On the Evasion of Delay-Based IP Geolocation

AbdelRahman M. Abdou
Dept. of Systems and Computer Eng.
Carleton University, Ottawa
Email: abdou@sce.carleton.ca

Ashraf Matrawy
School of Information Technology
Carleton University, Ottawa
Email: amatrawy@sce.carleton.ca

Paul C. van Oorschot
School of Computer Science
Carleton University, Ottawa
Email: paulv@scs.carleton.ca

***Abstract*—We explain a newly found vulnerability that allows circumvention of commonly used delay-based geolocation techniques that use ping or traceroute to sample delays. Attacks may leverage the echo request/reply type of the ICMP protocol. ICMP's echo request/reply protocol does not specify a mechanism to measure the delays between network nodes. Consequently, different implementations exist on different platforms to achieve this functionality. Other work in literature presented an adversary that can only increase the round trip times by delaying the echo reply messages. However, as we explain, current implementations of ping and traceroute also allow an adversary to decrease the round trip time, enabling it to evade delay-based geolocation techniques with high accuracy. We evaluate the effect of this attack on two delay-based techniques, and analyze an adversary's evasion capabilities, given its ability to also decrease the observed delays between itself and the set of landmarks conducting the geolocation process.**

I. Introduction

The recent proliferation of Location-Based Services (LBSs) on the Internet has motivated the need for secure, reliable and relatively accurate Internet geolocation tools. A number of geolocation tools have been proposed in both the academic and industrial domain. The research community has focused on *client-independent* geolocation [1], where the client's location is computed by a party other than the client itself. Delay-based IP geolocation is one common example in this class of techniques. In delay-based geolocation, the round-trip time (RTT) delay is measured between the client and a set of landmarks with known geographic locations, and the client's location is estimated in relation to these landmarks. Delay-based techniques assume the client responds to the delay-measurement probes, most commonly being Internet Control Message Protocol (ICMP) [2] `pings` and/or `traceroutes`. When compared to *client-dependent* geolocation (where the client computes its location and communicates it to the LBS, such as in the Global Positioning Sytem, or GPS [3]), client-independent geolocation is a better fit for security-sensitive applications [4], [1], i.e., those where the client's location impacts security. Examples include fraud detection and location-based access control.

We demonstrate a new vulnerability in delay-based geolocation techniques which is due to a lack of data integrity in the implementation of the `ping` and `traceroute` programs on most platforms. For example, most common `ping` implementations record the packet-creation time in the DATA field of the ICMP packet. The RTT is then calculated by subtracting this timestamp from the time the echoed packet was received, expecting the client to return the data unchanged. However, an adversary (playing the role of a legitimate client) can alter the timestamp in the DATA field before echoing the packet. This enables the adversary to accurately increase or decrease the observed RTT, and directly influence the computed location.

(a) Attempted distance = 6,656.6 km. Distance error = 37.55 km.

(b) Attempted distance = 15,091.7 km. Distance error = 70.7 km.

(c) Attempted distance = 11,685.2 km. Distance error = 133 km.

Fig. 1. Examples showing adversarial capabilities given the newly discovered vulnerability (figures are not to scale). ● = true location of adversary; × = intended location of adversary; ○ = location calculated by CBG. The *attempted distance* is that between the true location and the intended one; whereas the *distance error* (i.e. for the adversary) is that between the intended location and the calculated one.

Gill *et al.* [5] studied the ability of an adversary to evade measurement-based geolocation techniques, and the capability of the measuring party to detect if a client is trying to evade geolocation. For delay-based techniques, their analysis considered the vulnerability whereby an adversary can (only) increase the observed RTTs by delaying the echo-reply packets before echoing them. They found a tradeoff between the adversary's *attempted distance* (defined as the distance between the actual and the intended location) and the attack detectability. We explain how an adversary can fully manipulate (increase and decrease) the RTT, and explore the consequences of such manipulation on the geolocation process. To evaluate an adversary's accuracy in forging its location, we implemented two delay-based IP geolocation techniques, CBG [4] and GeoPing [6], and studied the effect of the newly discovered vulnerability on them. The vulnerability enabled some adversaries in our experiments to place themselves at different locations with *distance error* (defined as the distance between the adversary's

intended location and the one calculated by the geolocation technique) below 100 km. Even more worrisome, some of the adversaries that attempted to move more than 5,000 km away from their true locations had distance errors below 200 km. Figure 1 shows three examples while testing on CBG. Our work highlights the need for integrity of timing information, if the timing information is relied upon by security-sensitive applications. Our contributions include:

1) Demonstrating a previously unknown vulnerability in delay-based geolocation techniques that use the `ping` or `traceroute` programs for delay measurement. A proof-of-concept implementation is available at `checkmylatency.ccsl.carleton.ca`.
2) Evaluating its extent to two delay-based geolocation techniques.

In Section II, we provide background on how delay-based geolocation techniques work in general. In addition, we detail the implementations of the `ping` and `traceroute` programs on common platforms. Section III explains how the RTT measured by such implementations can be increased and, more importantly, decreased. We also define the adversarial model. In Section IV, we study the effect of manipulating the RTTs on delay-based geolocation, and analyze how accurately an adversary can forge its (client) location. Section V suggests countermeasures for the new presented exposure. We discuss related work in Section VI and conclude in VII.

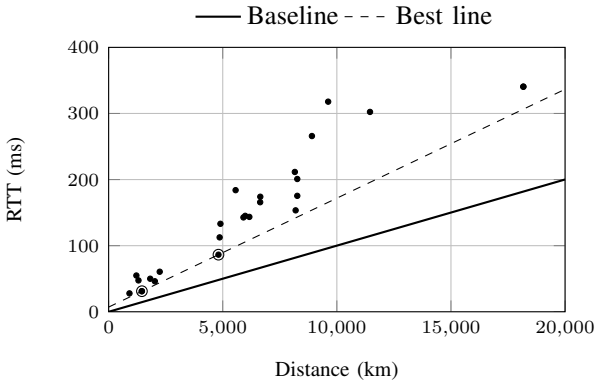


Fig. 2. An example of CBG's calibration at one of the landmarks from our experimental dataset (Section IV). Each marker represents a measured RTT between the landmark and another. The circled markers are the ones determining the best line function.

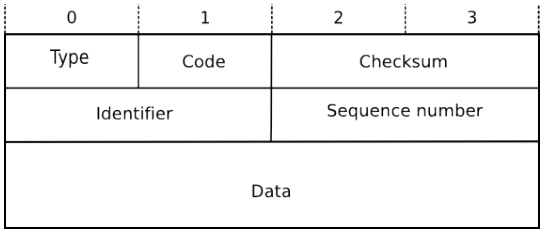


Fig. 3. ICMP header (8 bytes) and data.

II. Background

A. Delay-based IP Geolocation

Delay-based IP geolocation is a class of techniques where the geographic location of an IP address is determined based on the observed delay between the machine assigned that IP address and a set of geographically scattered landmarks with know locations. Despite a plethora of factors that affect the delays between two nodes over the Internet [7], [8], numerous studies over the past ∼13 years have established that there is a strong correlation between delays and geographic distances [6], [9], [4], [10]. The main characteristic relied on is the propagation delay. Most, if not all, delay-based geolocation techniques mitigate the effect of other undesired delay factors (e.g. queuing due to congestion) by using the minimum of multiple delay samples (e.g. 10-20 RTTs) to the target from each landmark. Some techniques perform an additional stage of outlier removal to better filter out noisy samples [11]. Once the delays are measured, the research question addressed by most techniques becomes finding the best way to map these delays to geographic distances. An exception is one of the first delay-based techniques—GeoPing [6]. Instead of mapping delays to distances, GeoPing matches the location of the target client to a location where the most similar delay behavior would be observed. Assuming n landmarks and m reference nodes with known locations, the landmarks in GeoPing first create a delay vector to each of the m nodes. A delay vector of a node contains n values corresponding to the RTT between the landmarks and the node. Each landmark then measures the RTT between itself and the target client. A delay vector is then created for that client, and the client's location is matched to the node with the *nearest* delay vector. The authors of GeoPing proposed to calculate delay nearness as the n-dimensional Euclidean distance between the two delay vectors [6]. However, Manhattan, Canberra, and Chebyshev distances were also considered as replacements to the Euclidean distance [12]. This class of delay-based geolocation returns a location from a discrete space depending on the number of reference nodes available to the locating party.

Most delay-based techniques formulate a delay-distance mapping function. Straight-forward multi-lateration is then applied to determine clients' locations. The difference between techniques lies in the delay-distance mapping step, wherein a set of delay-distance pairs is collected between the landmarks, and is used to calibrate a mapping function. CBG [4] achieves what its authors refer to as the *best line* function. On a distance-delay chart, the best line is the one passing by the two points having the minimum delay-to-distance ratio. Each landmark calibrates its own function based on the delays between itself and all other landmarks. An example of the best line (from our experiments) is shown in Figure 2. The *baseline* function, which represents the speed of light in fiber or (2/3)**c** (where **c** is the speed of light in vacuum) [13], is also plotted on the same chart for reference. After calibration, each landmark measures the RTT to the client, plugs it in its calibrated function and obtains an estimate of the distance between itself and the client. The client's location is then estimated as the centroid of the intersection of circles whose centers are the landmarks and radii are the distances.

B. ICMP Implementations

An ICMP packet [2] gets wrapped by an IP packet for delivery. However, since every layer-3 device (e.g. router) is

required to implement ICMP, it is deemed a layer-3 protocol [2]. Eleven ICMP types are specified by RFC 792. The type is indicated by the TYPE field of the header. The ICMP types relevant to our discussion are the echo request and reply protocols—types 8 and 0 respectively. They share the same header, which is shown in Figure 3. To craft an echo request, the sender sets the TYPE and CODE fields to 8 and 0 respectively, chooses two 16-bit values for the IDENTIFIER and SEQUENCE NUMBER fields, and finally calculates the checksum and places it in its field. Values in the DATA, IDENTIFIER and SEQUENCE NUMBER fields are up to the implementer, however the latter two may aid in matching requests with replies as specified by the RFC. When the receiver gets an echo request message, it should change the TYPE field to 0, recalculate the checksum and echo the packet. According to RFC 792, "the data received in the echo message must be returned in the echo reply message" [2]. However, the RFC does not specify a mechanism to ensure the receiver behaves as described nor, more interestingly, a mechanism to calculate the RTT (round trip time). That is, providing integrity checking is not stated as a requirement.

To the best of our knowledge, all implementations of the `ping` and `traceroute` programs leverage ICMP's echo request/reply to calculate the RTT. Different implementations are in use on different platforms. On some platforms, `ping` and `traceroute` are stateful. They locally record a timestamp s_i of the sender's system time when packet i was created.[1] The RTT is then calculated as the difference between the recorded time and the time the reply was received. Formally, assuming that the sender issues echo requests every t ms to a receiver, and the first one was created at time T, then for all packets $i \geq 0$:

$$s_i = T + i \cdot t \tag{1}$$

If the packets take γ_1 ms one-way delay from the sender to the receiver, then echo requests will arrive to the receiver at times

$$\begin{aligned} m_i &= s_i + \gamma_1 \\ &= T + i \cdot t + \gamma_1 \end{aligned} \tag{2}$$

A benign receiver issues echo replies once it receives the requests, i.e. at times m_i for all packet indices i. If the packets take γ_2 ms one-way delay from the receiver back to the sender, then those echo replies will arrive to the sender at times

$$\begin{aligned} r_i &= m_i + \gamma_2 \\ &= s_i + \gamma_1 + \gamma_2 \\ &= T + i \cdot t + \gamma_1 + \gamma_2 \end{aligned} \tag{3}$$

for which the RTT for each packet i is calculated as

$$\texttt{RTT}_i = r_i - s_i = \gamma_1 + \gamma_2 \tag{4}$$

On other platforms (specifically, Linux, BSD and Mac OS), `ping` is *stateless*—the sender stores no timing information locally about an issued echo request. To measure the RTT, the sender places a 16-byte timestamp, s, in the DATA field of a `ping` packet (recall, s is the sender's system time when the packet was created). Upon receiving the reply, the sender records the receiving time, r, and calculates the RTT as the difference between the receiving time and the timestamp in the DATA field of the received packet, again using Eqn. 4.[2]

[1] We observed this `ping` implementation in Windows XP and Windows 8, and believe this is the way it has been implemented on other versions.

TABLE I. CONTENTS OF AN ICMP ECHO REQUEST PACKET AS WE OBSERVE ON DIFFERENT IMPLEMENTATIONS OF `PING`.

	Type	ID	SN	Data
Linux	8	Unique per session	1,2,3,...	`timestamp` and fixed pattern
BSD	8	Unique per session	0,1,2,...	`timestamp` and fixed pattern
Mac OS	8	Unique per session	0,1,2,...	`timestamp` and fixed pattern
Windows	8	Constant per Windows version	A variable global to all sessions	Fixed pattern

On Linux, BSD and Mac OS implementations, we noticed that a unique identifier is chosen for each `ping` process (for the IDENTIFIER field) such that if the sender was `ping`ing multiple nodes at once, replies get matched to their corresponding issuing processes; the sequence number starts at 1 (mostly on Linux) or 0 (mostly on BSD and Mac OS), and is incremented by 1 on each subsequent `ping` message. On Windows (8 specifically), we noticed that the identifier and sequence numbers are global variables across multiple processes; the identifier is constant and the sequence number is incremented by 1 on each subsequent `ping` message. Table I summarizes our observations to how different implementations of `ping` fill the contents of an ICMP echo request header.

III. MANIPULATING LATENCIES AND ADVERSARIAL MODEL

A. Manipulating Latencies

Knowing how `ping` and `traceroute` are implemented (Section II-B), we now explain how a receiver can manipulate the RTT observed by the sender. To increase the RTT by δ ms, the receiver can simply hold on to the received echo requests for δ ms before issuing echo replies [5]. Decreasing is done in one of two ways, depending on whether the timestamp s (the time when the echo request packet was created) is recorded locally (stateful implementation) or in the DATA field of the packet (stateless).

For the case where the sender records s locally, the receiver decreases the RTT as follows. First, it estimates the *waiting* time, between successively received echo requests (recall that delay-based geolocation techniques take more than one delay sample to the client).[3] This enables the receiver to estimate when will the next echo request be received. In Eqn. 2, subtracting m_i from m_{i+1} for all i gives the same value t. In practice, the accuracy of the estimated t depends on the stability of the one-way delay from the sender to the receiver, γ_1. However, the receiver can average the calculated waiting time over multiple echo requests. To reduce the RTT that the sender will observe by δ ms, the receiver issues a fake (early) echo reply δ ms before the time it expects to receive the next

[2] We noticed that `ping` refrains from calculating the RTT if the packet size was set to be less than 16 bytes (8 bytes on some Linux and BSD versions). The packet size can be set by the `-s` option on Linux, BSD and Mac OS platforms.

[3] We noticed that on many platforms, the waiting time is set to 1 second by default, but users are usually given the option to set a different constant. The waiting time can be set by the `-i` option on the Linux, BSD and Mac OS platforms and the `-I` option for Solaris platforms.

echo request. That is, assuming the sender issues echo requests at times s_i as described by Eqn. 1, the receiver issues fake echo replies at times m'_i instead of m_i (Eqn. 2), such that:

$$m'_i = s_i + \gamma_1 - \delta \tag{5}$$

which requires the receiver to craft an echo reply packet before receiving the echo request. Following Eqn. 3, the sender then receives the echo replies at times:

$$\begin{aligned} r'_i &= m'_i + \gamma_2 \\ &= s_i + \gamma_1 - \delta + \gamma_2 \end{aligned} \tag{6}$$

and hence, calculates the manipulated RTT as:

$$\text{manipulated RTT}_i = r'_i - s_i = \gamma_1 + \gamma_2 - \delta \tag{7}$$

which is δ less than the actual RTT, $\texttt{RTT}_i$ (Eqn. 4). Note that the receiver (adversary) can trivially fake echo replies because, as discussed in Section II-B, the DATA and headers of an ICMP echo request packets are predictable. The SEQUENCE NUMBER of the next packet is mostly one plus that of the previous one; the packet IDENTIFIER is fixed for all packets created by the same `ping` session. The receiver can also set a specific value for the RTT, if it knows the actual RTT between itself and the sender, by setting $\delta = \texttt{RTT}_i - \tau$, where τ is the adversary's desired RTT value. In such case, the sender calculates the manipulated RTT as (from Eqn. 7):

$$\begin{aligned} \text{manipulated RTT}_i &= \gamma_1 + \gamma_2 - \delta \\ &= \gamma_1 + \gamma_2 - (\texttt{RTT}_i - \tau) \\ &= \texttt{RTT}_i - \texttt{RTT}_i + \tau \quad \text{from Eqn. 4} \\ &= \tau \end{aligned} \tag{8}$$

Finally, recall from Section II-A that delay-based techniques commonly implement an outlier-removal procedure, or use the smallest sampled RTT of a set of several samples. This gives the receiver greater ability to trick the server into using whatever RTT the receiver desires. For example, in the above procedure, the receiver needs to receive at least two successive echo requests to calculate the waiting time between them. To avoid having the sender using the RTT calculated from the first two echo requests, the receiver can delay their replies for a large period of time (e.g. 1 second) to make sure the sender will discard them as outliers. Doing so gives the receiver more time to estimate the waiting time.

For the case where the sender records the timestamp of the packet-creation time (s) in the DATA field of echo requests (i.e. the stateless implementation), the receiver only need edit this field before issuing an echo reply. The receiver does not even need to calculate the waiting time between packets in this case, nor does it have to issue fake echo replies before receiving their requests. To decrease the RTT calculated from echo request packet i by δ ms, the receiver changes s_i in the DATA field to $s_i + \delta$, in which the sender calculates the manipulated RTT as in Eqn. 7. This procedure can also be used to increase the RTT by δ ms if s_i is set to $s_i - \delta$. Similar to the above argument, setting $\delta = \texttt{RTT}_i - \tau$ makes the sender think the RTT is τ (Eqn. 8).

To demonstrate a proof of concept of the attack explained in this section, we set up a machine (`checkmylatency.ccsl.carleton.ca`) which, when `ping`ed from a Linux, BSD or Mac machine, behaves as follows: for the first 5 packets, the actual RTT is returned; for the next 25 packets, the actual RTT keeps decreasing by 2 ms. The same procedure repeats starting the 30^{th} packet, 60^{th} packet, etc.

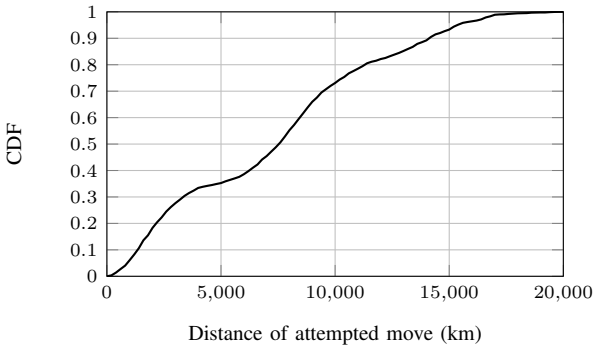


Fig. 4. CDF of the distances the adversaries attempted to move during the evasion attempts we conducted in our evaluation. A point (x,y) on the curve means a fraction y of all 2,550 evasion attempts attempted to move x km or less away from the true location. Note: this graph shows experimental design, not results.

B. Adversarial Model

The adversary is a client that attempts to manipulate a geolocation process trying to locate it. The adversary has full control over its own machine, but no other machines. We assume the LBS (Location-Based Service) uses a delay-based geolocation technique that leverages `ping` or `traceroute` for delay calculation (essentially, most of those in literature). However, the adversary cannot influence the delay-distance calibration process of the landmarks (see Section II-A), nor infer the calibration function.

The adversary is able to selectively manipulate the delays between itself and any landmark, as explained in Section III-A. Regardless of whether `ping` or `traceroute` is used, we assume the adversary is able to accurately increase or decrease the RTT observed by a measuring party.

We assume the only information the adversary absolutely knows is the geographic locations of the landmarks. As such, the adversary does not know which geolocation technique the LBS uses. The adversary also does not know the distance-delay calibration function of each landmark; it uses a linear function that we explain in Section IV below. We also assume the adversary neither knows the RTT between each landmark and its *intended* location (where the adversary wants to appear to be, in terms of the result computed by the geolocation technique), nor between each landmark and its true location. We make the latter assumption because the landmarks may already be taking countermeasures to prevent anyone from `ping`ing their addresses except themselves.

IV. Attack Evaluation

In this paper, we focus on delay-based geolocation, and select two techniques—GeoPing [6] and CBG [4]—as representatives for evaluation. However, we believe the effect of our attack on these techniques can be extended to other delay-based techniques, giving analogous effect. To evaluate the attack, we used the PlanetLab testbed [14], where we

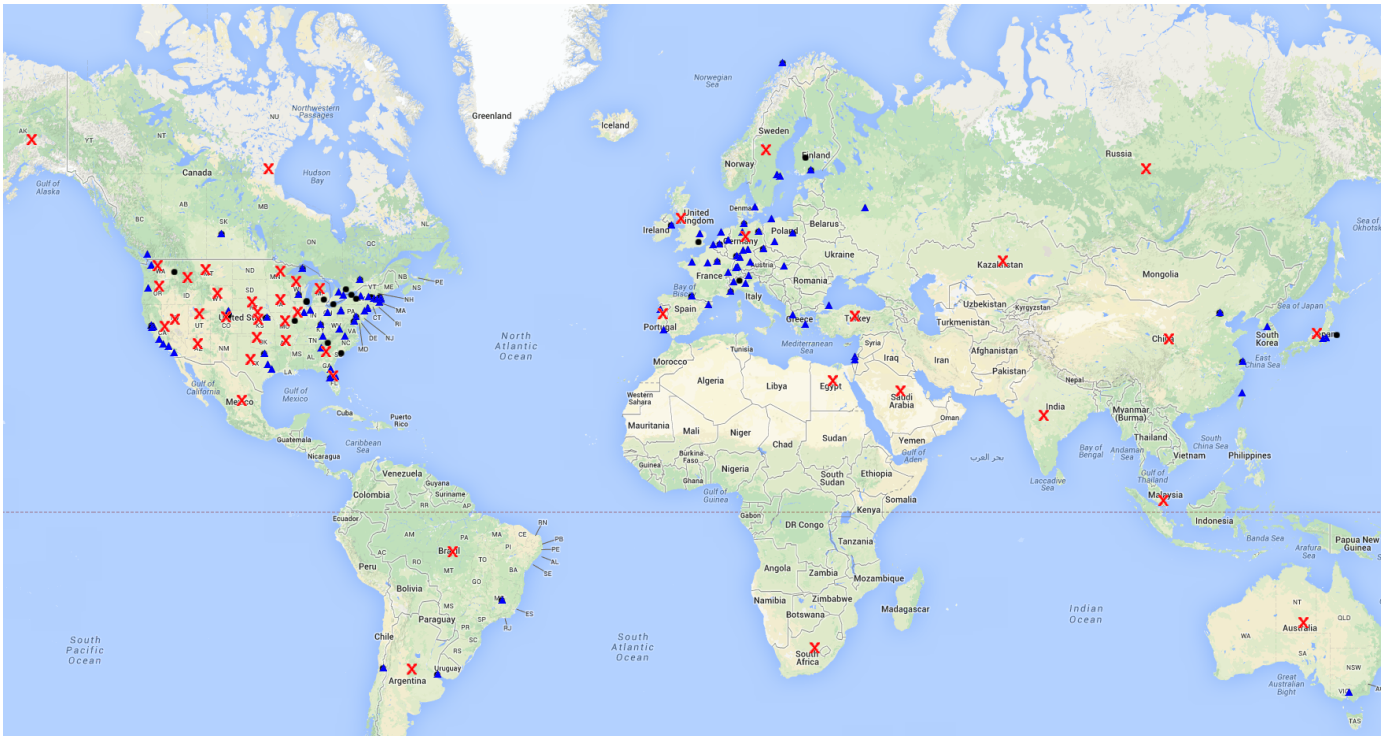


Fig. 5. ▲ = Landmarks; ● = true locations of adversaries; × = intended locations of adversaries. For each of 51 modeled adversary locations, an attempt is made to forge its location to each of the 50 intended locations, for a total of 2,550 evasion attempts. A total of 122 landmarks are used.

selected 144 PlanetLab nodes to represent 122 landmarks and 51 target clients (some nodes acted as both). Their locations are shown in Figure 5. We obtained the delays between these nodes from the iPlane project [15]. Those delays were collected on March 27, 2014. Each of the target clients shown in Figure 5 attempted to forge its location to 50 other locations, marked by × on the same Figure, giving a total of 2,550 evasion attempts. Figure 4 shows a CDF of the distances between the actual and intended locations; 50% of the 2,550 adversaries attempted to move at least ∼7,600 km away from their true locations. In practice, such large distances are not typically state or city level relocation, but rather country or continent level one. In fact, 20 of the chosen intended locations are the centroids of 20 countries, and the remaining 30 are the centroids of 30 US states. The evaluation metrics we use are the adversary's distance error and direction error. The first is the distance between the adversary's intended location and the one calculated by the geolocation technique. We used the great circle distance[4] to calculate this metric [16]. The second metric is the absolute spherical angle (i.e. ≤180) between the lines passing by both locations and the adversary' true location. We used spherical trigonometry to calculate this metric, where we used 6,371 km as an approximation to the Earth's radius [17]. Figure 6 shows the two metrics.

Recall from Section III-B that we assume the adversary is able to increase and decrease the delays, regardless of whether the stateful or stateless implementations of `ping` (or `traceroute`) are used. We assume the adversary's true location is $a = \{x, y\}$. L is the set of all 122 landmarks. The actual RTT between the adversary's true location and each

[4]A great circle is one whose center and radius are that of the Earth.

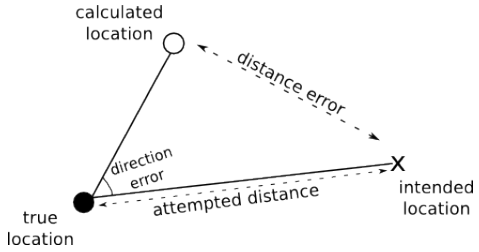


Fig. 6. Distance and direction errors. The calculated location is the one returned by the geolocation technique, whereas the intended location is the one where the adversary intended to appear at (fraudulently).

landmark $l \in L$ is denoted $\alpha(a, l)$. To evade geolocation, the adversary manipulates the RTT observed by each landmark $l \in L$. To forge its location to $a' = \{x', y'\}$, the adversary must mislead each landmark $l \in L$ to observe an RTT of $\alpha(a', l)$ instead of $\alpha(a, l)$. That is, from Eqn. 8, the adversary sets $\delta_l = \alpha(a, l) - \alpha(a', l)$.

However, this requires the adversary to know both $\alpha(a, l)$ and $\alpha(a', l)$, which we assume it does not (Section III-B). To obtain an approximation to those RTTs, the adversary may use the speed of light in fiber (i.e. (2/3)**c**) as an estimate for the one-way traffic propagation speed over the Internet [13]. However, Katz-Bassett *et al.* [18] found that a speed between (2/9)**c** and (4/9)**c** better describes the delay nature of the typically multi-hop Internet routes. We study the adversary's evasion capabilities when it uses (3/9)**c** = (1/3)**c** as an approximation to the traffic propagation speed. That is, using $\beta(a, l)$ and $\beta(a', l)$ as the adversary's estimated RTT between the true/intended

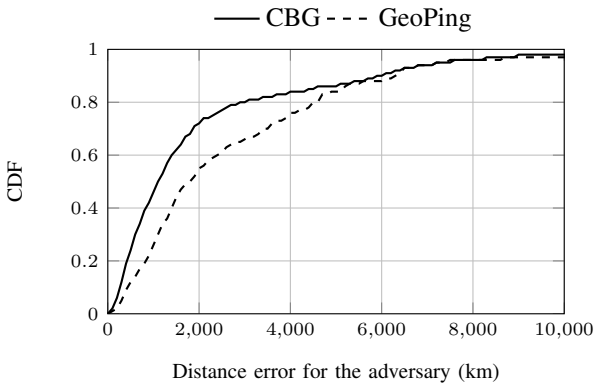


Fig. 7. CDF of the error in distance for the adversary upon attacking the selected delay-based techniques. A point (x,y) on the curve means a fraction y of all 2,550 evasion attempts resulted in distance error of x km or less.

location and landmark l, then

$$\beta(a,l) = \frac{2 \cdot dis(a,l)}{(1/3)\mathbf{c}} \tag{9}$$

and

$$\beta(a',l) = \frac{2 \cdot dis(a',l)}{(1/3)\mathbf{c}} \tag{10}$$

where $dis(a,l)$ and $dis(a',l)$ are the great circle geographic distances [16] between the adversary's true/intended location and landmark l. Note, these delays are round-trip and not one-way, hence the extra 2 in the numerator.

A. Evaluation Results

Figure 7 shows a CDF of the distance error. For CBG, one-third of all 2,550 adversaries resulted in below 700 km (close to the width of France) distance error, and two-third resulted in below 1,700 km. Both values are less than half the width of the US. For example, if Netflix is using CBG to enforce US geographic restriction policies, at least two-third of its clients are expected to bypass these restrictions. When evading GeoPing, the adversary had larger distance errors; one-fifth of all adversaries had errors below 850 km, and half of the adversaries had errors below 1,800 km. The difference between CBG and GeoPing, however, partly stems from CBG being generally more accurate than GeoPing [4]. Unfortunately, such higher accuracy helps the adversary: it more accurately returns the adversary's intended location.

The CDF of the direction error for both techniques is shown in Figure 9. For CBG, 88% of evasion attempts resulted in direction errors below 50 degrees; 82% for GeoPing. To interpret this result, one can think of a US bank restricting credit card transactions to the US. Using CBG, and assuming relatively small distance error, about 88% of European-based adversaries are expected to succeed to pretend to be in the (contiguous) US. That is because for most adversaries whose true locations are in Europe (excluding Iceland) and who intend to pretend to be in the US, a direction error below ∼50 degrees (and distance error below ∼4,500 km) enables them achieve their objective. Figure 8 shows the spherical angle at the intersection point, close to the extreme west of Europe, of two lines enclosing the contiguous US.

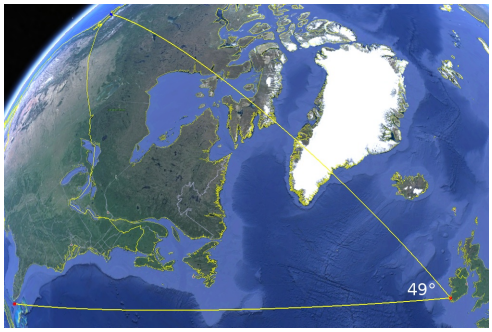


Fig. 8. The spherical angle at the intersection point, close to the extreme west of Europe, of two lines enclosing the contiguous US is ∼49 degrees.

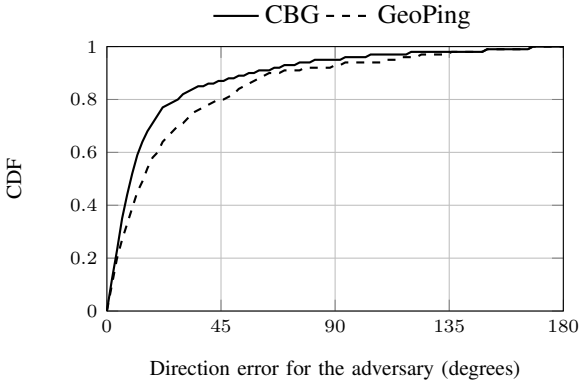


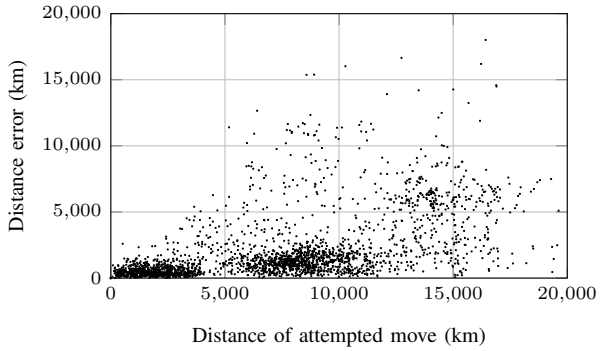
Fig. 9. CDF of the error in direction for the adversary upon attacking the selected delay-based techniques. A point (x,y) on the curve means a fraction y of all 2,550 evasion attempts resulted in direction error of x degrees or less.

Next, we explore the relationship between the distance the adversary attempts to move itself and the resulting distance error of the adversary. Figure 10 shows a scatter plot of both factors. The x-axis represents the distance between the adversary's actual and intended location, the y-axis shows the distance error. The coefficients of correlation between both factors is ∼0.5 for CBG and GeoPing. Recall from Equations 9 and 10 that the adversary subtracts $\beta(a,l)$ (the approximated RTT) from $\alpha(a,l)$ (the actual RTT), and then adds $\beta(a',l)$. As the distance between a (the true) and a' (the intended) locations become smaller, $|\beta(a,l) - \beta(a',l)|$ approaches 0. Thus, the RTT observed by each landmark l tends to remain close to the actual $\alpha(a,l)$. This explains the positive correlation.

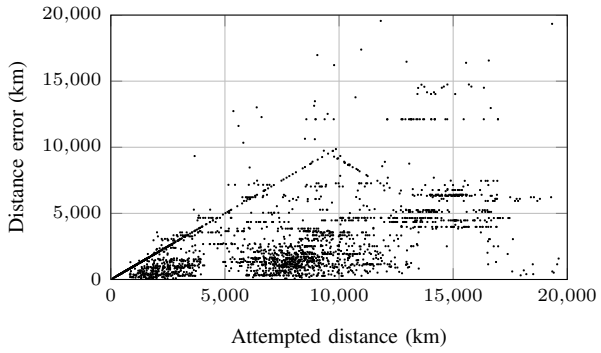
The correlation however is not extreme. For example, 50 % of all evasion attempts with attempted distance $\geq$ 5,000 km had distance errors below 470 km for CBG, and below 1,170 km for GeoPing. These results show that the newly discovered vulnerability enables even extremely remote adversaries to accurately relocate themselves (fraudulently).

B. Attack Detectability for CBG

We now analyze if the landmarks can detect the attack. Recall from Section II-A that CBG calculates the location as the centroid of a convex region enclosed by the intersection of multiple circles. The authors of CBG assume the area of this region represents the confidence factor of the calculated location, i.e. a larger area implies less confidence. Gill *et al.*



(a) Manipulating CBG



(b) Manipulating GeoPing

Fig. 10. Attempted distance versus distance error.

[5] suggested that the confidence factor could be used to detect their attack (which is an adversary increasing the RTT) because larger RTTs between a client and the landmarks lead to larger area of the intersection region. We follow Gill *et al.*'s steps, and analyze the area of the intersection region while attacking CBG. Figure 11 shows a CDF of the intersection regions' areas. A CDF of the intersection region areas while geolocating the true locations of the 51 nodes (see Figure 5) using CBG is plotted on the same chart for reference. As discussed earlier in this Section, we assume the adversary uses a speed of (1/3)**c** as an approximation to the one-way traffic propagation speed. Relative to the average traffic propagation speed [18], this is slightly slow. Thus, it results in relatively large $\beta(a', l)$ values (Eqn. 10), which elongates the distance that each landmark calculates based on this RTT. This explains the relatively large intersection areas shown by the chart of Figure 11.

Let's assume the area of the intersection region can be used to detect an adversary conducting this attack, as suggested by Gill *et al.* [5]. Despite the difference in areas of the two curves of Figure 11, one can see that 92% of the 51 true nodes' locations had the same region area as 71% of adversaries (evasion attempts) at $x = 0.2\times10^7$ km^2. This implies that if the geolocating party decides to reject clients with areas greater than this value, it would falsely reject 8% of legitimate clients and falsely accept 71% of adversaries. Similarly, a 0% false reject (i.e. at $x = 0.35\times10^7$ km^2) leads to 75% false accept. In conclusion, according to these results, detecting the proposed attack is not trivial.

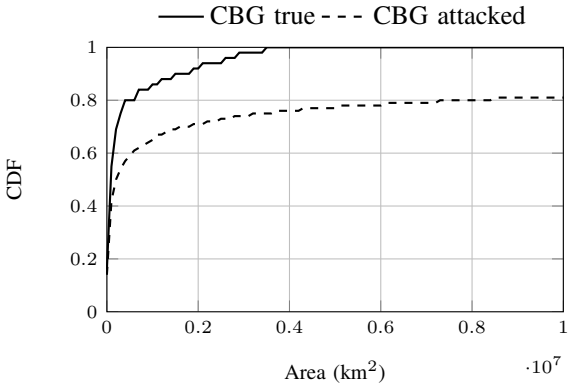


Fig. 11. CDF of the area of the constrained region. *CBG true* shows the areas while determining the true locations of the (51) adversaries in our test set; *CBG attacked* shows the areas for the (2,550) evasion attempt. A point (x,y) on the curve means a fraction y of all 2,550 evasion attempts had their intersection region areas of x km^2 or less.

C. Effect of the Added Ability of Decreasing the Delays

Gill *et al.* [5] studied the effect of an adversary increasing the RTT observed by a measuring party, by delaying the echo reply of ICMP packets. The vulnerability we explain therein also enables the adversary to decrease the observed RTT. To realize how much an adversary gains by also being able to decrease the RTTs, we compare the results obtained from our experiments to the adversary that can only increase the RTT, as explained by Gill *et al.*[5] We refer to the adversary that can increase and decrease the RTT as adversary *A*, and the one that can only increase as adversary *B*. Note however that adversary *B* (that of Gill *et al.*) uses a traffic propagation speed of (2/3)**c** to map delays to distance, adversary *A* (the one explained in this paper) uses (1/3)**c**. Gill *et al.* conducted 2,500 experiments (i.e. evasion attempts) on CBG [4]. Of our 2,550 experiments, 2,002 match their range of attempted distance (which is less than or equal to ∼11,000 km).

We begin by the distance error. Gill *et al.* [5] divided their experiments into two clusters based on the distance the adversary attempts to move. To reproduce similar experimentation design, we divided our experiments into two clusters almost in the same distance range as theirs. The first cluster included all distance attempts of 5,000 km or less; the second included attempts above 5,000 but equal to or less than 11,000 km. Figure 12 shows a CDF of the experiments carried out according to the distances attempted by the two adversaries, *A* and *B*, for each cluster. According to these charts (in Figure 12), we believe we have established a comparable experimental set up, where the effect of the adversary's added capability (decreasing the delays) could be scrutinized.

Figure 13 compares the distance errors of adversaries *A* and *B* for the two clusters. For the first cluster, 80% of the evasion attempts of adversary *A* resulted in distance errors below 900 km, versus 2,266 km for those of adversary *B*. The distance errors for both adversaries become quite large in the second cluster. About three-quarters of *A*'s evasion attempts

[5]All results belonging to Gill *et al.* [5] that we plot herein are obtained from the figures in that publication. We visually sampled 10 data points for each plotted curve. Those curves can be found in Figures 12, 13, 14, and 15 below.

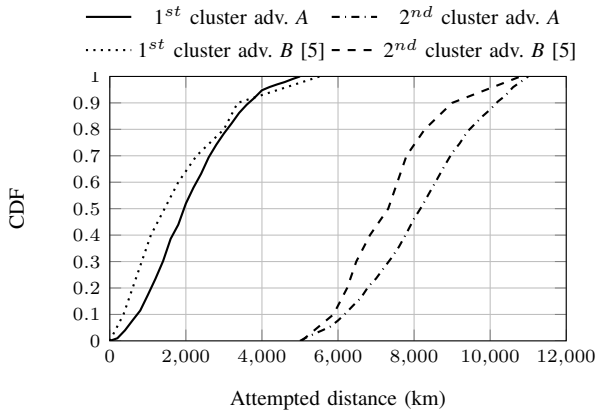


Fig. 12. CDF of the distances the adversaries attempted to move. A point (x,y) on the curve means a fraction y of all evasion attempts attempted to move x km or less away from the true location. Note: this graph shows experimental design, not results.

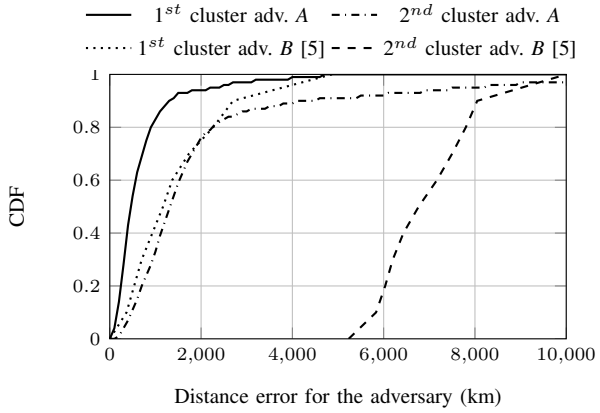


Fig. 13. CDF of the error in distance for adversaries *A* and *B* for the two clusters. Smaller distance error indicates more successful attack. A point (x,y) on the curve means a fraction y of all evasion attempts resulted in distance error of x km or less.

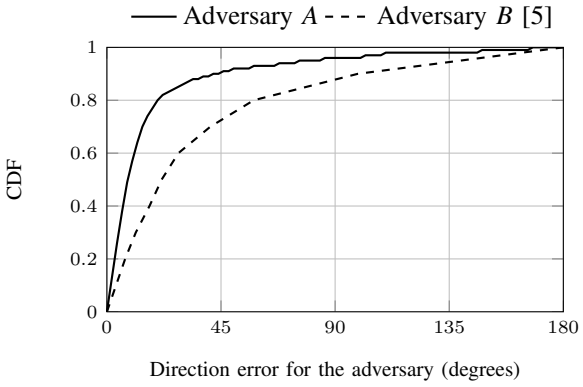


Fig. 14. CDF of the error in direction for our adversary and that of Gill *et al.* The results plotted on this chart are unclustered. A point (x,y) on the curve means a fraction y of all evasion attempts resulted in direction error of x degrees or less. Smaller direction error indicates more effective attack.

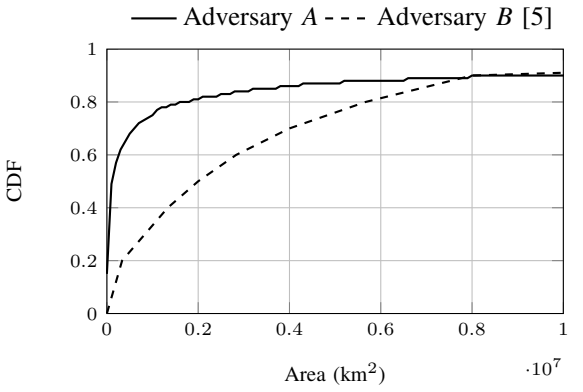


Fig. 15. CDF of the area of the constrained region for our adversary and that of Gill *et al.* The results plotted on this chart are unclustered. A point (x,y) on the curve means a fraction y of all evasion attempts had their intersection region areas of x km^2 or less. Smaller constrained regions indicate a harder-to-detect attack.

in the second cluster resulted in distance errors below 2,000 km; three-quarters of *B*'s evasion attempts had errors above 6,120 km. Also, in the same cluster, only 9% of adversary *A*'s evasion attempts resulted in a distance error above 5,000 km, versus all of adversary *B*'s evasion attempts. These results highlight that the new vulnerability disclosed herein makes the adversary significantly serious; the distance it attempts to move itself does not pose high impact on its distance error (as shown specifically by the 2^{nd} cluster), as was the case when the adversary was only able to increase delays.

Figure 14 shows the results for the direction error; 91% of adversary *A*'s evasion attempts resulted in direction errors below 45 degrees, versus 73% of adversary *B*'s attempts. A lower direction error indicates a more accurate (hence, more worrisome) attack.

Finally, because the vulnerability studied by Gill *et al.* allowed only increasing the RTT, the area of the intersection region constructed by the landmarks of CBG grows rapidly with the distance the adversary attempts to move away of its true location, which exposes the adversary's evasion attempts to being detected [5]. With the ability of decreasing the RTT, detection becomes harder. Figure 15 shows CDFs comparing the areas of the intersection regions for both adversaries. For adversary *A*, 19% of the evasion attempts resulted in areas above 0.2×10^7 km^2, 50% for adversary *B*. More than half (57%) of *A*'s attempts had areas below 0.025×10^7 km^2; 57% of adversary *B*'s attempts had areas above 0.151×10^7 km^2.

The impact on geolocation presented in this section shows the very serious nature of attacks when an adversary is able to decrease and increase the RTTs between itself and another party, as observed by that party. In our experiments, some geographically remote attackers managed to place themselves in the locations of their choosing, with distance errors below a few tens of kilometers. We are not aware of methods by which these attacks are easily detectable (but in the next section, we do discuss possible countermeasures). Comparison to an adversary that can only increase the RTTs showed how devastating it is if an adversary can fully manipulate the RTTs (increase and decrease).

V. Countermeasures

We now discuss possible countermeasures to defend against the proposed attack. The countermeasures suggested specifi-

cally aim at preserving the integrity of the delay-measuring phase prior to the operation of a geolocation algorithm. We stress that the vulnerability to attack explained herein is not of the ICMP protocol itself, rather in leveraging the protocol to carryout a task (geolocation) that it was not designed for. A high level countermeasure would therefore be to avoid using `ping`/`traceroute` in geolocation. If no other options exist, then the following countermeasures could be considered.

As discussed in Section III-A, the vulnerability lies in the adversary's ability to tamper with both, the sending (s) and receiving (r) times of echo request/reply packets of the `ping` and `traceroute` programs. Every landmark must ensure the integrity of both parameters. We begin with s.

Locally recording s enables a landmark to retrieve it from its memory instead of the DATA field of an echo reply packet. Obviously, the landmark's own local memory is more reliable than an unprotected packet returned from the receiver/adversary. This precludes the adversary from undetectably tampering with the value of s. If a stateless implementation is desired, landmarks may use a Message Authentication Code (MAC) protecting s, the ICMP identifier and the ICMP sequence number of the echo request, and place the MAC in the DATA field of the packet along with s (see Figure 3). Note that a landmark cannot include the TYPE and CHECKSUM fields in the MAC because the receiver has to change them in the echo reply, as specified by RFC 792 [2]. The landmark can store the key of the MAC locally. A single key suffices for multiple sessions. Landmarks can then verify the integrity of the timestamp s retrieved from a received echo reply.

As for the receiving time r, recall that a key factor of the success of an adversary to manipulate it is the adversary's ability to measure the waiting time between a successive pair of echo requests. Consequently, randomizing the waiting times raises the bar for the adversary to accurately predict this time. Such a precaution is simple to implement as it may not necessarily require modifications to the `ping` or `traceroute` programs. However, the adversary can still observe the minimum and maximum waiting times, calculate the average, and use the average as an approximation to the waiting time. Therefore, this precaution does not stop the adversary from undetectably manipulating r; it only increases the adversary's error range. Another countermeasure to preserve the integrity of timestamp r is to include a random *nonce* (a random number that is used once) in the DATA field of echo requests. For all practical purposes, a sufficiently large such nonce and ample entropy will prevent the adversary from issuing fraudulent echo replies; the adversary will be forced to wait to receive the echo requests first to obtain the nonce. This precaution is stateful; a landmark must store the nonce used with every created echo request until it receives the reply. Also, this solution does not prevent the adversary from increasing r simply by delaying echo replies [5]. Counteracting such evasion is hard to achieve without protocol-level changes.

VI. Related work

In 2010, Gill *et al.* [5] studied both topology-aware geolocation techniques and delay-based ones.[6] None of their attacks involved decreasing RTTs (round-trip times), which is a fundamental difference between their work and ours. They presented two adversaries, simple and advanced. The simple adversary had control over only its own machine (similar to our adversary in that sense, but different in the ability to decrease RTTs), while the advanced one controls a full network (representing an Autonomous System owner, Internet Service Provider or a cloud provider). They chose two of the most prominent techniques to analyze their vulnerability to evasion: CBG [4] as a representative to delay-based techniques and Octant [7] as a representative to topology-aware ones, and discussed the effect of both adversaries on the elected techniques [5]. They found that (1) the effect of both adversaries is similar on delay-based techniques, and (2) there is a tradeoff between the attempted distance and detectability of the attack, which matches our finding.

Muir *et al.* [19] investigated geolocation over the Internet from a security perspective, and enumerated a broad spectrum of tactics for an adversary to evade geolocation techniques, including using proxies to hide the IP address, and falsifying location records of public registries (*whois* databases, Domain Name Systems (DNS) LOC records [20], etc). They argued that despite a plethora of proposals to geolocate Internet hosts, none appears to be robust against all classes of adversaries. Our work is complementary as it provides concrete evidence, based on practical evaluations, supporting their assertion with respect to popular implementations of delay-based geolocation techniques.

Goldberg *et al.* [21] addressed the problem of Path Quality Monitoring, and devised a set of protocols to detect if an adversary sitting in the path between two end systems is manipulating their traffic from a networking perspective. Manipulation includes altering content, dropping/delaying packets or injecting ones. Although their research is motivated by the lack of integrity checking in popular network-monitoring tools such as `ping` and `traceroute` [21], their proposed solutions assume the collaboration of the two end systems using such tools. In our case, one of the end systems is the adversary itself and therefore collaboration cannot be assumed.

VII. Conclusion

We have shown how an integrity vulnerability allows a new attack on delay-based geolocation techniques that use `ping` and `traceroute` to obtain their delay samples. The attack exploits common implementations of `ping` and `traceroute`, as they do not provide integrity checking to the measured RTT (round trip time). An adversary is able to decrease the RTTs observed by the measuring party. We demonstrate the ideas with a publicly available machine that reduces the RTT by 2 more milliseconds with every subsequent `ping` packet (i.e. -2, -4, -6, etc). We also investigated the impact on two delay-based geolocation techniques—CBG and GeoPing. We are not aware of a means to reliably detect the attack; we proposed several countermeasures to mitigate it.

[6] Topology-aware geolocation techniques leverage network topology information to generate a richer set of constraints to geolocate clients.

Nonetheless, we expect it is likely that delay-based techniques will be vulnerable to other evasion tactics, and that the search for *secure* geolocation techniques remains challenging.

We continue to investigate vulnerabilities in Internet geolocation techniques by investigating the applicability of this attack and others on all classes of geolocation techniques in general. We hope this work raises awareness of the importance of devising a location verification mechanism—where the results of current state-of-the art geolocation techniques could be corroborated—and encourage further research in the area of secure geolocation over the Internet.

ACKNOWLEDGEMENTS

We thank members of the Carleton Computer Security Lab for their enthusiastic discussion on this topic. This research is supported by the Natural Sciences and Engineering Research Council of Canada (NSERC)—the second author through a Discovery Grant; the third through a Discovery Grant and as Canada Research Chair in Authentication and Computer Security.

REFERENCES

[1] Y. Wang, D. Burgener, M. Flores, A. Kuzmanovic, and C. Huang, "Towards street-level client-independent IP geolocation," in *the 8th USENIX NSDI*, 2011.

[2] J. Postel, "Internet Control Message Protocol," RFC 792 (INTERNET STANDARD), Internet Engineering Task Force, Sep. 1981, updated by RFCs 950, 4884, 6633, 6918. [Online]. Available: http://www.ietf.org/rfc/rfc792.txt

[3] D. Hu and C.-L. Wang, "GPS-Based Location Extraction and Presence Management for Mobile Instant Messenger," in *Embedded and Ubiquitous Computing*. Springer LNCS, 2007, vol. 4808.

[4] B. Gueye, A. Ziviani, M. Crovella, and S. Fdida, "Constraint-based geolocation of Internet hosts," *IEEE/ACM Trans. Netw.*, pp. 1219–1232, 2006.

[5] P. Gill, Y. Ganjali, B. Wong, and D. Lie, "Dude, where's that IP?: circumventing measurement-based IP geolocation," in *19th USENIX Security Symposium*, 2010.

[6] V. N. Padmanabhan and L. Subramanian, "An investigation of geographic mapping techniques for Internet hosts," *SIGCOMM Comput. Commun. Rev.*, vol. 31, pp. 173–185, August 2001.

[7] B. Wong, I. Stoyanov, and E. G. Sirer, "Octant: a comprehensive framework for the geolocalization of Internet hosts," in *4th USENIX NSDI*, 2007.

[8] M. Crovella and B. Krishnamurthy, *Internet Measurement: Infrastructure, Traffic and Applications*. John Wiley & Sons, 2006.

[9] A. Ziviani, S. Fdida, J. F. de Rezende, and O. C. M. Duarte, "Improving the accuracy of measurement-based geographic location of Internet hosts," *Computer Networks*, vol. 47, no. 4, pp. 503 – 523, 2005.

[10] R. Landa, R. G. Clegg, J. T. Araujo, E. Mykoniati, D. Griffin, and M. Rio, "Measuring the Relationships between Internet Geography and RTT," in *ICCCN*. IEEE, 2013.

[11] Z. Dong, R. D. Perera, R. Chandramouli, and K. Subbalakshmi, "Network measurement based modeling and optimization for IP geolocation," *Computer Networks*, vol. 56, no. 1, pp. 85 – 98, 2012.

[12] A. Ziviani, S. Fdida, J. Rezende, and O. Duarte, "Toward a Measurement-Based Geographic Location Service," in *Passive and Active Network Measurement*. Springer LNCS, 2004, vol. 3015.

[13] R. Percacci and A. Vespignani, "Scale-free behavior of the Internet global performance," *The European Physical Journal B - Condensed Matter and Complex Systems*, vol. 32, no. 4, pp. 411–414, 2003.

[14] "PlanetLab: An open platform for developing, deploying, and accessing planetary-scale services." [Online]. Available: http://www.planet-lab.org/

[15] H. V. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, and A. Venkataramani, "iPlane: An Information Plane for Distributed Services," in *Proceedings of the 7th Symposium on Operating Systems Design and Implementation*. USENIX Association, 2006.

[16] "Geodetic Calculation Methods - Geoscience Australia." [Online]. Available: http://www.ga.gov.au/earth-monitoring/geodesy/geodetic-techniques/calculation-methods.html

[17] H. Moritz, "Geodetic Reference System 1980," *Journal of Geodesy*, vol. 74, no. 1, pp. 128–133, 2000.

[18] E. Katz-Bassett, J. P. John, A. Krishnamurthy, D. Wetherall, T. Anderson, and Y. Chawathe, "Towards IP geolocation using delay and topology measurements," in *the 6th SIGCOMM IMC*. ACM, 2006.

[19] J. A. Muir and P. C. van Oorschot, "Internet geolocation: Evasion and counterevasion," *ACM Comput. Surv.*, vol. 42, pp. 4:1–4:23, 2009.

[20] C. Davis, P. Vixie, T. Goodwin, and I. Dickinson, "A Means for Expressing Location Information in the Domain Name System," RFC 1876 (Experimental), Internet Engineering Task Force, Jan. 1996. [Online]. Available: http://www.ietf.org/rfc/rfc1876.txt

[21] S. Goldberg, D. Xiao, E. Tromer, B. Barak, and J. Rexford, "Path-quality Monitoring in the Presence of Adversaries," in *ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*. ACM, 2008.