Secure Client and Server Geolocation over the Internet

ABDELRAHMAN ABDOU, PAUL C. VAN OORSCHOT

AbdelRahman Abdou is a Postdoctoral Researcher in the Department of Computer Science at ETH Zurich. He received his PhD (2015) in systems and computer engineering from Carleton University. His research interests include location-aware security, SDN security, and using Internet measurements to solve problems related to Internet security.
abdoua@inf.ethz.ch

Paul C. van Oorschot is a Professor of Computer Science at Carleton University, and the Canada Research Chair in Authentication and Computer Security. He was the program chair of USENIX Security 2008, NDSS 2001–2002, and NSPW 2014–2015; a co-author of the *Handbook of Applied Cryptography*; and a past Associate Editor of *IEEE TDSC*, *IEEE TIFS*, and *ACM TISSEC*. He is an ACM Fellow and Fellow of the Royal Society of Canada. His research interests include authentication and Internet security.
paulv@scs.carleton.ca

We provide a summary of recent efforts towards achieving Internet geolocation securely, that is, without allowing the entity being geolocated to cheat about its own geographic location. Cheating motivations arise from many factors, including impersonation (if locations are used to reinforce authentication) and gaining location-dependent benefits. In particular, we provide a technical overview of Client Presence Verification (CPV) and Server Location Verification (SLV)—two recently proposed techniques designed to verify the geographic locations of clients and servers in real time over the Internet. Each technique addresses a wide range of adversarial tactics to manipulate geolocation, including the use of IP-hiding technologies like VPNs and anonymizers, as we now explain.

Internet geolocation is the process of determining the geographic location of an Internet-connected device. Secure geolocating of a web client (a client visiting a website) is useful for location-aware authentication, location-aware access control, location-based online voting, location-based social networking, and fraud reduction. From the client's perspective, geolocating the remote server can provide higher assurance to the server's identity, and justify conducting certain sensitive transactions—for example, those requiring certain privacy measures or requiring data sovereignty [1]. Independent of server and client geolocation, geolocating network intermediate systems (e.g., routers) can also be useful for monitoring [2] and network mapping [3].

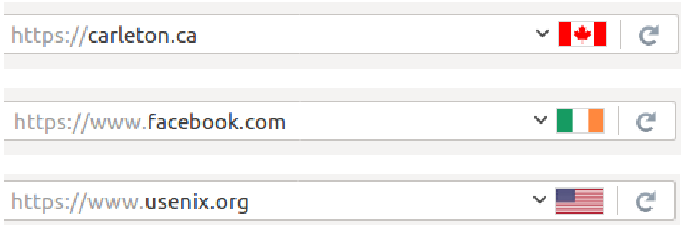
Both CPV and SLV are based on network measurements, where delays are measured from trusted network nodes dubbed *verifiers* and are analyzed in real time to verify physical presence inside a prescribed geographic region. We explain the threat model of both techniques, how they militate against known adversarial tactics, how they adapt to various network dynamics, and what distinguishes them from other geolocation approaches.

Geolocation Background

Many academic geolocation methods have been proposed, but there has been very limited deployment in practice. As of this writing, most of the geolocation conducted in practice relies on the clients' IP address or GPS coordinates of hand-held devices, as explained below.

Geolocation in Practice

There are several methods for device geolocation over the Internet. If the device belongs to a user that is acting as a web client (i.e., visiting a website), the Geolocation API is a W3C standard that enables browsers to obtain location information of the device they are running on and communicate it to a webserver. Servers request location coordinates using JavaScript as follows:



https://carleton.ca

https://www.facebook.com

https://www.usenix.org

Figure 1 [a-c]: Snapshots of the Flagfox browser extension

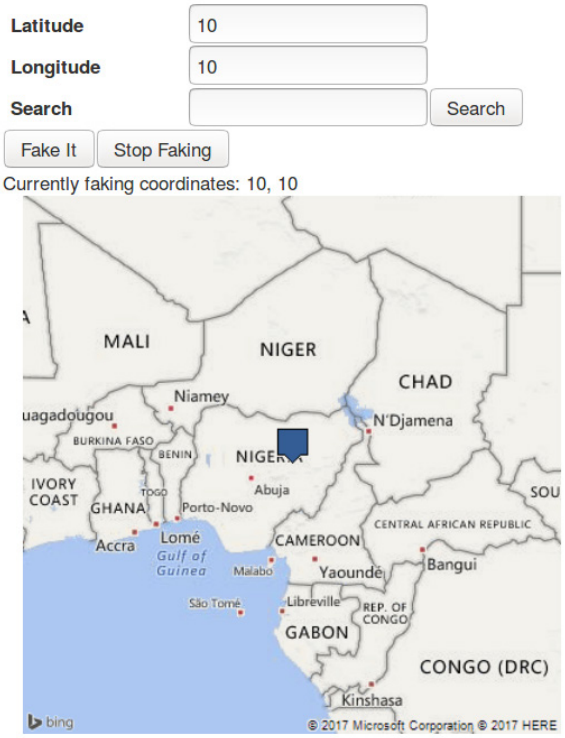
```
if(navigator.geolocation) {
    navigator.geolocation.getCurrentPosition(success, error,
geoOptions);
  } else {
    console.log("Geolocation is not supported on this
browser.");
}
```

The geolocation methods a browser uses are left to the browser vendor's discretion. Most major browsers rely on the following in varying orders (that is, when one fails, the next is tried): GPS, WiFi Positioning System (WPS), IP address-based location lookups, or cell-tower triangulation of mobile devices. The location of an IP address can be obtained from publicly available routing information or public registries, such as whois. Many IP location service providers (commercial and free) maintain lookup tables to instantly map IP addresses to locations. Such static *tabulation* methods may take long times to reflect changes or IP address reassignments, which occur quite often for client geolocation to be up-to-date (studies were conducted to confirm this [4]). IP address-based geolocation can, however, be reliable for benign server geolocation. *Flagfox* is an example Firefox extension that visually indicates a flag of the country corresponding to the IP address resolution of the URL (Figure 1).

From a security perspective, none of the above techniques is resilient to adversarial manipulation. When the geolocation API is in use, the server normally makes no effort in geolocating the client device; it rather trusts the browser-communicated coordinates, which can easily be forged on the fly before being sent to the server. Firefox extensions that enable forgery include *Fake Location* (Figure 2) and *Location Guard*; both enable a user to specify where in the world they would like to appear to be. If the server relies on tabulation methods to geolocate the client (instead of asking the browser for its coordinates), the common practice of clients hiding their own IP addresses behind proxies and anonymizers comes into play.

Geolocation in the Literature

A wide set of techniques can be used, mostly for a server to geolocate clients [5]. These enable a server to infer a client's geographic location from hints obtained from browser-generated HTTP headers such as preferred language or time zone. Locations can also be obtained through crowd-sourcing by interpolating a device's location from its proximity to nearby devices, like phones or WiFi access points (APs), with known GPS locations.



Latitude 10

Longitude 10

Search

Currently faking coordinates: 10, 10

Figure 2: Snapshots of the Fake Location extension—an example browser extension allowing users to fake their locations

Another class of Internet geolocation approaches is based on network measurements. Similar to GPS triangulations that are based on the delays between the receiver and satellites, measurement-based techniques also aim to locate devices (clients or server) by estimating their distance from landmarks in the network with known locations. These landmarks measure network delays from themselves to the device, typically identified by its IP address, and map these delays to geographic distances. The accuracy of such mapping, however, is not anywhere near that of mapping satellite delays to distances, and is thus the primary source of inaccuracies in such techniques. Still, measurement-based geolocation is generally considered more accurate than methods like tabulation-based geolocation.

From the security point of view, although most of the above methods are positioned as resilient to evasion, examination has shown otherwise. Delay-increasing attacks can allow an adversary to distort its perceived location [6]. Delay-decreasing was also studied, for example, by manipulating ICMP "ping" and "traceroute" as they fail to preserve the integrity of timing measurements.

Combining both attacks, an adversary can forge the calculated location to an accuracy of a few tens of kilometers relative to a target desired location [7].

Client Presence Verification—CPV

CPV [8] is a measurement-based technique designed to verify the geographic locations of web users (clients) over the Internet. The client is assumed to be motivated to misrepresent its location to gain location-dependent benefits. CPV's design takes into consideration various adversarial location-forging tactics, including delay manipulations and IP-hiding technologies like VPNs and anonymizers. CPV does not rely fundamentally on the clients' IP addresses, nor does it determine geographic locations. Rather, it *verifies* an asserted (unverified) location, typically made by a client. The client's location could be asserted using the client's GPS coordinates, the client's IP address, or even explicitly asking the user to fill-in their street address in an online form during login.

To verify location assertions, CPV relies on an infrastructure of geographically scattered nodes, dubbed *verifiers*. The technique works as follows. When a client visits a website and asserts the geographic location from which he/she is currently browsing, three verifiers surrounding the asserted location are selected. The verifiers measure (in real time) network delays between themselves and the client's browser, and analyze these delays to corroborate that the client is present somewhere inside the triangle determined by their (the verifiers') geographic locations. Because the verifiers cannot pinpoint where exactly the client is within the triangle, the size of the triangle is the verification granularity.

Secure One-Way Delay Estimation

The verifiers do not measure round-trip times (RTT) between themselves and the client. Rather, they estimate the smaller of the forward and reverse one-way delays (OWDs) between each of them and the client. The larger OWD is discarded because propagation delays between two network nodes are bounded by the physical distance between them, so a smaller OWD measurement is a better representation to the geographic distance between both nodes than the larger—the larger must have been affected by other factors such as network congestion or circuitous routing.

To measure the OWD between a verifier and the client, CPV does not rely on standard OWD-estimation protocols like OWAMP (RFC 4656), as those require honest client cooperation: for example, client clock synchronization and honest reporting of delays. As such, CPV relies on the *minimum-pairs* (MP) protocol [9]. MP requires the three verifiers, A, B, and C, to first synchronize their clocks and pre-share cryptographic keys to ensure operational integrity.

Through JavaScript, the client's browser is first directed to establish a WebSocket (RFC 6455) connection to the three verifiers, which are chosen based on the client's asserted location. Verifier A begins by sending a cryptographically protected timestamp (in millisecond precision) to the client, which the browser forwards to the other two verifiers. On receiving this, verifier B calculates the propagation time from A → client → B, and likewise when the timestamp is received by C. Verifiers B and C then follow suit, taking turns in sending timestamps. When all three verifiers are done exchanging timestamp messages, they will have six delay values as follows:

- A→ client → B
- A→ client → C
- B→ client → A
- B→ client → C
- C→ client → A
- C→ client → B

Between each pair of verifiers, e.g., between {A → client → B} and {B → client → A}, the verifiers exclude the larger OWD and solve a system of three equations simultaneously for an estimate to the smaller OWD between the client and each verifier. That is, if the smaller of the forward and reverse OWD between the client and A, B, and C, respectively, is *a*, *b*, *c*, then (note: = sign here is used to indicate mathematical equality rather than an assignment operator):

- $a + b = min(AtB, BtA)$
- $a + c = min(AtC, CtA)$
- $b + c = min(BtC, CtB)$

where *AtB* is the delay A → client → B, and so on. Analysis of MP's accuracy showed that the protocol is likely to provide more accurate estimates to the smaller OWD than simply using half the RTT [9].

Corroborating Presence Inside the Triangle

In order to avoid potential inaccuracies from delay-to-distance mapping, the calculated OWDs are not mapped to distances. Rather, they are compared to the smaller OWDs between the verifiers themselves, which are measured and updated periodically in a background process, independent of whether or not a client is currently being verified. Assuming x = min(AB, BA) is the smaller of the forward and reverse OWDs between verifiers A and B *directly* (not to be confused with min(AtB, BtA) from the previous section), and likewise y = min(BC, CB) and z = min(AC, CA), then the client's asserted location is accepted as inside the triangle if:

$$area(\Delta xab)+area(\Delta ybc)+area(\Delta zca) \le area(\Delta xyz) + \epsilon$$

such that area (Δxab) is the area of that triangle calculated from its side lengths *x*, *a*, and *b*. The value of ϵ is used to account for the two extra access network traversals occurring at the client when the timestamps propagate from a verifier to the client to another verifier.

Iterative Delay Measurement

To account for abrupt delay spikes or network irregularities, the above process of OWD calculations and comparison with those between the verifiers is iteratively repeated *n* times. If the condition is met for the majority of the conducted iterations, the location assertion is accepted.

CPV Calibration

There are several parameters that tune CPV's reaction to events. The most important three are ϵ, *n*, which is the number of delay measurement iterations, and τ, which is the fraction of those iterations that must *pass* if the condition is met for the client's asserted location to be accepted. This calibration should take place before the location verification process begins. To do that, the three verifiers may use network nodes that they know as a ground truth to be inside the triangle. From the network delays of these nodes, the verifiers compute values for the above-mentioned three parameters and then run CPV to verify a client's location.

Hindering Illicit Traffic Relaying

In an attempt to defeat geolocation, a middlebox (like a proxy server or a VPN gateway) that is physically inside the triangle can be specifically customized to filter out the verifiers' timestamps from the client's traffic and forward them to the verifiers on behalf of the client. This threat against CPV is exacerbated by the presence of numerous cheap public VPN providers whose primary service is to enable subscribers to evade geolocation technologies.

Techniques like CPV can mitigate this by adapting known proof-of-work techniques [10]. The verifiers generate a cryptographic client puzzle with each timestamp message, which the client's browser must solve before forwarding the message (puzzle solution and timestamp) to the other two verifiers. The puzzles must be easy to solve so that they do not (1) overwhelm the client with high processing costs and (2) overshadow the network propagation delays. In the case of a middlebox connected to many simultaneously cheating clients, the middlebox will choose to either solve these puzzles on behalf of the clients or forward them to the clients. In the latter case, the network delay between the middlebox and the client will get added to the time the verifiers observe for location verification, which results in CPV correctly detecting the client's absence from the respective triangle. It is thus in the middlebox's interest to choose the former case—solving the puzzles on behalf of the clients. However, this means that as more clients are connected, the middlebox will have to solve more puzzles. When these puzzles begin to accumulate, they will increase queueing delays, which contribute to the delays observed by the verifiers, eventually causing CPV to reject the location assertions of all middlebox-connected clients.

In this model, there are two main parameters contributing to the puzzle queueing rate at the middlebox: the puzzle difficulty and the middlebox's computational resources. Queueing analysis [10] shows that the puzzle difficulty has a higher impact on the rate of puzzle queueing than the middlebox's computational power. This analysis suggests that this puzzle mechanism will effectively hinder illicit middlebox relaying.

Evaluation Results

CPV was evaluated using PlanetLab—a distributed testbed for Internet measurement research and network experiments—using 80 PlanetLab nodes in North America. Three of the nodes were selected to act as verifiers, and the remaining 77 acted as clients. Some of the 77 nodes were inside the triangle and others were outside. All 77 nodes carried out the protocol with the verifiers simultaneously to get their locations verified. Knowing the ground-truth of which nodes were inside and which were outside (the geographic locations of PlanetLab nodes are publicly disclosed on PlanetLab's website), we could count the number of false rejects, nodes inside the triangle identified by CPV as outside, and false accepts. The process is repeated after choosing a different triangle, a different set of three nodes to act as verifiers, again counting false rejects and false accepts. In total, 34 triangles where chosen. Triangles were chosen to be nearly equilateral (physically), with inside angles ranging from 50–70 degrees (0.87–1.22 radians). The smallest triangle had an area equivalent to a circle of radius 100 km, and the largest of 400 km.

When the inside nodes were not too close to the triangle's sides, that is, away from the closest side by at least 10% of its length, CPV resulted in a total of 1.0% false accepts and 2.0% false rejects [8]. These results were obtained when $n = 600$ CPV iterations were performed with each client. The results were not much different when only 100 iterations were performed, where the false accept rate increased only to 1.1% and the false reject rate remained unchanged. However, when only 10 iterations were performed, false accepts and false rejects were at 2.1% and 4.1%, respectively.

Testing was later repeated to assess the effect of WiFi access networks on CPV's efficacy [11]. WiFi access networks often have higher delays and delay jitters. A different evaluation technique was used, as the PlanetLab infrastructure used above involved nodes connected using wired access networks. To model WiFi clients, 802.11 delay models from the literature were used to generate the last-mile delays, which were added to the delay traces collected from PlanetLab. Since higher network delays for nodes inside the triangles may result in higher false rejects, the generated 802.11 delays were only added to the delays of inside nodes to create the most stressful testing situation. 802.11 networks employ slotted retransmissions. The delays were generated such that each slot was 20 μsec, the propagation delay from the device to the wireless gateway was 1 μsec, and the four other wireless devices were continuously competing for the wireless media along with each wireless CPV client. With these parameters, CPV's false accepts were at 2% and false rejects at 4%. Although CPV's efficacy was affected by the WiFi access network, increasing the number of iterations can improve the results (see [11]).

Live Demo

A live demo of CPV is currently running on http://cpv.ccsl.carleton.ca. This link hits a webserver in Ottawa, Canada, which enables clients to verify whether they are present inside a US-based triangle determined by verifiers in San Francisco, Las Vegas, and San Diego. The verifiers are provided by hosting services DigitalOcean, ServerPoint, and M5 Hosting. Each VM has a 500 MB RAM and runs Ubuntu 16.04. NTP is used to synchronize their clocks. Additionally, each verifier issues an NTP query every 30 minutes using the "ntpq" utility to calculate the clock offset with the other two verifiers, which is added to the calculated OWDs between the verifiers for more accurate OWD estimates. Each verifier issues a timestamp to the other two verifiers every six seconds for direct OWD measurements between the verifiers.

A Java implementation of a CPV verifier runs on top of a lightweight custom-written WebSocket server, which is also implemented in Java. When a location verification request is initiated, the verifiers first check that it was issued from the authentic server (the one based in Ottawa in that demo implementation), because this server digitally signs connection IDs when they are issued. Additionally, each exchanged timestamp message between the verifiers through the client is corroborated using an MD5-based HMAC (a stronger HMAC is recommended to be used in practice). For the currently running demo, eight delay-measuring iterations are performed, once every 300 ms. When all iterations are performed, the verifiers send the measured delays back to the Ottawa server, which processes the result and returns it to the browser as a jQuery response.

No client puzzles are implemented yet in this demo as of this writing, nor is any automatic calibration of CPV's parameters. Instead, the main server has manually set parameters of $\epsilon = 10$ ms and $\tau = 0.7$, which are static and used across all clients.

Server Location Verification (SLV)

Analogous to CPV but on the server side, SLV [12] works by finding evidence of a server's physical presence inside a geographic region by measuring the server's network delays. A browser typically communicates with an SLV Manager, which orchestrates a network of server location verifiers. The challenges faced in doing so are quite different from verifying clients: (1) clients do not normally have the ability to write and run code on the server, whereas that was easily achievable by the server on the client, typically using JavaScript; (2) the common physical distribution of web content using content distribution networks (CDNs) and replication technologies begs the questions: *Of the multiple physical servers that may serve client content, which such servers should be selected to geographically locate (verify) in order to provide a useful server-authentication service? How should that machine be identified?*

The answers to these questions depend on the threat model and the application for which geolocation is to be used. Since the goal of SLV is to reinforce server authentication, the implementation of SLV takes the view that the first machine that terminates the client's TCP (and TLS) handshake is the most critical one. The protection provided from verifying that first machine would be comparable to that provided by TLS in the cases where the browser fetches content from multiple machines, some of which are not TLS-protected: for example, a page with mixed content.

For deciding on the mechanism used to identify machines, it is important to dissect man-in-the-middle (MITM) and server impersonation attacks. In MITM attacks, an adversary hijacks network traffic intended for the authentic server and relays it to the authentic server with or without modification. Hijacking could occur on several layers of the network stack as follows. (Note that using uncompromised TLS protects against the following hijacking cases; the value of using server location to reinforce server authentication is more profound for non-TLS-enabled websites or to catch attacks against the TLS system.)

- **Case 1: Attacker's machine has a different IP address than the authentic server**. In upper layers, phishing and pharming attacks are prominent traffic hijacking examples; the outbound traffic from the client has a different IP address from that of the authentic server. If the browser submits the domain name of the visited website to the SLV Manager, the Manager may resolve it to a different IP address from that seen by the browser (which could also occur benignly in the cases of CDNs). Verifying the geographic location of that IP address then becomes useless to the browser because a MITM adversary would go undetected. It is thus important to have the browser resolve a domain and submit the IP address to the SLV Manager.
- **Case 2: Attacker's machine has same IP address as authentic server**. In a lower layer hijacking, such as MAC table poisoning, ARP spoofing, and BGP spoofing, outbound traffic from the client has the same destination IP address as that of the authentic server. Such tactics are based on routing manipulation, so that traffic intended to the authentic server's IP address reaches a different network location (versus geographic location), which corresponds to the attacker's machine.

In comparison to upper layers, lower layer hijacking attacks tend to be more scalable, affecting a larger proportion of clients. For MAC table poisoning and ARP spoofing, the closer the attacker's machine is to the authentic server's network, the more the affected clients. Likewise, BGP spoofing can cause traffic hijacking at a global scale [13]. This implies that as with higher layer traffic hijacking attacks (discussed above), identifying the server by its IP address will likely allow the SLV Manager to detect whether the browser-intended machine is at a different geographic location from that asserted through a static location mapping previously obtained for that IP address.

Revisiting the above questions, if SLV targets the IP address as resolved by the browser of the first machine that the client initially handshakes, regardless of whether the browser will be instructed to fetch other content from different places later in the session, it can detect most of the above server impersonation attacks.

Verification Mechanism

After obtaining an unverified server location assertion, three verifiers surrounding that location are selected. The verifiers measure network RTTs to the server over several layers, including an application using HTTP request-response times and transport using TCP handshake responses. By means of comparing these delays with the delays between the verifiers, each pair of verifiers then verify whether the server is physically present inside the circle whose diameter is the physical distance between the verifier pair, and whose center is the midpoint between them (Figure 3).

Evaluation Results

Pilot testing of ~200 experiments was conducted on SLV using PlanetLab, half of which were true location assertions made by servers and the other half were false assertions. As with CPV, the rates of false rejects and false accepts were the fundamental evaluation parameters. SLV resulted in 0% false accepts and 2.4% false rejects [12]. Although the false reject rate may seem high for some applications, it can be improved by proper selection of verifiers, those with sufficient network bandwidth and processing resources.

SLV Browser Extension

We have built a Firefox browser extension to reinforce TLS by integrating the webserver's verified physical location, as described above, into the server authentication process. The extension sends the IP address of the server to the SLV Manager and receives the location verification result. The extension uses FlagFox to obtain an unverified assertion for the server's location. It also displays a flag in the URL (Figure 1) and a green tick mark or a red cross indicating whether the location asserted by FlagFox is true (according to SLV's verification) or not. This process takes a few seconds to execute, during which a throbber is displayed by the flag instead. Note that such visual cues are only meant as visual feedback in prototypes and are not an indication that we would expect end-users to base decisions upon. See below for how policies could be implemented to automatically make decisions on behalf of users.

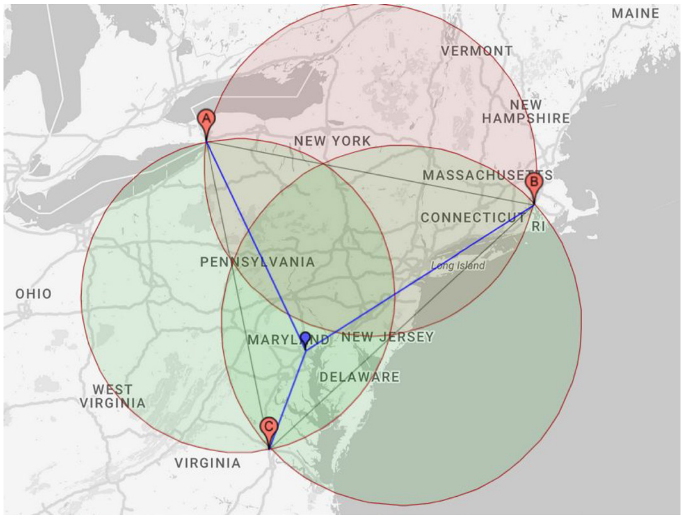


Figure 3: Server Location Verification (SLV) using network measurements from three verifiers (A, B, and C) to a server. Please view the online version of this article to see the figure in color. Map data: Google, INEGI.

Server Location Pinning in the Browser.

To avoid having the user interpret visual icons, the SLV extension is supported with a *location pinning feature*, whereby a browser saves the fact that a website identified by its URL was previously verified to host content from a particular geographic location, analogous to key pinning [14]. Although location verification is performed based on the IP address, the SLV Manager only receives an IP address from the browser, with location pinning in the browser based on the domain name. Upon receiving the verification result for a website, its location gets pinned only if the result is positive. This operation follows a trust on first use (TOFU) concept.

In general, for interpreting a received verification result, the SLV extension checks whether that location to some degree of geographic precision was pinned before for that website. The result of the operation falls into one of three categories: Critical, Suspicious, or Unsuspicious. Critical means the verification result for a previously pinned location was negative. A Suspicious outcome occurs when the location verification result is negative, but no location was previously pinned for that website . Finally, an Unsuspicious outcome is when location verification passes for a domain that was not previously pinned. Note that these are only meant to illustrate how a client might utilize SLV, but we expect different applications would make different choices.

Such outcomes could result in the browser automatically taking decisions through a policy-based engine. An example would be to instruct the browser to block/terminate the connection for all Critical outcomes of the user's personal banking website. Such terminology is subject to more research scrutiny and is not yet part of the above-described SLV extension.

Conclusion

This article provides a technical overview of recent advancements in the field of secure geolocation over the Internet. Two technologies, CPV and SLV, were explained to address client and server geolocation, respectively. Both rely on network timing measurements for secure location verification, taking into consideration safety measures to limit adversarial manipulations. Of the wide variety of applications that may benefit from secure location information of clients and servers, reinforcing authentication (location-aware authentication) for both ends remains an important example. Future research on CPV and SLV includes further enhancing their accuracy in terms of the false reject and accept rates and their efficiency for large-scale deployments in practice.

References

[1] Z. N. J. Peterson, M. Gondree, and R. Beverly, "A Position Paper on Data Sovereignty: The Importance of Geolocating Data in the Cloud," in *Proceedings of the 3rd USENIX Conference on Hot topics in Cloud Computing (HotCloud '11)*: https://www.usenix.org/legacy/event/hotcloud11/tech/final_files/Peterson.pdf.

[2] B. Huffaker, M. Fomenkov, and k. claffy, "DRoP: DNS-Based Router Positioning," *SIGCOMM Computer Communication Review*, vol. 44, no. 3 (2014), pp. 5–13: http://www.caida.org/publications/papers/2014/drop/drop.pdf.

[3] A. Csoma, A. Gulyás, and L. Toka, "On Measuring the Geographic Diversity of Internet Routes," *IEEE Communications Magazine*, vol. 55, no. 5 (2017), pp. 192–197: https://arxiv.org/pdf/1601.01116.pdf.

[4] I. Poese, S. Uhlig, M. A. Kaafar, B. Donnet, and B. Gueye, "IP Geolocation Databases: Unreliable?" *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 2 (2011), pp. 53–56: https://inl.info.ucl.ac.be/system/files/paper_2.pdf.

[5] J. A. Muir and P. C. van Oorschot, "Internet Geolocation: Evasion and Counterevasion," *ACM Computing Surveys*, vol. 42, no. 1 (2009): https://www.ccsl.carleton.ca/~jamuir/papers/TR-06-05.pdf.

[6] P. Gill, Y. Ganjali, B. Wong, and D. Lie, "Dude, Where's That IP?: Circumventing Measurement-Based IP Geolocation," in *Proceedings of the 19th USENIX Conference on Security (Security '10)*, pp. 241–256: https://people.cs.umass.edu/~phillipa/papers/UsenixSec2010.pdf.

[7] A. Abdou, A. Matrawy, and P. C. van Oorschot, "Accurate Manipulation of Delay-Based Internet Geolocation," in *Proceedings of the ACM Asia Conference on Computer and Communications Security (ASIACCS '17)*, pp. 887–898: http://people.scs.carleton.ca/~paulv/papers/asiaccs-2017.pdf.

[8] A. Abdou, A. Matrawy, and P. C. van Oorschot, "CPV: Delay-Based Location Verification for the Internet," *IEEE Transactions on Dependable and Secure Computing (TDSC)*, vol. 14, no. 2 (2017), pp. 130–144: https://sce.carleton.ca/~abdou/CPV_TDSC.pdf.

[9] A. Abdou, A. Matrawy, and P. C. van Oorschot, "Accurate One-Way Delay Estimation with Reduced Client-Trustworthiness," *IEEE Communications Letter*, vol. 19, no. 5 (2015): http://people.scs.carleton.ca/~paulv/papers/OWD.pdf.

[10] A. Abdou, A. Matrawy, and P. C. van Oorschot, "Taxing the Queue: Hindering Middleboxes from Unauthorized Large-Scale Traffic Relaying," *IEEE Communications Letter*, vol. 19, no. 1 (2015): http://people.scs.carleton.ca/~paulv/papers/taxing-the-queue.pdf.

[11] A. Abdou, A. Matrawy, and P. C. van Oorschot, "Location Verification of Wireless Internet Clients: Evaluation and Improvements," *IEEE Transactions on Emerging Topics in Computing (TETC)*, vol. 5, no. 4 (2017), pp. 563–575.

[12] A. Abdou and P. C. van Oorschot, "Server Location Verification (SLV) and Server Location Pinning: Augmenting TLS Authentication," *ACM Transactions on Privacy and Security (TOPS)*, vol. 21, no. 1, (2017), pp. 1–26.

[13] R. Hiran, N. Carlsson, and P. Gill, "Characterizing Large-Scale Routing Anomalies: A Case Study of the China Telecom Incident," in *International Conference on Passive and Active Network Measurement* (Springer, 2013), pp. 229–238: https://people.cs.umass.edu/~phillipa/papers/Hiran_Pam2013_full.pdf.

[14] M. Kranch and J. Bonneau, "Upgrading HTTPS in Mid-Air: An Empirical Study of Strict Transport Security and Key Pinning," Network and Distributed System Security Symposium, Internet Society, 2015: http://www.jbonneau.com/doc/KB15-NDSS-hsts_pinning_survey.pdf.