Fighting Cybercrime with Packet Attestation

Andreas Haeberlen, University of Pennsylvania
Pedro Fonseca, Rodrigo Rodrigues, Peter Druschel, Max Planck Institute for Software Systems (MPI-SWS)

Technical Report MPI-SWS-2011-002

ABSTRACT

IP source addresses are often the only initial lead when investigating cybercrime in the Internet. Unfortunately, source addresses are easily forged, which can protect the culprits and lead to false accusations. We describe a new method for *packet attestation* in the Internet. Packet attestation establishes whether or not a given IP packet was sent by a particular network subscriber. This capability allows network operators to verify the source of malicious traffic and to validate complaints, identity requests, and DMCA take-down notices against their clients. As a result, innocent users cannot be falsely accused, while the culprits no longer enjoy plausible deniability. Support for packet attestation can be deployed incrementally by ISPs, and requires no changes to end hosts or to the network core.

1. INTRODUCTION

With cybercrime on the rise, governments, law enforcement agencies, and copyright owners are increasingly calling for better *attribution* on the Internet [25]. Here, attribution means the ability to identify whoever is responsible for sending a given set of network packets. With reliable attribution in place, it would be much easier to quickly establish countermeasures to an ongoing attack and to hold cybercriminals accountable for their actions.

Unfortunately, reliable attribution is difficult because there is no direct support for it in the Internet—packets do not have "license plates" that tie them irrefutably to the party responsible for sending them [8]. Moreover, adding such a capability to the Internet would face substantial technical, legal, political and administrative challenges. It would need strong user authentication and secure software on every network-attached device, and it would require an internationally accepted user certification authority and jurisdiction to bring criminals to justice no matter where they reside. It would also raise privacy concerns, and potentially undermine the Internet as a platform for whistleblowers and dissidents. Worst of all, such a capability would still not be sufficient to catch sophisticated cybercriminal who rely on multi-stage attacks through a series of compromised machines owned and operated by other victims [7].

Clark and Landau [8] recently made similar points and added that not only is strong attribution insufficient to cover multi-stage attacks, it is also not strictly necessary to investigate cybercrime. What victims and law enforcement agencies need most is evidence sufficient to (1) confidently take immediate steps to stop an ongoing attack, and (2) provide a starting point for an investigation of the responsible party. In serious cases, such an investigation must in any case rely on additional evidence, obtained through a search warrant or by following the trail of money, to be admissible in a court of law.

Today, network administrators and law enforcement agents rely on circumstantial evidence, such as log entries and IP addresses, to attribute traffic to a source. For example, if a server log contains an entry that indicates illegal traffic from a particular IP address, this IP address is combined with other information (such as IP prefix ownership and DHCP server logs) to identify a particular subscriber, who is then presumed responsible [28].

Relying on such evidence alone for attribution is problematic. First, the process involves several manual steps, which is labor-intensive and error-prone. The consequences of mistakes can be severe: innocent users can be falsely accused [11,36] or even go to jail [16]. Second, it is easy to create a perfect forgery of a log entry that incriminates an arbitrary user—all that is needed is that user's current IP address. Researchers at the University of Washington have demonstrated the problem by generating DMCA take-down notices against printers and wireless access points [27], but it is not difficult to imagine the damage a malicious attacker could do. Finally, when a real cybercriminal is caught, he can plausibly deny responsibility by pointing to the weakness of the evidence [12, 19].

In this paper, we propose to add a capability to the Internet that provides evidence of a packet's immediate origin. The evidence is strong enough to (1) enable swift and confident action to stop an ongoing attack, (2) convince ISPs to take action against a subscriber who appears to originate offensive traffic, (3) protect falsely accused subscribers, and (4) in the case of a multi-stage attack, provide a starting point for an investigation of the ultimately responsible party. Moreover, our proposed technique protects users' privacy and can be deployed today.

Specifically, we propose a primitive called *packet attestation*, which determines whether a given set of packets was sent over a given access link. Thus, packet attestation corroborates the evidence available today, by verifying that an alleged attack packet originated from the access link associated with the packet's source IP address. The primitive strikes a favorable tradeoff between the strength of the evidence it provides on the one hand, and its privacy implications and deployability on the other.

Attestations preserve user privacy because they only confirm what the querier already knows; a malicious querier would have to guess both the complete payload of a packet and its transmission time to learn anything nontrivial. At the same time, attestations corroborate the circumstantial evidence available about an attack. In case of a negative attestation, they protect users from false accusations. In case of a positive attestation, they give victims and network operators a basis for decisive steps to stop an attack, and law enforcement agencies a basis to start an investigation in serious cases.

Packet attestation does not require changes to the Internet architecture and can be deployed incrementally in the current Internet. Indeed, the technical requirements are surprisingly simple: it is sufficient to add a few strategically placed boxes that record a hash of every packet they see, and that store these hashes for a limited time. Its main overhead is storage; we estimate that a global deployment would require, on average, about 32 commodity hard disks per autonomous system. Packet attestation does *not* require changes to existing core routers, protocols, applications, or end hosts. It also does *not* increase packet sizes or require cryptography on the critical path.

To test whether a deployment of packet attestation would be practical, we have conducted a feasibility study to estimate the cost of a global deployment, and we have built and evaluated a prototype to measure both performance and possible adverse effects on key performance metrics, such as packet loss rate and jitter. Taken together, our results indicate that, at least in principle, a deployment of packet attestation would be feasible today.

The rest of this paper is structured as follows. We discuss related work in Section 2. In Section 3, we make the case for packet attestation in the Internet. Next, we describe a practical system for packet attestation called HAL in Sections 4 and 5, and we perform a feasibility study in Section 6. In Section 7, we discuss options for incremental deployment. Section 8 describes how HAL can be applied to automate complaint management at ISPs, Section 9 discusses possible variants and extensions, and Section 10 concludes this paper.

2. RELATED WORK

Spoofing defenses: The most common approach to attribution in the literature has been to defend against spoofing. For example, ingress and egress filtering [10, 18] check the source address at the edge, where the permissible address ranges are known. If a spoofing defense were ubiquitously deployed and diligently used, the recipient of a packet could be sure that the packet has the correct source address. However, unlike packet attestation, such a packet could not be used as evidence to convince a third party. If Alice complains to Charlie that Bob has been attacking her and presents the offending packet, Charlie cannot be sure that the packet was actually sent by Bob, since Alice could easily have fabricated it.

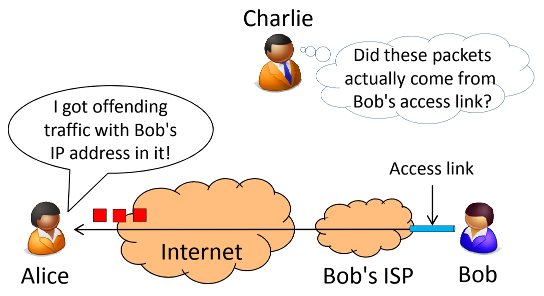
Traceback: Another approach to the spoofing problem is to mark each packet in the network in such a way that the recipient can determine its true origin, even if the source address is incorrect [20, 29, 35]. Traceback does not create evidence either: a malicious recipient could easily modify the packets or tamper with the markings such that they appear to have come from somewhere else.

AIP and Passport: Some of the more recent systems rely on cryptography to prevent spoofing: Passport [21] uses MACs and a key distribution system to detect spoofed packets in the middle of the network; AIP [1] introduces self-certifying addresses and a challenge-response mechanism for authenticating packets. As proposed, neither of these two systems creates evidence: Passport's hashes cover only the first eight bytes of the packet payload (which means that the rest of the packet cannot be validated), while AIP's challenge-response mechanism requires the cooperation of the sender to validate packets and therefore cannot authenticate packets from malicious senders. It seems feasible to extend AIP or Passport to also support the primitive we propose here; however, both systems would still require changes to the Internet core and to existing applications, while our proposed solution would require neither.

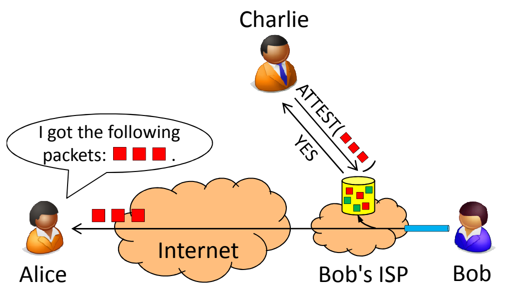
Collecting packets: This paper is not the first to propose that the network remember a digest of every packet. For example, single-packet traceback [35] does this as well, although it uses Bloom filters to limit the required storage and therefore cannot reliably identify a specific packet.

Application-layer authentication: Several application protocols provide some form of authentication; for example, TLS, HTTPS, and IPSec can authenticate either one or both endpoints. This can be used for attribution, but only for traffic that conforms to one of these protocols. HAL, on the other hand, works even when the attacker uses a non-authenticated protocol or floods the victim with random garbage.

Identifiers: Several papers have proposed changes


Charlie
Did these packets actually come from Bob's access link?
I got offending traffic with Bob's IP address in it!
Access link
Internet
Alice
Bob's ISP
Bob

(a) Complaint handling today



(b) Complaint handling with packet attestation

Figure 1: Today, a third party has no reliable way to verify a complaint (left). With packet attestation, the complainant can give the offending packets to the third party as evidence, and the third party can request an attestation to verify them (right).

to the Internet's addressing scheme such that each address contains a stable, globally unique endpoint identifier [1, 9, 26]. This change would eliminate the error-prone step of mapping IP addresses to endpoints. However, by themselves these proposals would neither prevent spoofing nor enable the recipient to convince a third party that a given packet is genuine.

Heuristics: Some systems use heuristics to attribute traffic to users. For example, HostTracker [38] tracks bindings between hosts and IP addresses over time by extracting application-level IDs from traffic logs. HostTracker's main focus is on analysis, so it need not identify all traffic, and a certain misclassification rate is acceptable. However, its output could be useful as a complement to HAL, e.g., to distinguish between multiple users behind an access link.

Prevention and mitigation: Off-by-default [3] is a clean-slate design of the Internet routing infrastructure, in which hosts explicitly declare the sources from which they are willing to accept network traffic; shut-off protocols enable receivers of offending traffic to reactively throttle the sender by sending a "shut-off" message [1, 2, 13, 15, 22, 32]. These systems can be used to prevent or mitigate certain forms of cybercrime, although most of them would require changes to the network and/or to end hosts. We note that, while HAL addresses a different problem, it can also be used to implement a shut-off service that is effective even against malicious senders and requires support at the sender's edge network only.

Network forensics: Systems like ForNet [31] solve the problem of reconstructing the events that led to an attack. These systems are complementary to HAL; an investigator could use forensics to understand which packets were part of the attack, and then submit these packets to HAL to obtain evidence.

PGPA [14] can determine whether a given packet was sent from a particular source address. This workshop paper sketches the design of a system that shares many of our goals, but differs in some important design decisions; for example, PGPA requires users to install a monitor box on their premises, which raises issues about cost and the possible destruction of evidence. Furthermore, the PGPA paper neither contains a detailed design nor a feasibility study or experimental results.

To the best of our knowledge, HAL is the first system that can *reliably* convince a third party (and not just the recipient) that a given IP packet is genuine, and allows access providers to validate complaints against their subscribers. Moreover, HAL does not require any changes to end hosts or network core and can be deployed independently by edge networks.

3. BACKGROUND AND OVERVIEW

In this section, we review the limitations in the current practice of attributing offending network traffic. Then, we show that packet attestation solves an important aspect of the problem, and we present a simple but effective way of implementing this capability.

3.1 Scenario

Figure 1a illustrates the basic scenario we are concerned with in this paper. There are three parties, Alice, Bob, and Charlie. Alice contacts Charlie and complains that she has received some *offending network traffic*, such as copyrighted content, spam, or attack traffic, apparently from Bob. How can Charlie decide whether the traffic actually originated from Bob's access link? Moreover, faced with an accusation that he has sent offending traffic, how can Bob prove his innocence in case Alice is mistaken or someone has spoofed Bob's IP address?

Variants of this scenario are frequently occurring in the Internet today. For instance, the role of Charlie is often played by an Internet Service Provider (ISP), and the role of Alice is played by 'netwatch' services such as REN-ISAC, Shadowserver, or SpamCop, or by lawyers representing copyright holders who file DMCA take-down notices. A large ISP, such as a university network, can receive hundreds of such complaints every month.

Note that a defense against spoofing is *not* sufficient to handle this scenario. If spoofing were impossible, Alice would learn the sender of any offending packets she receives, but from Charlie's perspective the situation would not change – he could still not be sure that Alice has actually received the offending packets (she might simply have made them up) and that the packets have not been modified somewhere along the path from Alice to Bob.

3.2 The state of the art

To convince Charlie that the offending network traffic originated from Bob's access link, Alice must present some form of *evidence*. Today, this evidence typically consists of log entries or IP addresses. For example, a complaint about spam might include the offending message, with full headers included, and a DMCA take-down notice might contain the date, time, and IP address of the alleged violation. The ISP then matches this information to its own records (often manually). This practice is not satisfactory, for at least three reasons.

First, the process is prone to clerical errors. For example, in 2004, two Wichita residents were accused of sharing child pornography because of mistaken information from their Internet service provider [36], and in 2003, the RIAA had to withdraw a copyright infringement suit against a Newbury resident because the file-sharing software she had allegedly used was not even compatible with her computer [11].

Second, log entries and IP addresses are easily forged. Recently, a team of researchers was able to provoke obviously incorrect DMCA take-down notices against printers and wireless access points [27]. An adversary could easily use a similar approach to falsely accuse an innocent individual, with potentially serious consequences for the victim, such as public embarrassment or even a criminal investigation.

Third, *because* log entries and IP addresses are so easily forged, they provide actual offenders with plausible deniability. For example, Roger Duronio, who was convicted of having planted 'logic bombs' in UBS PaineWebber network, claimed that some unknown third party had caused the damage but had used IP spoofing in such a way that the evidence would point to Duronio [12]. A similar argument was used in the case of Capitol vs. Thomas [19].

3.3 In search for better evidence

The approach we propose is based on three key insights. First, the key problem with using log entries as evidence is that they are *difficult to verify independently*. In the offline world, this is a key requirement for accepting evidence. If, for instance, Alice were to accuse Bob of keying her car, Charlie would not take Alice's statement at face value, but rather look for *someone other than Alice or Bob*, such as an eye witness, to *attest* to the validity of the evidence, i.e., to either confirm or deny Alice's claim. Our goal is to apply this successful, time-tested principle to the online world.

Who could Charlie ask to attest to the validity of Alice's evidence? Our second key insight is that that *this role naturally falls to Bob's ISP*. First, the ISP (potentially) has all the necessary information, since all traffic that Bob has sent through his access link must necessarily have passed through the ISP's network. Second, the ISP typically has a contract with Bob and is therefore in a position to attest to Bob's offline identity. Finally, if the ISP is a reputable business, it has little to gain from colluding with Bob, but a lot to lose (such as its business and its reputation), so it is reasonable for Charlie to trust its attestations.

Our third key insight is that Alice can *use the offending network traffic itself as evidence*. This choice has many advantages: The evidence is easy for Alice to collect (e.g., using `tcpdump`), it is easy for Bob's ISP to recognize, and it exists independently of the type of offending traffic. Most importantly, it is available today – without any changes to applications, protocols, end-user devices, or network infrastructure.

3.4 Proposed solution: Packet attestation

Based on the above three insights, we propose a primitive called *packet attestation*. At a high level, packet attestation works as follows (see also Figure 1b):

1. Each ISP records a hash of each packet (including payload) that is sent over one of its access links;
2. When Alice receives offending network traffic, she captures a packet trace and sends it to Charlie as evidence;
3. Charlie asks any attestation-enabled provider for the corresponding origin ISP. If Charlie recognizes the origin ISP as reputable, he sends the packets and their respective arrival times to that ISP and asks for an attestation that either 1) these packets were sent over some specific access link L that this ISP controls, or 2) these packets were not sent over any access link that this ISP controls.
4. If the ISP returns a positive attestation, Charlie believes Alice (and potentially takes further action, e.g., asking the ISP to identify the subscriber of L). Otherwise Charlie ignores Alice's request.

Of course, this overview glosses over many technical and nontechnical challenges; for example, the packet Alice receives is not identical to the packet Bob would have sent, since some header fields change in the network, and the ability to ask providers about packets sent by their customers raises privacy concerns. We discuss these and other challenges in Sections 4 and 5.

3.5 Is packet attestation the right solution?

Clearly, packet attestation is not as strong as personal attribution. Here, we briefly describe its limitations, and we discuss what it would take to remove them.

First, the availability and credibility of a packet attestation depends on the support and trustworthiness of the sender's ISP. This dependency could be removed by involving additional networks along the path; however, this would require support from components in the network core, which would make the system much harder to deploy.

Second, packet attestations by themselves cannot trace a multi-stage attack back to the source. This limitation could be removed by securely tracking data dependencies on end systems; however, this would require secure software on every network-attached device, which seems infeasible today.

Third, to obtain a packet attestation, the victim must have collected at least some packets as evidence. One could avoid this by storing a complete record of all transmitted packets (without hashing), which could then be searched retroactively. However, the existence of such a record would jeopardize user privacy.

Finally, packet attestations do not directly tie a specific individual to a specific cybercrime; they merely confirm that certain packets have been sent over a certain access link at a certain time. However, to tie packets to specific individuals, one would seem to require a global system of CAs that issued keypairs to everyone. To establish specific crimes, one would need international agreement on what constitutes a crime. Given the diversity of legal and political systems, such an agreement would seem out of reach.

To summarize: On the one hand, while it may be possible to build a system that avoids some or all of the limitations of packet attestation, such a system would face considerable technical and non-technical challenges. On the other hand, we have argued that many of these limitations can be overcome by combining packet attestations with established forensic techniques, such as following the money or executing search warrants. Packet attestation is 1) strong enough to address the problems with today's state of the art, such as clerical errors or false accusations; it is 2) potentially useful for verifying complaints, mitigating ongoing attacks, and guiding investigations; and it 3) preserves desirable properties, such as user privacy, incremental deployability, low overhead, and technical simplicity. Therefore, we believe that, in the spectrum of possible solutions, packet attestation occupies an attractive spot.

3.6 Roadmap

So far, we have made a case for packet attestation as the network's contribution to the attribution of cybercrimes. In the rest of this paper, we will:

- give a more detailed definition of packet attestation (Section 4);
- present the design of HAL, a packet attestation service for the Internet (Section 5);
- show that a global deployment of HAL in the Internet would be feasible (Section 6);
- describe how HAL could be deployed incrementally (Section 7);
- perform a case study to show how HAL could be used in practice (Section 8); and
- discuss possible variants and extensions of HAL (Section 9).

4. PACKET ATTESTATION

In this section, we describe the assumptions and the threat model we use in this paper, define an packet attestation request, and specify the properties packet attestation is designed to provide.

4.1 Assumptions

Our design is based on the following assumptions about providers that support packet attestation:

1. Providers use ingress filtering on their access links;
2. Providers loosely synchronize their clocks;
3. Each provider has a public/private keypair and a certificate that binds the keypair to its identity;
4. Each access link has a unique identifier.

The first two assumptions are current best practices. The third assumption could be satisfied in a number of ways, e.g., by using existing SSL certificates. The final assumption could be satisfied by using MAC addresses or any identifier currently used for billing.

4.2 Threat model

We assume that Internet customers may behave arbitrarily, but cannot steal or break the providers' cryptographic keys. In particular, customers can send offending network traffic to other customers, they can fabricate accusations against innocent customers, and they can try to use packet attestation to spy on other customers' traffic.

We make no assumptions about the nature of the offending traffic. It can be sent by any application and use any current or future protocol, it can be encrypted,[1] or it can consist of raw IP-level packets.

4.3 Attestation requests

Customers can request attestations by invoking a primitive called ATTEST. Given a provider P that supports packet attestation, a packet p, and a timestamp t, an invocation of ATTEST (P, p, t) at time t_0 returns

[1]To convince Charlie that the attested traffic is indeed offending, Alice can give him the necessary decryption keys.

- $\sigma_P(P, \text{YES}, L, p, k, t)$ if k copies of the packet p have been sent over access link L in the interval $[t - \Delta, t + \Delta]$;
- $\sigma_P(P, \text{NO}, p, t)$ if no subscriber of P has sent the packet p in the interval $[t - \Delta, t + \Delta]$;
- $\sigma_P(P', p)$ if the request should be directed to provider P' instead, because P' owns the source IP address recorded in p; or
- *nil* if $t \notin [t_0 - R, t_0]$.

Here, σ_P denotes a signature with the provider's private key. Two packets p and p' are considered identical if they differ at most in the IP header fields that normally change while the packet is forwarded in the network, i.e., the ToS and TTL fields, the IP checksum and any IP options that can change during forwarding.

The *tolerance* Δ is needed to account for propagation delays and imperfect clock synchronization; it should be at most a few seconds. The *retention time* R limits the required amount of storage space by allowing providers to discard data after R has elapsed; it should be on the order of several weeks. Note that, since the attestation is signed, Alice's evidence remains verifiable beyond R as long as she invokes ATTEST before R elapses. Reporting the number of identical copies (in the parameter k) is necessary to obtain attestations for certain denial-of-service attacks in which the same packet (e.g., a SYN) is sent many times in short succession.

4.4 How packet attestation protects privacy

A potential concern is that ATTEST could be used to compromise user privacy, that is, to find out *which* packets a given user has sent. Such an attack is theoretically possible, but practically very difficult because ATTEST only confirms information that the invoking entity already has. The attacker would have to guess an entire packet *and* the approximate time at which it was sent in order to get an affirmative attestation. Even if the attacker knows the time and the likely destination (say, `cnn.com`), and if he can guess a common packet (say, a TCP SYN or an ACK), he would still have to correctly guess all the header fields, including the source port number, the TCP sequence number, and the IPID. If all these fields are properly randomized,[2] the attacker would have to guess at least 64 bits, and thus should expect to send more than 10^{18} attestation requests before getting an affirmative response.

Another privacy-related concern is that an implementation of ATTEST might require the provider to keep extensive traffic logs, which could then be subpoenaed. As we will show in Section 5, this is not the case; it is sufficient to record a digest of each packet. These digests protect privacy in the same way as the narrow ATTEST interface. Even if someone gains offline access to the entire record, he can only confirm information he already knows. Anything beyond this would require guessing an entire packet, which is difficult.

[2]If the sender host does not do this, the provider can randomize the fields, similar to a NAT box (Section 5.6).

4.5 Properties

Given our assumptions and our threat model, packet attestation is designed to ensure the following:

1. **Evidence is available:** If Bob's provider correctly implements ATTEST and Bob has sent traffic T to Alice, Alice can (within the retention period) obtain a positive attestation that Bob sent T.
2. **No false accusations:** If Bob's provider correctly implements ATTEST and Bob did *not* send any offending traffic, nobody can obtain evidence of such traffic against Bob.
3. **Exoneration is possible:** If Bob's provider correctly implements ATTEST and Bob did *not* send T to Alice, Bob can (within the retention period) obtain an attestation that he did *not* send T.
4. **No privacy violations:** It is infeasible for Alice or any other customer to use ATTEST to find out which packets Bob has sent, except for packets that this customer has previously received or intercepted.

5. THE HAL SERVICE

Next, we describe the design of HAL (short for "Hashing at the Access Link"), a simple Internet service that implements ATTEST by recording digests of upstream packets at every access link.

HAL consists of two types of devices, monitors and coordinators. A *monitor* is a simple middlebox that is attached to a set of access links and records a hash and a timestamp for each upstream packet. A *coordinator* maintains the mapping from IP addresses to monitors. Each HAL-enabled provider has one coordinator and at least one monitor.

5.1 HAL coordinators

At any given time, each HAL coordinator is responsible for a specific set of IP prefixes. The coordinator also has a keypair for signing attestations about these prefixes, as well as a certificate that binds the keypair to the prefixes, the IP address of the coordinator, and a specific time interval. The certificate is signed with the provider's master signing key, which, for security reasons, is not stored on the coordinator itself and is only used offline, to sign certificates.

In addition, each coordinator internally maintains two lists, an *address list* and a *prefix list*. The address list records all IP addresses that have been assigned to access links during the retention time; it is populated, e.g., from the local DHCP server. The prefix list con-

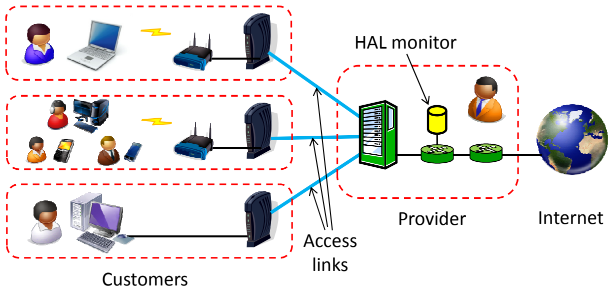


Figure 2: Example scenario. The HAL monitor is connected to a mirrored port and receives copies of all upstream packets.

tains a mapping from IP prefixes and timestamps to providers and their coordinators. This list is created and updated as follows. Each provider configures its coordinator with the address of the coordinators at a few other HAL-enabled providers, and when a new certificate is installed, the coordinator uses these addresses to flood the new certificate to all the other coordinators.

Since prefix assignments change rarely and the total number of certificates cannot exceed the number of providers, the bandwidth and storage cost should be small. Note that the coordinators can delete old certificates after the retention time has elapsed, but not before – if a ATTEST query occurs shortly after a prefix has been reassigned to another provider, the query must be directed to the "old" coordinator.

To obtain an attestation, Alice invokes ATTEST on an arbitrary coordinator (e.g., the local one, if her provider supports HAL). If this coordinator is not responsible for the relevant IP prefix, it consults its prefix list and redirects Alice to the responsible coordinator. The responsible coordinator then looks up the source address and timestamp in its address list to find the relevant monitor, queries that monitor, produces an attestation based on the monitor's response, and finally returns the attestation back to Alice.

5.2 HAL monitors

Each HAL monitor contains some stable storage, such as a harddisk or an SSD. When a monitor receives a packet, it records a digest and a timestamp in its storage. As we will show in Section 6, this simple operation can be performed at high speed with commodity hardware.

Monitors must be deployed in such a way that each upstream packet – that is, a packet sent from a subscriber to the Internet – is 1) received by at least one monitor, and 2) can be reliably associated with a specific access link. This can be accomplished in several different ways. The monitors can be directly on the data path, receiving upstream packets on one port and forwarding them to another port, or they can be connected to a mirrored port on a switch to get a copy of each upstream packet (Figure 2). They can be directly on the access link, next to a DSL access multiplexer (DSLAM) or a cable head-end (CMTS), or further upstream, receiving packets with MPLS tags from a DSLAM or CMTS. They can be separate boxes (server blades running special software) or their functionality can be integrated into existing equipment.

Although we have separated monitors and coordinators here for clarity of presentation, small providers could integrate the two components and implement them both on a single machine.

5.3 Packet digests

Monitors record digests instead of the entire packets because this saves space, and also because it mitigates the privacy risk if the monitor is subpoenaed. However, recall that the monitor records the packets at the source, whereas attestation requests refer to packets observed at the receiver or in the network. Since header fields like the TTL can change in the network, the digest must not depend on these fields; otherwise the hash of the evidence will not match the recorded hash.

To avoid this, we use the well-known technique from Snoeren et al. [35] and mask out any variant parts of the packet before calculating the digest, specifically the ToS field, the TTL value, the IP checksum, and any IP options that are not known to be invariant (the payload itself is invariant and can therefore be safely included). For the digest itself, we use a simple SHA-1 hash, but this choice is not critical; in principle, different monitors could even use different hash algorithms, as long as the requesters support all the algorithms and properly negotiate the correct algorithm during each request.

Monitors also record a four-byte timestamp for each packet, so the overall storage required per packet is 24 bytes.

5.4 Expiring old entries

Monitors store each digest until its minimum retention time R has elapsed. Setting the value of R involves a trade-off between the storage requirements and the ability to obtain an attestation a sufficiently long time after a packet was transmitted. For example, Alice may not discover an intrusion attempt until several days after the incident, or the monitor may be unreachable for some time. A setting of R on the order of one month should be sufficient for Alice to discover suspicious traffic and obtain an attestation; and, as we will show in Section 6.1, the resulting storage requirement is still reasonable. Note that once a packet is attested, the signed attestation remains valid for a long time.

5.5 Keeping time

Recall that all attestation requests include a specific timestamp, and that a negative attestation must be returned when the given timestamp is not within Δ of the recorded one, even if the digest is otherwise present. Thus, we must a) ensure that the monitors record the correct time, and b) choose Δ such that the reception time of a packet as observed by Alice is close enough to the recorded timestamp for ATTEST to succeed.

If the provider sets a monitor's clock incorrectly, its subscribers gain deniability for any packets they send, since they will be recorded with an incorrect timestamp and cannot be found by a request that contains the correct timestamp. To prevent this, monitors could periodically synchronize their clocks via NTP, or they could be equipped with a WWV receiver similar to the ones used in radio clocks.

To enable Alice to record useful timestamps when collecting evidence, we normalize all timestamps to the same timezone (UTC). Nevertheless, there will be some differences, due to both the propagation delay δ_1 and the (inevitable) discrepancy δ_2 between the monitor's clock and Alice's clock. Our goal is to choose Δ such that *typically* $\Delta \geq |\delta_1 + \delta_2|$; if the converse is sufficiently rare, Alice and the third party can simply vary the timestamps and retry a few times in case they get a negative answer. Most propagation delays in the Internet are below $1s$, and we can keep δ_2 below $2s$ if the monitor and Alice have access to a reference clock, e.g., a NTP server. Hence, $\Delta = 5s$ should be a safe choice; that is, it is long enough to avoid having to submit many requests to get a positive answer, and short enough not to impact attestation performance, and not to jeopardize privacy by increasing an attacker's chances of guessing a valid packet.

5.6 Adding randomness

Recall that in Section 4.4, we argued that HAL does not compromise user privacy because, even under very favorable conditions, an attacker would have to issue a very large number of queries in order to get any useful information. A crucial argument was that the attacker would have to guess at least one entire packet, including quasi-random header fields such as the source port number, the IPID, and the TCP sequence number, which together create a search space of 2^{64}. In practice, however, there are packets that lack some or all of these fields, and there are OSes that do not properly randomize them (for example, the IPID is sometimes set to zero). Moreover, if the attacker is very powerful or has additional information, even 64 bits may not afford enough protection.

If this is a concern, the monitor can add more randomness to upstream packets before it processes them further.[3] For example, the monitor could transparently replace the three header fields, analogous to a NAT device; some modern firewalls already randomize the initial TCP sequence number in this way to prevent attackers from hijacking TCP connections. Or the monitor could add a few extra random bytes to each packet, e.g., in the IP header after an end-of-options-list option, or – more cleanly – in a special HAL option. The second approach also works for non-TCP packets that do not contain sequence numbers and/or port numbers, such as UDP and ICMP packets.

[3]This requires the monitor to be on the data path, rather than on a mirrored port.

5.7 Attestation interface

Each coordinator implements a very simple attestation interface. An attestation request consists of a list of tuples (h_i, t_i) and an IP address S, where h_i is the hash of the ith packet, t_i is the corresponding timestamp, and S is the source address in the original packets. The coordinator forwards this information to the monitor to whose access link S was assigned at the relevant time; the monitor identifies the access link that was assigned address S at the relevant time, then opens the corresponding log, and finally checks for each tuple how many records containing h_i exist in the interval $[t_i - \Delta, t_i + \Delta]$. The monitor then returns this information to the coordinator. If no such tuples exist, the coordinator returns a negative attestation, otherwise it issues a positive one.

To enable customers whose local provider does not support HAL to obtain attestations, the attestation interface should be publicly accessible. However, a denial-of-service attack could cause online attestation requests to be dropped or delayed. To mitigate this problem, the coordinator can use standard DoS defenses; for example, it could ask each requester to solve a client puzzle [17], and it could increase the difficulty of the puzzles when it becomes overloaded. Note that DoS attacks can *only* affect online attestation requests. For high-stakes evidence, e.g., in a criminal investigation, it is always possible to physically retrieve a copy of the log stored in the monitor and inspect it offline.

5.8 Middleboxes on the path

Middleboxes can change the values of normally invariant header fields or even the bodies of IP packets. Thus, care must be taken to ensure that HAL can attest to packets in the presence of such boxes.

We first consider NATs and firewalls, the most common form of middleboxes. On the sender's side, it suffices to ensure that the HAL monitor is placed upstream of any NAT or firewall associated with a subscriber's access link; this is a natural position for the monitor. On the receiver's side, the best approach is to record offending network traffic outside the NAT or firewall, because such packets can be directly attested. Security-conscious organizations log traffic at this point anyway, because it allows them to record all traffic, including failed attempts to breach the NAT/firewall. Moreover, the most popular targets of offending traffic are servers, which have public addresses.

However, let us assume a case where the only evidence of offending traffic is recorded at a device inside a private network. Packets recorded at such a device would have a private local address and port. To enable a HAL attestation, the corresponding header fields must be translated to the external address/port number used by the NAT/firewall. Many middleboxes support the UPnP Internet Gateway Device (IGD) protocol [37], which can be used to obtain this information on-line. Failing that, the required information has to be looked up manually in the middlebox's logs.

Finally, let us consider transparent middleboxes that change the bodies of IP packets. Such devices are often used to reduce the bandwidth requirements of rich media content at the edge of cellular networks, and by some ISPs to insert advertisements. While such devices would defeat attestation of modified packets, this may be of little practical consequence. First, organizations running such devices would normally also monitor offending traffic at the public side of the device to protect their customers. Second, the middlebox does not modify control packets, and it is usually sufficient to obtain an attestation for one packet of a flow.

5.9 Resellers and multihomed customers

HAL actually supports a more general scenario than the one depicted in Figure 2. For example, some customers can choose between multiple providers that share the same access link. This happens, for instance, when one company has a dominant position in the access market and is forced by regulators to resell connectivity to other organizations. In this case, the company that owns the access links typically tags the traffic with a link identifier (e.g., a MPLS label) and then forwards the traffic to the corresponding provider. Thus, each provider could run its own monitor and use the link identifiers to associate traffic with specific customers.

Multihomed customers are another special case. If a customer has been assigned its own IP range but for some reason cannot run a monitor, his providers must interconnect their monitors; one of them acts as the primary monitor and forwards any requests it cannot answer to the secondary monitors. If the customer uses a different IP range for each access link, no special action is required.

6. FEASIBILITY STUDY

Next, we focus on the question whether a HAL deployment in the Internet would be feasible. We begin with a simple back-of-the-envelope calculation to estimate the cost of a global deployment; then we show that even a simple monitor device built from inexpensive commodity hardware would easily be able to handle the worst-case traffic on a DOCSIS upstream channel.

6.1 Global deployment

Recall that, from a technical perspective, HAL is extremely simple. If we assume for a moment that queries are relatively infrequent, HAL basically hashes each packet at the upstream link and stores the resulting hashes for some time. What would it take to do this for *every single packet* that is being sent in the Internet today?

A forecast performed by Cisco [6] estimates the global amount of IP traffic in 2011 at 28,491 PB per month. If we assume that the average packet size is 300 bytes (based on [34]) that would be about $9.5 \cdot 10^{16}$ packets per month. If we assume that we need 24 bytes of storage per packet (for a 20-byte hash and a 4-byte timestamp) and that each packet has to be remembered for one month, then it would take $2,279$ PB of storage, or about $1,140,000$ commodity hard disks, to store this data. Since the Internet consists of about $36,000$ autonomous systems [4], we arrive at about 32 disks per autonomous system, which certainly seems feasible.

What about hashing? `openssl speed` shows that a single core of a commodity CPU (Intel E8400 3GHz) can calculate SHA-1 hashes at about 3.17 Gbps; special-purpose hardware could probably do even better. However, since some CPU power is needed for I/O, we conservatively estimate the effective hashing throughput at 100 Mbps, i.e., about 3% of the maximum. Since the Cisco forecast suggests an average Internet traffic of $85,099$ Gbps, it would take only about $850,990$ of these cores, or about 24 per autonomous system, to hash the entire traffic. In other words, hashing is very very cheap.

Of course, this back-of-the-envelope calculation is far from perfect. For example, one would have to provision some additional storage to handle load spikes, and so on. But even if we have underestimated the cost by an order of magnitude, a global deployment of HAL still seems feasible.

6.2 Monitor prototype

To substantiate our claim about the achievable hashing throughput, and to provide some reassurance that we have not missed any major practical issues, we have built and evaluated a monitor prototype that is based on commodity hardware. Our prototype is implemented as a pair of Linux daemons. The first daemon has two threads: One reads raw packets from one network interface, associates them with a specific customer[4], stores their hashes in a memory buffer, and then forwards

[4]The daemon uses the source IP to map packets to customers, which assumes that the CMTS already performs ingress filtering. If not, an MPLS flow label could be used instead.

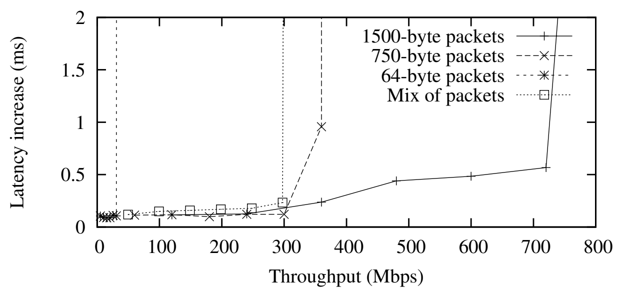


Figure 3: Forwarding performance for different packet sizes and traffic rates, and additional latency caused by the monitor.

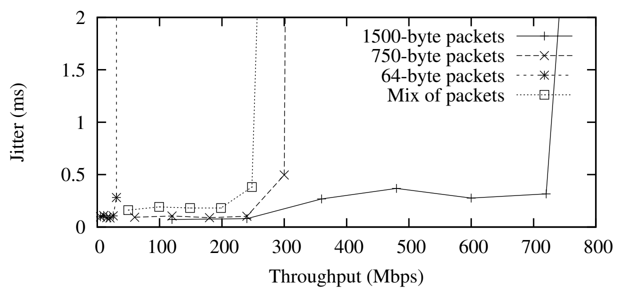


Figure 4: Jitter introduced by the monitor in a pass-through deployment. We define jitter as 95^{th} percentile delay minus minimum delay.

the packets to another network interface. The second thread periodically writes the contents of the buffer to disk sequentially. In addition to the logs, this daemon also creates a simple index that contains one timestamp for every $4,096$ hashes and the associated disk address.

The second daemon is responsible for answering any incoming attestation requests. It keeps the index in memory; when a request comes in, it performs a binary search on the index to find the start of a 2Δ window around the requested timestamps, and then sequentially reads at least 2Δ's worth of hashes and timestamps from the log. Finally, it scans the resulting data for the requested hashes and timestamps, counts the occurrences, signs a positive or negative attestation as appropriate, and returns the attestation to the requester.

Our prototype implementation consists of only $1,329$ lines of C code, as counted by the number of semicolons. When configured with a minimal Linux kernel and no other services running, the software is simple enough so that it can be hardened against attacks.

6.3 Experimental setup

We ran the prototype on a Dell Optiplex 760 workstation that was running a Linux 2.6.31 kernel. The machine had a 3.16 GHz Intel E8500 CPU, 3GB of memory, and two Gigabit Ethernet cards: One for incoming traffic and one for outgoing traffic. For storage, we used two external CnMemory hard disks, each with a capacity of 1.5 TB. In most of our experiments, we injected synthetic network traffic on the first card and captured the forwarded packets from the second card. The prototype was configured with a 2048-bit RSA key.

6.4 Performance: Recording

An HAL monitor must be able to sustain the worst-case aggregate upstream traffic of the access links that are connected to it. To measure the throughput our prototype can handle, we injected synthetic CBR traffic into the monitor. We varied the transmission rate in packets per second, using either a constant packet size between 64 and 1500 bytes or packet sizes drawn from a distribution observed at a regional ISP [34]. Each flow lasted for 60 seconds. Since a real monitor must handle recording and attestation at the same time, we flooded the monitor with attestation requests during the entire experiment.

Figure 3 shows the achieved throughput, as well as the median increase in latency caused by the monitor, compared to the minimum direct transmission time between the sender and the receiver. While the monitor is not saturated, the latency increase is very low; it reaches 570 μs for the largest packet size. After saturation, the latency increases due to queueing delays. The saturation points are between 39 Mbps for the smallest packet size and 700 Mbps for the largest; with the packet size mix, it is about 300 Mbps. Regardless of the packet size, saturation occurs at around $70,000$ packets/sec, which suggests that our simple user-level prototype is bottlenecked by the number of kernel/user transitions, and that a kernel-level implementation could achieve even better performance.

We conclude that even a simple monitor built from commodity-hardware can easily handle the worst-case traffic on a DOCSIS upstream channel. Furthermore, this setup should scale seamlessly to a large provider by placing a monitor next to each CMTS.

6.5 Performance: Loss and jitter

Next, we measured the loss rate and the latency distribution in the experiment described earlier; we report the 95^{th} percentile latency minus the minimum latency as jitter. Figures 4 and 5 show our results. Below the saturation point, the jitter is consistently lower than 0.5 ms, which is low enough to be compatible with jitter-sensitive protocols such as VoIP or TCP Vegas. We did not observe any packet loss at throughputs below the saturation point.

6.6 Performance: Attestation requests

Next, we measured the rate at which our prototype can answer attestation requests. To answer a request, the daemon must read at least 2Δ's worth of hashes around

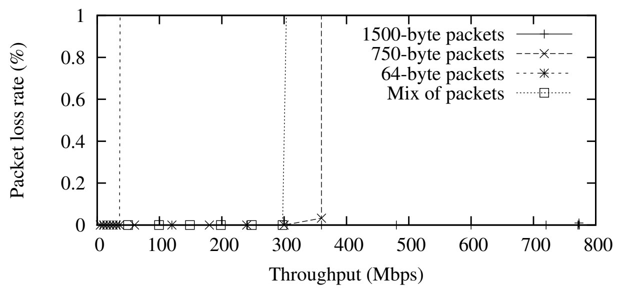


Figure 5: Loss rate introduced by the monitor in a pass-through deployment scenario.

Operation	*Sparse log*	*Dense log*
Disk reads	43 *ms*	70 *ms*
RSA signature	10 *ms*	10 *ms*
Other	1 *ms*	1 *ms*
Total	54 *ms*	81 *ms*

Table 1: Average response times for attestation requests when the log is dense or sparse.

the requested timestamp. The relevant location in the log can be found efficiently using the in-memory index, but the actual read must typically go to disk because the logs are much larger than main memory, so caching is not likely to be effective unless there is significant locality in the requested timestamps.

Since the size of the 2Δ window depends on the packet density in the log, we performed measurements with two different logs, a *dense* log with $50,000$ packets/sec (close to the packet rate that would saturate a DOCSIS upstream port with minimum-sized TCP packets) and a *sparse* log with $10,000$ packets/sec. The total size of each log was 1 TB. We saturated the daemon with attestation requests, whose timestamps we picked uniformly at random from the time period covered by the log, and we measured the average response times. Table 1 shows our results. The response time is dominated by the time required for the disk access, and it stays well below 100 ms, even for the dense log.

6.7 Summary

HAL's main costs are CPU power for hashing packets, and space for storing the resulting hashes. Assuming a per-packet storage requirement of 24 bytes and a hashing throughput of 100 Mbps, we have estimated that an *average* AS would need about 32 disks and 24 cores (more for larger ASes, less for smaller ones). Using a prototype implementation of a HAL monitor on commodity hardware, we have shown that these assumptions are realistic, and in the case of throughput even a bit conservative. In the absence of overload, our monitor also had no significant effect on quality metrics such as latency, jitter, or packet loss rate.

It is worth noting that there are many types of overhead HAL (unlike most comparable systems) does *not* have, and which we therefore have not evaluated. For example, HAL does not require any processing at core routers, it does not increase the size of packet headers, and it does not require any changes to existing protocols or applications.

7. DEPLOYMENT

Having shown evidence that an HAL deployment is technically feasible, we now consider the question how such a deployment might come to pass. Specifically, we ask 1) whether HAL can be deployed incrementally, 2) whether there are incentives for deployment, and 3) whether a full deployment is realistic.

7.1 Incremental deployment

Recall that HAL does not require *any* changes to existing protocols or applications. Therefore, an ISP could essentially deploy HAL today by installing our monitor software on an old workstation, by putting that workstation next to an edge router (DSLAM, CMTS, ...), and by connecting the workstation to a mirrored upstream port on that router. Such a deployment would immediately be useful; there would be no need to wait until other ISPs have deployed HAL or, indeed, to deploy the system on all access links at the same time.

7.2 Incentives for early adopters

The long-term benefit of HAL would be the ability to handle offending traffic more effectively. However, both customers and providers also have an immediate incentive for deploying HAL, even if they are the first adopters. Honest customers benefit because they are protected from false accusations, which (as we have shown) are not uncommon today and can have devastating consequences. Providers benefit because HAL helps with complaint handling: with HAL in place, providers can ask complainants for a positive attestation before they even look at a complaint. If a positive attestation does exist, they are on much firmer ground when taking action against the offender.

Quantitatively, we must weigh these incentives against the price tag. While we cannot claim to be experts in ISP economics, we can at least roughly estimate the likely cost to get a ballpark figure. Let us assume the provider is a cable ISP. Recent studies of broadband traffic [5, 24, 33, 34] have shown that the average upstream transmission rate of broadband customers is about 15 packets per second. If 100 customers share the same CMTS and hashes are stored for one month, about 96.4 GB of storage would be needed. Off-the-shelf mirrored RAID storage costs about \$0.30/GB. If we factor in the cost of a \$2k server blade, assume an equipment lifetime of five years with an annual disk replacement rate of 3% [30], and further assume that operational costs including energy and admin costs will double the hardware cost, we arrive at a rough estimate of $0.68 per customer per month.

7.3 Other deployment options

It is difficult to say how much customers would be willing to pay for the protection that HAL provides them personally. The situation is somewhat analogous to buying home insurance – events like burglaries are typically rare, but their effects can be devastating when they do occur. Therefore, we also discuss another potential deployment option.

Suppose users do not value the direct benefits of HAL enough to become early adopters, but suppose they *do* value the overall benefits of a widespread deployment – namely the ability to handle offending traffic more effectively. This would be a typical instance of a *market failure*: joining the system would produce positive externalities, but without remuneration for the joiner. Market failures are a well-known phenomenon in economics, and there is a variety of instruments that can be used to overcome them, such as subsidies or regulatory action. Whether or not such instruments should be used is a policy question, which we do not address here. We merely point out that relying on market forces is not the *only* way to deploy a system like HAL.

7.4 Is a full deployment realistic?

It seems safe to assume that, for most providers, the large majority of their customers are honest and would stand to profit from HAL. However, it is known that some providers send a lot more offending traffic than others; for instance, McColo provided hosting for major botnets [23]. So it is possible that, even if HAL were deployed almost universally, some providers would exploit a "business niche" by offering Internet access to senders of offending traffic. Such providers would clearly not deploy HAL voluntarily, and our assumption of providers do not collude with users would not hold for them.

However, we argue that such a situation would already be a huge success. If most of the world's senders of offending traffic were concentrated at a small number of providers, it would become possible to set up special cases for these providers, e.g., by blocking or rate-limiting their traffic. If they continued sending offending traffic, it seems likely that many of them would eventually share McColo's fate, which was disconnected by its upstream providers in November 2008.

8. CASE STUDY

To demonstrate the usefulness of HAL, we built an automated Digital Millennium Copyright Act (DMCA) complaint handling system that relies on HAL to validate complaints against the ISP's customers.

8.1 The problem

The DMCA includes a provision that allows copyright owners to file complaints regarding copyright infringement. ISPs that want to take advantage of certain protections against legal action regarding such copyright violations must be willing to handle and act upon these complaints. The larger the volume of complaints, the more costly it is for the ISP to both validate the complaints and verify claims from their customers that they are being falsely accused.

To gauge the volume of complaints that are received by large edge networks today, we obtained data from two large U.S. universities, each with more than $20,000$ students. Each receives more than $2,000$ complaints per year. One university provided us with a detailed breakdown, which shows that more than 80% of the complaints are DMCA notices. These are generated by sites like BayTSP, which, on its web site, reports sending more than one million DMCA takedown notices each month.

We also confirmed that the universities we contacted take some steps to validate the complaint, e.g., to rule out MAC spoofing. Further, users do occasionally dispute the evidence and assert their innocence; this again illustrates the two problems we described in Section 3.2 and the resulting dilemma for the operator.

8.2 Using attestation to validate complaints

We implemented a web-based service that enables ISPs to automatically process the DMCA complaints they receive. Our service uses HAL to validate these complaints, thus avoiding false accusations against its customers. The implementation consists of 121 lines of HTML, 92 lines of PHP scripts, and 565 lines of C code.

Our complaint handling service provides a web interface for copyright holders to upload not only the standard information that is required by law, but also a (set of) packet(s) that show the alleged offense (e.g., a few packets that contain part of the copyrighted material). Recall that most complaints are filed by automated services even today. These services can easily provide such evidence, e.g., by capturing packet traces on their honeypots; administrators could do the same using tools such as `tcpdump`.

The packets that are part of the complaint are then attested automatically using HAL, and in case the attestation is positive the provider can be confident in taking action against the user. Thus our system is advantageous both for the users, who are protected from false accusations, and especially for the provider who is able to (1) avoid an expensive and error-prone manual verification upon receiving the request, (2) handle innocence claims from his customers more effectively, and (3) avoid false accusations against its customers, which are awkward for provider and customer alike.

9. DISCUSSION

9.1 HAL and anonymous communication

At first glance, it might seem that HAL removes a legitimate (albeit weak) degree of user anonymity in the current Internet. There are two things to note here. First, the reality is that the current Internet provides *neither* useful anonymity *nor* useful evidence of packet origin at the network level. Source addresses can be mapped to individuals and cause suspicion and embarrassment; but by themselves they are not strong enough to justify serious measures against the suspected source, and may not hold up in a court of law.

Second, HAL provides a form of nonrepudiation at the IP layer. However, it does not affect anonymity (or lack thereof) at the application (session) level. Anonymous end-to-end communication can be implemented at the application level through the use of an anonymizing overlay network, regardless of whether HAL is in place. In such an overlay, HAL can establish whether a packet was sent by a particular overlay node, but it cannot establish whether that node originated the packet.

9.2 Extension: Traffic capture

HAL can confirm that a given packet originated from an access link, i.e., it corroborates available information about offensive Internet traffic. It does not show by itself which packets were sent or received by a given subscriber. Such information would be useful, for instance, to show that a subscriber's computer was compromised and used as part of a multi-stage attack.

To provide this information, HAL could be combined with traffic logging on customer premises equipment (CPE), e.g., a home router. This combination preserves subscriber's privacy yet provides valuable evidence that can show a subscriber's innocence and allow law enforcement agents to follow the trail to the upstream source of a multi-stage attack. Subscriber privacy is protected because the subscriber can decide if, when and what information to disclose, and control how long to store the information in order to limit exposure in case of a subpoena. Yet, a malicious subscriber cannot fabricate false evidence because the packets stored on the CPE can be validated by the hashes stored independently by the network access provider.

Traffic capture on CPE requires that the network access provider stores hashes of both upstream and downstream traffic, doubling the storage required. Moreover, the CPE must store full traffic information. Behavioral studies of residential broadband customers [5] have shown that on average customers transfer very modest amounts of data (e.g., downloaded 862 MB per day in 2008) allowing multiple day's worth of traffic to easily fit in common hard disks. But even in the extreme case of a customer fully utilizing a high-end broadband connection with 100 Mbps, the customer would only be able to transfer around 1 TB of traffic per day which still allows one day worth of traffic to be stored in a single commodity hard disk.

Despite its costs, this type of traffic capture may be worthwhile, given that it provides network access providers and law enforcement agencies additional evidence to investigate multi-stage attacks, and enables customers to clear their names if they fell victim to a multi-stage attack. A full study and evaluation of this and related approaches remains as future work.

10. CONCLUSION

In this paper, we proposed packet attestation. Packet attestation corroborates available evidence about malicious activity in the Internet, by verifying whether a specific packet originated from a particular access link. Packet attestation enables swift action to stop an ongoing attack and can convince ISPs to take action against a subscriber while protecting falsely accused subscribers. In the case of a multi-stage attack, packet attestation provides a starting point for an investigation of the ultimately responsible party. Moreover, packet attestation can be combined with traffic logging on customer premises equipment to provide additional evidence in the case of multi-stage attacks.

We think that packet attestation occupies a sweet spot in the trade-off between the strength of its evidence on one hand, subscriber privacy and deployability on the other. Packet attestation protects individual privacy since it merely confirms what the querier already knows. Moreover, packet attestation does not require changes to either the network core or to end systems, and can be deployed at reasonable cost today.

11. REFERENCES

[1] D. G. Andersen, H. Balakrishnan, N. Feamster, T. Koponen, D. Moon, and S. Shenker. Accountable Internet protocol (AIP). In *Proc. SIGCOMM*, 2008.

[2] K. Argyraki and D. R. Cheriton. Active internet traffic filtering: Real-time response to denial-of-service attacks. In *USENIX Annual Tech. Conf.*, Apr. 2005.

[3] H. Ballani, Y. Chawathe, S. Ratnasamy, T. Roscoe, and S. Shenker. Off by default! In *Proc. of HotNets*, Nov. 2005.

[4] T. Bates, P. Smith, and G. Huston. CIDR report. `http://www.cidr-report.org/as2.0/`.

[5] K. Cho, K. Fukuda, H. Esaki, and A. Kato. Observing slow crustal movement in residential user traffic. In *Proc. ACM CoNEXT*, 2008.

[6] Cisco Systems Inc. Cisco visual networking index: Forecast and methodology, 2009–2014. White paper, `http://www.cisco.com/en/US/`

solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360.pdf, June 2010.

[7] D. Clark and S. Landau. The problem isn't attribution: it's multi-stage attacks. In *Proceedings of the Re-Architecting the Internet Workshop*, ReARCH '10, 2010.

[8] D. Clark and S. Landau. Untangling attribution. In *Proceedings of a Workshop on Deterring CyberAttacks: Informing Strategies and Developing Options for U.S. Policy*, 2010.

[9] D. Farinacci, V. Fuller, D. Meyer, and D. Lewis. Locator/ID separation protototcol (LISP). Expired Internet draft, http://tools.ietf.org/html/draft-farinacci-lisp-12, Mar. 2009.

[10] P. Ferguson and D. Senie. RFC 2827: Network ingress filtering: Defeating Denial of Service. 2000.

[11] C. Gaither. Recording industry withdraws suit. Boston Globe, http://www.boston.com/business/articles/2003/09/24/recording_industry_withdraws_suit/.

[12] S. Gaudin. Spoofing defense dissed by security experts. InformationWeek, http://www.informationweek.com/news/security/cybercrime/showArticle.jhtml?articleID=189500626, June 2006.

[13] S. Guha, P. Francis, and N. Taft. ShutUp: End-to-end containment of unwanted traffic. Technical Report http://hdl.handle.net/1813/11101, July 2008.

[14] A. Haeberlen, R. Rodrigues, K. Gummadi, and P. Druschel. Pretty good packet authentication. In *Proc. HotDep*, Dec 2008.

[15] F. Huici and M. Handley. An edge-to-edge filtering architecture against DoS. *SIGCOMM Comput. Commun. Rev.*, 37(2), 2007.

[16] India News. Techie jailed due to Airtel mistake. http://www.indiaenews.com/technology/20071103/78777.htm, Nov. 2007.

[17] A. Juels and J. G. Brainard. Client puzzles: A cryptographic countermeasure against connection depletion attacks. In *Proc. NDSS*, Feb. 1999.

[18] T. Killalea. RFC 3013: Recommended Internet service provider security services and procedures. Nov. 2000.

[19] D. Kravets. Defense planting seeds of doubt with RIAA jurors. Wired, http://www.wired.com/threatlevel/2007/10/defense-plantin/, 2007.

[20] J. Li, M. Sung, J. Xu, and L. Li. Large-scale IP traceback in high-speed Internet: Practical techniques and theoretical foundation. In *Proc. IEEE Security & Privacy*, May 2004.

[21] X. Liu, A. Li, X. Yang, and D. Wetherall. Passport: Secure and adoptable source authentication. In *Proc. NSDI*, Apr. 2008.

[22] X. Liu, X. Yang, and Y. Xia. NetFence: preventing internet denial of service from inside out. In *Proc. of SIGCOMM*, 2010.

[23] J. Ma. *Learning to Detect Malicious URLs*. PhD thesis, UC San Diego, 2010.

[24] G. Maier, A. Feldmann, V. Paxson, and M. Allman. On dominant characteristics of residential broadband internet traffic. In *Proc. IMC*, Nov. 2009.

[25] M. McConnell. Mike McConnell on how to win the cyber-war we're losing. The Washington Post, February 28, 2010.

[26] M. O'Dell. 8+8 - An alternate addressing architecture for IPv6. Exp. Inet draft, http://bgp.potaroo.net/ietf/all-ids/draft-odell-8+8-00.txt, 1996.

[27] M. Piatek, T. Kohno, and A. Krishnamurthy. Challenges and directions for monitoring P2P file sharing networks. In *Proc. HotSec*, July 2008.

[28] J. Rosenblatt. Fully automated DMCA processing. Presentation at the Security Professionals Conference (SEC), 2009.

[29] S. Savage, D. Wetherall, A. Karlin, and T. Anderson. Practical network support for IP traceback. In *Proc. SIGCOMM*, Aug 2000.

[30] B. Schroeder and G. A. Gibson. Disk failures in the real world: what does an MTTF of 1,000,000 hours mean to you? In *Proc. FAST*, Feb. 2007.

[31] K. Shanmugasundaram, N. Memon, A. Savant, and H. Bronnimann. Fornet: A distributed forensics network. In *Comp. Netw. Security*, 2003.

[32] M. Shaw. Leveraging good intentions to reduce unwanted network traffic. In *Proc. SRUTI*, 2006.

[33] M. Siekkinen, D. Collange, G. Urvoy-Keller, and E. W. Biersack. Performance limitations of ADSL users: A case study. In *Proc. PAM*, 2007.

[34] R. Sinha, C. Papadopoulos, and J. Heidemann. Internet packet size distributions: Some observations. Technical Report ISI-TR-2007-643, Univ. Southern Calif., Info. Sci. Inst., May 2007.

[35] A. C. Snoeren, C. Partridge, L. A. Sanchez, C. E. Jones, F. Tchakountio, B. Schwartz, S. T. Kent, and W. T. Strayer. Single-packet IP traceback. *IEEE/ACM Trans. Netw.*, 10(6), 2002.

[36] R. Sylvester. IP address typo leads to a false arrest in Kansas. The Wichita Eagle, http://www.kansas.com/mld/eagle/news/local/crime_courts/12620843.htm.

[37] UPnP forum. Internet gateway device standard v1.0. Available from http://www.upnp.org/, Nov. 2001.

[38] Y. Xie, F. Yu, and M. Abadi. De-anonymizing the Internet using unreliable IDs. In *SIGCOMM*, 2009.