditions. The scenarios used were the same as in Section IV-B3. Setting a probing rate for RTT to 60 seconds, Figure 9 illustrates the values for round trip time in different scenarios.

During this period the mean deviations of the samples were low. The scenario with VA and CA, has the biggest mean deviation of approximately 5.02. This implies that the RTT usually remains constant, being difficult to observe anomalies and pass to heavier methods. Considering a value of $T_{Lat}$ = 1.5, the total ping and traceroute messages for this period follow the following formulas, where $\#Msg\_Ping$ and $\#Msg\_Traceroute$ correspond to number of ping and traceroute messages, respectively $\#Msg\_Ping = T \times k$ and $\#Msg\_Traceroute = \#Msg\_Ping/n$ .

Where $T$ is the total time of the experiment, $k$ is the ping period and $n$ is the traceroute period. For this experiment $\#Msg\_Ping = 100 \times 1 = 100$ ping messages and $\#Msg\_Traceroute = 100/5 = 20$ traceroute messages. All of this demonstrates that even for low values of $T_{Lat}$, the total number of $\#Msg\_Ping$ and $\#Msg\_Traceroute$ end up only being dependent on $k$ and $n$.

## V. RELATED WORK

Many solutions have been proposed for the IP prefix hijacking problem. Some of them are crypto-based such as [3]–[6]. These solutions require deep changes in routers and network protocols. BGP routers need to sign and verify announcements which leads to a non negligible overhead.

Other solutions like [7]–[10] are more deployable because they do not require changes in routers, they only need access to public registries, like Route Views and European IP Networks (RIPE) to conduct passive monitoring and look out for

Multiple Origin Autonomous Systems (MOAS) [25]. An IP prefix should only be generated by a single AS, so this conflict may indicate a prefix hijacking. The problem associated to these solutions is that many times the public registries may be outdated and inaccurate, leading to an increase of the number of false positives.

Finally, there are solutions that rely only on the data plane like ours. They are not constrained by the availability of BGP information and are more accurate. [12] uses a set of monitors to detect prefix hijacking in real time. These vantage points monitor a prefix from topologically diverse areas. Each monitor keeps track of the hop count and the path to a target prefix and if past measurements disagree with new ones then a route hijacking is declared, the need for vantage points end up limiting the system. On the other hand, [11] detects IP prefix hijacking by observing unreachability events. It is owner-centric, in a point that the mechanism keeps sending probes to transit ASes. If enough ASes stop responding, the system declares the attack. If the attacker forwards the responses of the probes back to the sender, the attack is not detected.

In this paper we make use of the propagation delay as another criteria to detect route hijacking. This delay has been used in [26], which proposed a system that presents undeniable proof about traffic traversing a certain forbidden zone defined by the sender. To know if a certain relay node is not in the forbidden region, the minimum possible RTT from the source to any node in the forbidden zone was calculated, with the propagation delay. In case the RTT from the source to the relay node is less than the RTT calculated earlier, then the relay node is not in the forbidden region.

The design of the lightweight and end-host-based probing techniques was inspired by Hubble [27], where low overhead probing techniques are used first and heavier, but more reliable techniques, are only used when there is such a need.

## VI. Conclusion

This paper presented DARSHANA, a route hijacking detection system. By only applying active probing methods, we ensure accuracy and deployability. Different techniques turn the system redundant enough to not be avoided by attackers. The design of the detection system minimizes the overhead, by using techniques with low overhead more often. Techniques with greater reliability and overhead are only executed when necessary. Our system is the first to use the propagation delay in this context, providing one more metric for the purpose of detection. We evaluated the system with small scale and real scenarios.

*Acknowledgements* This work was supported by the European Commission through project H2020-653884 (SafeCloud) and by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UID/CEC/50021/2013 (INESC-ID).

## References

[1] S. H. Y. Rekhter, T. Li, "A border gateway protocol 4 (RFC 4271)," January 2006.

[2] D. Goodin, "Hacking Team orchestrated brazen BGP hack to hijack IPs it did not own," 2015. [Online]. Available: http://arstechnica.com/security/2015/07/hacking-team-orchestrated-brazen-bgp-hack-to-hijack-ips-it-didnt-own/

[3] M. Lepinski, "BGPSEC protocol specification, draft-ietf-sidr-bgpsec-protocol-17," 2016.

[4] S.Kent, C.Lynn, and K.Seo, "Secure border gateway protocol (S-BGP)," *IEEE JSAC Special Issue on Network Security*, 2000.

[5] W.Aiello, J.Ioannidis, and P.McDaniel, "Origin authentication in inter-domain routing," *Proceedings of ACM Conference on Computer and Communications Security*, 2003.

[6] K.Butler, P.McDaniel, and W.Aiello, "Optimizing BGP security by exploiting path stability," *Proceedings ACM Conference on Computer and Communications Security*, 2006.

[7] M. Lad, D. Massey, D. Pei, and Y. Wu, "PHAS: A prefix hijack alert system," *Proceedings of the Usenix Security Conference*, pp. 153–166, 2006.

[8] J. Karlin, S. Forrest, and J. Rexford, "Pretty good BGP: improving BGP by cautiously adopting routes," *Proceedings of the 2006 IEEE International Conference on Network Protocols*, pp. 290–299, 2006.

[9] X. Hu, Mao, and Z. Morley, "Accurate Real-time Identification of IP Prefix Hijacking," *IEEE Symposium on Security and Privacy*, no. 2, pp. 3–17, 2007.

[10] X. Shi, Y. Xiang, Z. Wang, X. Yin, and J. Wu, "Detecting prefix hijackings in the Internet with ARGUS," *Proceedings of the 2012 ACM Internet Measurement Conference*, 2012.

[11] Z. Zhang, Y. Zhang, Y. C. Hu, Z. M. Mao, and R. Bush, "iSPY: detecting IP prefix hijacking on my own," *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication*, 2008.

[12] C. Zheng, L. Ji, D. Pei, J. Wang, and P. Francis, "A light-weight distributed scheme for detecting IP prefix hijacks in real-time," *Proceedings of the 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pp. 277–288, 2007.

[13] K. Butler, T. 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.

[14] M. Brown. (2008) Pakistan hijacks youtube. [Online]. Available: http://research.dyn.com/2008/02/pakistan-hijacks-youtube-1/

[15] H. Ballani, P. Francis, and X. Zhang, "A study of prefix hijacking and interception in the Internet," *ACM SIGCOMM Computer Communication*, 2007.

[16] "The handbook for radio communications, 89th edition," 2012.

[17] M. Luckie, Y. Hyun, and B. Huffaker, "Traceroute probe method and forward IP path inference," *Proceedings of the 8th ACM SIGCOMM conference on Internet measurement conference*, p. 311, 2008.

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

[19] Team cymru. [Online]. Available: http://www.team-cymru.org/IP-ASN-mapping.html

[20] Maxmind. [Online]. Available: http://dev.maxmind.com/

[21] Planetlab : global research networks that supports the development of new network services. [Online]. Available: https://www.planet-lab.eu/

[22] Amazon web services. [Online]. Available: https://aws.amazon.com/

[23] J. Cowie. (2013) The new threat: Targeted Internet traffic misdirection. [Online]. Available: http://research.dyn.com/2013/11/mitm-internet-hijacking/

[24] ——. (2010) China 18-minute mystery. [Online]. Available: http://research.dyn.com/2010/11/chinas-18-minute-mystery/

[25] X. Zhao, D. Pei, L. Wang, D. Massey, A. Mankin, S. F. Wu, and L. Zhang, "An analysis of BGP multiple origin AS (MOAS) conflicts," *Proceedings of the First ACM SIGCOMM Workshop on Internet Measurement Workshop*, pp. 31–35, 2001.

[26] D. Levin, Y. Lee, L. Valenta, Z. Li, V. Lai, C. Lumezanu, N. Spring, and B. Bhattacharjee, "Alibi routing," *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, pp. 611–624, 2015.

[27] J. John, E. Katz-Bessett, H. Madhyastha, A. Krishnamurthy, D.Wetherall, and T. Anderson, "Studying blackholes in the Internet with hubble," *Proceedings of NSDI*, 2008.