Evasive Internet: Reducing Internet Vulnerability Through Transient Addressing

Michael Rabinovich
Case Western Reserve University
misha@eecs.cwru.edu

Oliver Spatscheck
AT&T Labs–Research
spatsch@research.att.com

***Abstract*— This paper presents our vision for *Evasive Internet*, where destinations are only reachable through *capabilities*, which serve as hosts' flat transient addresses. Just as today's host addresses, our capabilities are obtained from the DNS hierarchy, thus never exposing destinations themselves to unprotected traffic. Our design supports in-network authentication of transient addresses and attribution of traffic they generate; our design further gives hosts full control over incoming flows. We achieve these objectives without exposing hosts to unprotected capability request traffic and without distributed filtering infrastructure. While significant work is needed to flesh out our vision, we hope it will contribute to improving security in future networks.**

I. Introduction

As Internet continues to permeate through every aspect of our lives, the stakes are rising in securing it against a variety of subversive behaviors. Sender's accountability over generated traffic [3], [21] and recipient's control over incoming flows [6], [1] have emerged as recognized objectives for Internet design that can improve its security. An influential approach to achieve these objectives is through a notion of *capability*, which represents authorization to communicate with a host, and which is valid only for a specific sender and for a limited amount of time. Capabilities have been used in several systems [25], [19], [24], [4]. While significantly improving network security, most capability systems are vulnerable to the *denial of capability* attack, where the attackers are trying to exhaust the unprotected capability request channel [5].

This paper proposes a new capability-based approach based on two key ideas. First, capabilities are the *only* means for hosts to be reached; in particular, hosts do not handle capability requests. Second, capabilities are distributed by the DNS system rather than recipient hosts or distinct infrastructure. Because hosts can only be reached through capabilities, a packet sender lists its capability in place of the IP address as its source address. Similarly, DNS servers themselves are only reached through their own capabilities, issued by higher-level servers in the DNS hierarchy in the same way current higher-level servers return name server IP addresses. Thus, capabilities become in effect the hosts' addresses, which are *transient* due to the limited validity of the capabilities.

By relegating capability assignment to DNS, our approach leaves the root DNS servers exposed since as in today's Internet, they must be always reachable and hence offer well-known permanently valid capabilities. Some sort of residual vulnerability is inevitable in any system for bootstrapping security [23]. However, as foundational enablers in today's Internet communication, root DNS server are already a frequent target, and while protecting them is not an easy task, their operators have demonstrated these servers – unlike the rest of today's Internet – can be protected successfully. Thus, we believe narrowing the scope of vulnerability to the root DNS servers is a sensible design choice.

Our architecture achieves sender's accountability because it forces the sender to include its proper capability in the packet header – otherwise the packet will be blocked by the first compliant router [1]. Our architecture enables recipient's control because each host can implement fine-grained capability-issuing policies for particular external destinations: By communicating with its authoritative DNS server (or, in the case of a DNS server, with its higher-level DNS server), the host can shorten the validity period of its capability for suspicious senders or disallow certain senders altogether. Note that this would make the host and all its applications and ports unreachable to these senders. Again, any attempt by blocked senders to reach the host will be terminated by the first compliant router, shielding most of the network from a potential DoS attack on a link. This has the same effect as a remote filter in filtering systems ([15] and references therein) but without propagating filter requests.

We named our approach Evasive Internet Protocol (EIP) to reflect its lack of permanent means of reaching a host. While seemingly radical, Evasive Internet could be deployed incrementally using an approach similar to existing technologies for migration from IPv4 to IPv6, such as Microsoft's Teredo [22]. These technologies combine overlay and tunneling techniques to enable co-existence of a new and current Internet version.

Realizing our idea presents interesting questions. How would it affect BGP scalability? The routing table sizes have reached 250,000 entries and are growing rapidly [8]; making addresses flat and temporary would seem to exacerbate this situation. How would the new addresses affect DNS? DNS has held up astonishingly well to the explosive Internet growth, due in large part to caching, but source-specific addresses would seem to limit their cacheability. We address these and other issues in the rest of this paper.

II. The Architecture

We now sketch the main architectural components of the Evasive Internet we envision.

A. Addressing

Our architecture involves two concepts to represent destinations on the Internet – IP addresses and capabilities. The

The work of M. Rabinovich on this project is supported in part by the NSF under grants CNS-0831821 and CNS-0916407.

[1]See Section III for a scenario where a host "steals" someone else's capability.

IP addresses are still used in routing algorithms for route computation and forwarding table indexing. This allows EIP to inherit existing routing protocols (e.g., BGP) and to retain their scalability properties that stem from the topological information embedded in IP addresses. In fact, EIP can benefit from recent proposals to improve aggregation and provider independence of IP addresses by splitting them into identifiers and locators (see [20] and references therein), in which case IP addresses would be replaced with locators throughout the design below.

Although IP addresses are used for route computation and forwarding tables, the host IP address *cannot* be used to communicate with the host. A compliant router will only forward a packet with a valid destination capability, which in effect becomes a transient destination address and which we refer to as a *t-address*.

The host t-address contains the following fields[2]: destination IP address (16 bytes[3]); source IP address (16 bytes); validity constraint (10 bytes); signature generated by the destination or authoritative name server using its private key (approximately 256 bytes); the certificate chain from the root name server used by the destination (approximately 1KByte per certificate). The whole address is signed with the destination's or the authoritative name server's private key.

The validity constraint is set to 10 bytes: 8 bytes for a timing constraint and 2 bytes for the number of bytes constraint. The 8-byte time constraint field is sufficient to carry a UNIX-like time stamp without the fear of wrapping. Such a time stamp can be used to time out t-addresses with one second granularity. The 2-byte size constraint represents the maximum amount of traffic that can be sent before the capability expires. Thus, a valid capability must satisfy both constraints. The addressing and validity information is signed, and all necessary certificates (typically 2-3) are attached to the t-address. This leaves us with a t-address size in the 2.5-3.5KByte range. While this size is too large to be transmitted without fragmentation in today's networks, which often have a 1500Byte MTU, it is a fraction of the packet sizes allowed in newer network technologies. We discuss this and a related issue of header overhead in EIP datagrams later in this section.

B. Datagrams

Our current design calls for three types of datagrams in the Evasive Internet protocol. It assumes loosely synchronized clocks among hosts and routers on the Internet. Note that properly configured hosts have clocks that are well synchronized in practice within the order of milliseconds. Improperly configured hosts will be penalized in our architecture, which may actually alert their operators about the issue.

Type-1 datagrams are the initial datagrams that open a communication between two hosts. Their header (we discuss only addressing-related fields) includes the source and destination t-addresses, an optional *delegation* token, whose purpose will become clear in Sec. II-E3, a flow ID (FID), a flow time-to-live (TTL), and a signature signing both FID and TTL. The source and destination t-addresses (transiently) identify the end-points of the communication. An (FID) is a 128-bit nonce (a "number used only once" with high probability) that identifies this communication in subsequent packets. Flow TTL tells a router processing this datagram how long the flow ID should remain valid. Flow signature signs the FID and TTL with the sender's public key, allowing verification of the voracity of the corresponding fields.

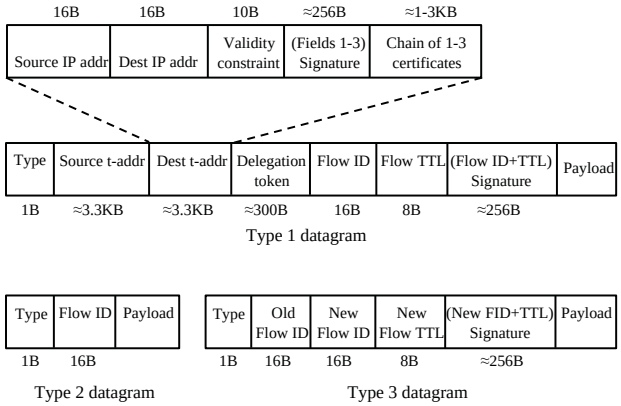

Fig. 1. Datagam structure.

Because t-addresses include certificate chains, the initial datagrams have a large header - in the order of several kilobytes. Consequently, type-2 datagrams amortize the header overhead over subsequent communication by including only a flow ID. Our current design assumes these flow IDs are sent in the clear for efficiency, relying on short lifetime of these FIDs for a "good enough" protection against an injection of fraudulent packets with the same FID by man-in-the-middle. This is only a specific design choice reflecting our view on the current costs/risks landscape. We sketch an alternative design, trading efficiency for better protection and reflecting one possible trend on computer development, below. The address-related header size of type-2 datagram is only 16 bytes for the FID; we assume that these datagrams will carry most of data exchanges.

To ensure uninterrupted communication, the sender must replace the current FID with a new one before the current FID expires (and the sender knows the expiration time because it assigns the TTL). The replacement is accomplished with a type-3 datagram type, which lists the current FID and the new signed FID and TTL. Note that whenever a new FID is introduced, it is signed to ensure its integrity. This is important because this information affects security for the entire next TTL period, and it incurs acceptable computational overhead at the sender because the sender must generate the signature only once every TTL interval. Our current assumption is that flow TTL would be in the order of 5-10 seconds, which is several orders of magnitude more than typical local clock drift. The address-related header size of type-3 datagram is approximately 300 bytes (256 bits for the two FIDs, 8 bytes for the new TTL, and around 256 bytes for the signature).

Finally, before its t-address becomes invalid, the sender may choose to include a subsequent-use t-address with its next packet, if it is willing to continue the communication.

[2]We may introduce additional fields as we flesh out our design in the course of this work; also, several fields could use more careful tightening but we are presenting a first-cut design.

[3]We allocate 16 bytes to IP addresses to stress the compatibility of our design with IPv6 addresses.

This is done with an optional header field in type-2 or type-3 datagrams, adding up to 3.5KB (t-address size) to the header. The subsequent-use t-address enables the source to have a long-running communication with the destination without re-acquiring a new t-address from the name service. This allows t-address distribution by the name service to remain application-oblivious without adding the overhead of repeated name resolutions.

Alternative design: As mentioned, the above design was driven by our desire to reduce the frequency of signature generation by senders; this is a computationally expensive operation, and requiring every sender (e.g., a high-volume web server) to generate a signature on every packet would be taxing. However, hardware-implemented signature generators are becoming available. Once their cost decreases enough, we can simplify the protocol and make it more secure as follows.

We will no longer use flow ID TTL or type-3 datagrams; the flow ID does not expire. Instead, every type-2 datagram includes the current timestamp, and the signature of the FID and timestamp pair generated using the sender's private key. The signature ensures the packet could only originate from the same destination that issued the initial datagram. The timestamp makes it extremely unlikely that a signed FID could be replayed by a man-in-the-middle. Indeed, a router would drop any message older than a reasonable estimate of the maximum network message delay, forcing the sender to resend the packet with full type-1 headers (see Section II-C for details). This drastically narrows the window of opportunity for the attacker to inject a fraudulent packet, and hence the duration of a possible attack.

C. Routing

By using flat transient addressing, Evasive Internet brings a challenge of keeping the size of routing tables and updates manageable. Our approach to this challenge utilizes two distinct means of representing Internet destinations. In Evasive Internet, route computation and maintenance is all done using destination IP addresses. In particular, routing tables will be keyed and aggregated based on destination IP addresses. However, packet forwarding utilizes t-addresses: no legitimate router will forward a packet addressed with the destination IP address - the valid t-address is required.

Routers rely heavily on caching to achieve acceptable forwarding performance. When processing a type-1 datagram, a router extracts the destination IP address from the t-address, verifies that the t-address is still valid and if the source IP address from the source t-address matches the one from the destination t-address. If the above checks are satisfied, the router looks up the routing entry for the destination's IP address and forwards the packet accordingly. Finally, the router caches the state reflecting the end-point IP addresses and t-address and flow expiration. Otherwise, the router drops the datagram. The cached entry is removed upon the flow expiration unless it is replaced with a new FID in a type-3 datagram. The cache is indexed by the flow ID from the datagram as well as by a combination of source and destination IP addresses extracted from the t-addresses (for the reasons to be clear shortly).

On receiving a type-2 datagram with a non-expired FID, the router looks up the corresponding cached flow entry by the FID and forwards the datagram accordingly. The router also updates the number of bytes transmitted in the flow. If the router cannot find the cache entry for the received FID, it drops the datagram. Thus, any route change during an ongoing communication will result in a dropped datagram, which would presumably be retransmitted by the higher-layer protocols at the end-hosts using type-1 datagram. On receiving a type-3 datagram with a non-expired old FID, the router updates its corresponding cached entry with the new FID, increments the number of bytes in the flow (so that the overall size constraint for the capability can be tracked), and extends the flow retention time. While processing type-1, and to a less extent, type-3 datagrams is expensive, most traffic will be carried by type-2 datagrams, which require trivial processing.

When the router receives a type-1 datagram, before creating a cache entry for a new flow, the router extracts the source and destination IP addresses and checks if there is an existing flow for these end-points (hence double-indexing of the cache). If so, the router compares the timestamp of the capability from existing flow with that of the current datagram. If the timestamps match, the new type-1 datagram does not represent a new authorization from the destination. Consequently, the router simply replaces the FID in the existing cache entry with the new FID from the datagram and increments the number of bytes in the flow. Thus, the sender cannot circumvent the size restriction by sending extra type-1 datagrams. If the new timestamp is greater, the new capability represents a new authorization and the router resets the byte count in the flow entry with the payload size in the datagram. In either case, the new type-1 datagram updates the state in the existing flow entry as long as one exists. Thus, a router maintains at most one flow entry for a given end-point pair, preventing a sender from attacking the router cache space by sending multiple type-1 datagrams. Furthermore, a router follows the elegant mechanism from [25] to utilize the capability size constraint to keep up with the line speed with a fixed cache size (we refer the reader to [25] for details).

One consequence of the above design is that a capability represents a common authorization to send traffic for all applications on a host, with the constraints shared among all the applications. This is a consciously conservative design choice. In the absence of misbehaving applications, the recipient will extend its authorization by issuing a subsequent-use t-address before the current authorization becomes invalid. A misbehaving application may penalize other applications running on the same host. Note also that flows in our approach are uni-directional, and so asymmetric routing is supported naturally: the flows in either direction would simply be cached by the routers on the corresponding path.

An advantage of this scheme is that invalid or illegitimate packets are discarded early (at the first router to detect the problem). Even if all routers on the routing path to the destination collude to forward the packet without a valid t-address, this packet will be stopped by the the destination's ISP routers or, as the last resort, quickly discarded by the destination itself

before the packet is ever passed to the transport layer or above. However, such multiple router collusion is highly unlikely.

D. MTU Limitations

Depending on the number of certificates in source and destination t-addresses, the size of type-1 datagram header can reach 7K. While this size is too large to be transmitted without fragmentation in today's networks, which often have a 1500Byte MTU, it is less than the packet sizes allowed in newer network technologies. In fact, most core networks utilizing 10GETH and OC3-768 technologies already support jumbo Ethernet frames with an MTU of 9000 Bytes. For access networks, the only thing that prevents jumbo Ethernet frames is relatively low bandwidth (which can cause jitter in real-time communication due to the transmission delay of a competing frame). However, Cisco routers provide standard support for *link fragmentation and interleaving* (LFI) [13]. With this technology, the MTU will still show the full Ethernet frame, and the fragmentation is only limited to a single link (the fragment is reassembled by the router on the other side of the link). Given that even DSL is moving away from ATM and towards Ethernet for the last mile, and that LFI reassembly would only be required for type-1 datagrams (limiting routers' LFI processing costs), EIP datagrams are feasible with most of today's access links.

Furthermore, Cisco's rule of thumb is to keep frame transmission delay within 10ms. Assuming this is the right guideline, a jumbo frame can be serialized in 10ms on a roughly 9Mbps link or higher. Residential broadband is clearly moving into this territory and beyond. Indeed, Korea is planning to provide country-wide gigabit access by 2012 [14]. Cablevision offers 101Mbps downlink and 15Mbps uplink access [12]. Similar speeds are offered by Comcast [16] and Verizon. With these speeds, one can support jumbo frames straight, without LFI. The same considerations also apply to modern wireless links which offer bandwidth of 54Mbps and beyond.

So, even if today's home access networks do not support jumbo frames, there seems to be no technological reason to prevent them, and we assume they will be available by the time t-addresses would be introduced natively at large scale.

E. Name Service

Many attempts at securing the Internet through mandating authenticated communication were stymied by the need for key distribution infrastructure. We sidestep this thorny issue by leveraging the Internet name system, which is already a crucial component in the current Internet infrastructure, for distributing t-addresses (and certificates) without any additional infrastructure.

Our approach relies on everyone being able to authenticate the entity that issues a t-address. We piggyback the certificate authority hierarchy needed for authentication with the name service hierarchy. Most organizational networks today operate both local and authoritative DNS servers (in fact, it is common that the same server fulfils both tasks). Our architecture formalizes this practice and assumes that *every* subnet belongs to an authoritative name server hierarchy.

1) Bootstrapping: Root name servers act as top-level certificate authorities, which delegate their authority to lower level servers if any (such as top-level domain name servers in today's DNS). When an authoritative name server registers with a higher-level name server, it obtains from it a separate certificate for each range of destination IP addresses that it will be handling (the certificate encrypts the IP address range). When a host connects to the network, as part of the initialization, it obtains a certificate from its authoritative name server that delegates the host the authority to generate its own t-addresses. Whether it is an authoritative name server that constructs and signs a t-address for a destination, or a host itself issues a subsequent-use t-address, anyone can verify that the issuer is authorized to mint t-addresses for the host with the given IP address and trace the authority to the corresponding root server. Note that, as Needham and Schroeder point out in their classic paper, this only means that the entity (and t-addresses it returns) is as trustworthy (or as suspicious) as the root server that ultimately certified it [18]. We refer to the path from the hosts to their authoritative name server to the roots as the registration path.

Registration path represents bootstrapping of t-addresses, and hence registrations themselves require out-of-band communication. We do not consider the protocols for such communications in this paper; they can take different forms, from a separate secure network control protocol over raw IP addresses to a non-computer communication such as obtaining a compact disk with necessary certificates via postal service. In all cases, because registration involves a delegation of trust, it typically involves human interaction.

Once the registration process is complete, the root servers can return a t-address of the authoritative server for a domain name, the authoritative server can return a t-address for a host within its domain, and a host can generate its own t-address as a source address when initiating communication or as a subsequent-use t-address.

2) Root Name Servers: Root name servers must be well-known and reachable by any host just in today's Internet. Thus, they must have an infinite expiration and hence remain a permanent target for the attackers. This residual corner of vulnerability is necessary to bootstrap the system. However, the scope of this vulnerability in our architecture is limited to just a few well-administered addresses. It is feasible to defend them well, and even build address-specific defences into the Internet, given their small number (currently there are only thirteen root DNS server IP addresses). To our knowledge, after major ISPs started using anycast to access root DNS servers, they have not seen any outages due to security issues at root servers.

3) Obtaining a Destination T-Address: We now describe the general procedure for communication between two hosts. A host that wishes to communicate with another host must obtain that host's valid t-address. To this end, the source host generates its own source t-address and sends a DNS query with the destination hostname to the source's local name server (LDNS). The local name server routes the query through name service infrastructure on *its own behalf*, that is, by providing its own t-address as the requester. Thus, when obtaining the

t-address on behalf of its clients, the LDNS will receive the t-address specific to its own (not the client's) IP address. The name server will then give this t-address to the source host (and to subsequent clients until this t-address expires).

Of course, when the source now sends a type-1 datagram opening the communication with the destination, the IP address from the source t-address (which is the source host's IP address) will no longer match the source IP address embedded into the destination t-address (which is the LDNS's IP address). We bridge this gap with the help of delegation tokens. When returning a non-client-specific t-address, the local name server will also give the client a signed delegation token, which includes both the client and the name server IP addresses. The size of this token is approximately 300 bytes (32 bytes for the two IP addresses and around 256 bytes for the signature[4]). The client then includes this token into its type-1 datagram to the destination. Routers will use this token to verify proper delegation of the destination t-address.

In today's DNS, hostname-to-IP mappings are cached by both client machines (sources) and their local DNS servers. In particular, local DNS servers provide a shared cache that can benefit a potentially large number of clients. It might seem that Evasive Internet would defeat the shared cache of local DNS servers because it makes destination t-addresses specific to the client's IP address. Fortunately, the above procedure reveals this is not the case.

Note that with LDNS caching, the validity constraint in t-address is shared among all clients using this t-address. Thus, the destination's policy for validity constraint assignment becomes coarser grained, in a sense that it applies to all the clients behind the LDNS server. This introduces an interesting trade-off between allowing LDNS caching and coarsening the granularity of the validity constraint, which remains to be explored as future work.

III. Security

We already discussed the advantages of transient addresses and relegating their distribution to the DNS system (which removes the unprotected capability request channel from paths to the destinations thus limiting the vulnerability to denial of capability attacks).

Other security properties of Evasive Internet stem from the following observation. Unless the attacker colludes with the destination or eavesdrops on the network (see below), it can only acquire a valid destination t-address by revealing its own identity. Indeed, the sender can only obtain a non-expired t-address by requesting it from the name service. To get the response, the attacker needs to supply the name service with its valid source t-address, including its identity. However, this same identity will be embedded into the unforgeable destination t-address returned. Some of the consequences of the above observation are as follows.

An attacker cannot fake its identity: such a packet will be discarded at the first router. This protection is much stronger than current ingress filtering that (a) is in imprecise science, except for stub networks, and (b) gets in the way of innovation such as mobile IP.

The attacker can "amplify" its identity by sending packets with the same identity from multiple hosts. However, the effect of this would be limited because (a) the current destination t-address will expire in a short time, and subsequent t-address requests by the same requester can be penalized (see Sec. V for more details); (b) the return packets will all be delivered to the original host (consuming its resources and making it the bottleneck, especially in application-level attacks that require the sender's continued participation); and (c) the original host will be traceable. Furthermore, a router can verify the rightful owner of the source t-address with a mechanism similar to [3].

If the attacker gains network eavesdropping capability, it can attempt a replay attack, by resending packets observed on the network. The damage from such an attack would be mitigated by the fact that flows and destination t-addresses quickly expire. The attacker would need a *continued* eavesdropping capability over a network link with *continued* legitimate traffic to the destination to launch a sustained attack against the destination. Thus, Evasive Internet would significantly narrow the scope of replaying attacks.

IV. Related Work

The literature on network and host security is too large to survey here. This section touches upon some prior work that is most related technically to our approach.

We use capabilities as transient addresses. Existing capability systems [25], [19], [24], [4] include an unprotected communication channel to hosts for initial capability requests; these request reach hosts using their IP addresses. In Evasive Internet, DNS is used to distribute capabilities, thus end-hosts are never exposed. We also leverage the DNS hierarchy for distributing authentication certificates to all parties involved. The SANE architecture [7] uses separate network components to issue capabilities in the context of enterprise networks. By operating at a higher layer, Evasive Internet offers less hard-and-fast protection but targets public Internet and does not require new components. In fact, our approach does not preclude a SANE deployment within an enterprise.

Filtering approaches ([15] and references therein) improve network security by allowing hosts to install remote filters blocking identified malicious senders. They rely on a separate filtering mechanism, which itself needs to be carefully protected [15]. Several other approaches propose to improve security by leveraging separate rendezvous points [2], a name-routed signaling infrastructure [9], or by isolating client and server address spaces [11]; other approaches utilize distributed hash tables for rendezvous server distribution [10]. Evasive Internet involves less ambitious architectural changes. The Host Identity Protocol [17] separates host identifiers from IP addresses and stores the former in DNS. Our approach goes beyond this as it is based on capabilities as *transient* identifiers and enables in-network host authentication.

There has been a lively debate in the research community on the relative merits of filter-based and capability-based approaches to network security. Argyraki and Cheriton pointed

[4]Note that we do not need a certificate chain here because it will be the same as the one from the source t-address.

out the vulnerability of capability systems to the *denial of capability* attack, where the attackers are trying to exhaust the unprotected capability request channel [5]. Parno et al. countered with an elegant scheme using proof-of-work to ensure that legitimate capability requests will eventually get through [19]. On the other hand, Liu et al. proposed a filter-based approach that protects the filter infrastructure by using a separate closed-control plane for filter requests [15]. By relegating capability assignment to DNS, our approach narrows the scope of vulnerability to the root DNS servers (see Sec. II-E). Protecting these servers is still not easy, but it is much simpler than protecting the entire Internet.

Several approaches address network security by enabling packet source attribution and accountability [3], [21]. These are key capabilities in the network defence arsenal. Our approach also allows attribution but goes further by preventing hosts from being permanent targets for attacks.

V. Further Thoughts: Destination Rating

While we currently assume a trivial t-address expiration policy, simply as a constant validity time, we can envision much more elaborate policies in the future.

Authoritative name servers maintain close relationship with destinations they represent - they are either operated by the same network or by an external service with a business relationship with the destinations' network. We can exploit this relationship to implement a fine-grained control over the t-address expiration, with the goal of improving scalability without hurting security. When a name resolution query arrives from a new requester, the authoritative name server can assign a low expiration time to the t-address to keep the security risk low. However, depending on the nature of the ensuing communication with the corresponding host, the host can often determine the clients that exhibit "expected" behavior. In these cases, the host can (a) include the optional subsequent-use t-address with its own packets to the client so that the client can extend its communication without going back the the name service (as discussed earlier), and (b) assign an appropriate *rating* to this client and communicate this rating to the authoritative name server. The authoritative name server can store ratings from more active clients and return t-addresses with longer expiration to clients with high rating. Note that, as explained in the previous section, the client rating may in fact apply to a set of hosts from the same network. Further, similar to capabilities, the client rating applies to all applications running on the client host, so a misbehaving application will affect other co-located applications. We believe this is an appropriate consequence of running a misbehaving application.

Client rating is somewhat similar to reputation in reputation systems except its scope is limited to the destination: it reflects the local experience of the destination site with this client and it is interpreted and acted upon by the destination site. Because of that, it is free from typical issues in reputation systems, such as data pollution, privacy, and trust. The destination site has full authority to assign and interpret client ratings in any way it sees fit. While purely a heuristic-driven performance hint, this mechanism could prove a powerful scalability tool in practice.

VI. Conclusion

The paper outlines our vision of an *Evasive Internet* - a network where destinations can only be reached through transient addresses. We achieve this by utilizing capabilities as the sole means of reaching hosts on the Internet. Our design leverages name service infrastructure to distribute hosts' capabilities as well as security certificates, which are used for authentication of transient addresses and traffic attribution and to ensure that destinations themselves control over their own address duration. While significant work is needed to flesh out our vision, we hope it will contribute to improving security in future networks.

References

[1] A. Seehra and J. Naous and M. Walfish and D. Mazieres and A. Nicolosi and S. Shenker. A policy framework for the future Internet. In *HotNets–VIII*, 2009.

[2] D. Adkins, K. Lakshminarayanan, A. Perrig, and I. Stoica. Towards a more functional and secure network infrastructure. Technical Report UCB/CSD-03-1242, UC Berkeley, 2003.

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

[4] T. Anderson, T. Roscoe, and D. Wetherall. Preventing Internet denial of service attacks with capabilities. In *HotNets-II*, 2003.

[5] K. Argyraki and D. Cheriton. Network capabilities: The good, the bad and the ugly. In *HotNets-IV*, 2005.

[6] H. Ballani, Y. Chawathe, S. Ratnasamy, T. Roscoe, and S. Shenker. Off by Default! In *HotNets-IV*, 2005.

[7] M. Casado, T. Garfinkel, A. Akella, M. J. Freedman, D. Boneh, N. McKeown, and S. Shenker. SANE: a protection architecture for enterprise networks. In *USENIX Security Symposium*, 2006.

[8] Growth of the bgp table - 1994 to present. http://bgp.potaroo.net/.

[9] S. Guha and P. Francis. An end-middle-end approach to connection establishment. In *SIGCOMM*, 2007.

[10] A. Gurtov, D. Korzun, A. Lukyanenko, and P. Nikander. Hi3: An efficient and secure networking architecture for mobile hosts. *Computer Communications*, 31(10):2457 – 2467, 2008.

[11] Mark Handley and Adam Greenhalgh. Steps towards a DoS-resistant Internet architecture. In *FDNA (ACM SIGCOMM Workshop)*, 2004.

[12] S. Hansell. Cablevision goes for U.S. broadband speed record. New York Times, 04/28/2009. http://bits.blogs.nytimes.com/2009/04/28/cablevision-goes-for-us-broadband-speed-record/.

[13] D. Hartmann. Cisco QoS: Link fragmentation and interleaving. Network World, 03/04/2009. http://www.networkworld.com/community/node/39221.

[14] R. C. Hodgin. Gigabit broadband coming to Korea by 2012. TG Daily, 02/03/2009. http://www.tgdaily.com/content/view/41292/103/.

[15] X. Liu, X. Yang, and Y. Lu. To filter or to authorize: Network-layer dos defense against multimillion-node botnets. In *SIGCOMM*, 2008.

[16] J. Meisner. Comcast revs its engine in broadband-speed race. Linux Insider, 12/11/2008. http://www.linuxinsider.com/story/trends/65472.html?wlc=1256158928.

[17] R. Moskowitz and P. Nikander. Host identity protocol (HIP) architecture. Request for Comments 4423.

[18] R. M. Needham and M. D. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–999, December 1978.

[19] B. Parno, D. Wendlandt, E. Shi, A. Perrig, B. Maggs, and Y.-C. Hu. Portcullis: protecting connection setup from denial-of-capability attacks. In *SIGCOMM*, 2007.

[20] B. Quoitin, L. Iannone, C. de Launois, and O. Bonaventure. Evaluating the benefits of the locator/identifier separation. In *MobiArch (ACM SIGCOMM Workshop)*, 2007.

[21] A. C. Snoeren, T. Kohno, S. Savage, A. Vahdat, and G. M. Voelker. Privacy-preserving attribution and provenance. www.nets-find.net/Funded/Privacy.php.

[22] Teredo Overview. January, 2003. http://www.microsoft.com/technet/network/ipv6/teredo.mspx.

[23] J. Touch. Personal communication.

[24] A. Yaar, A. Perrig, and D. X. Song. SIFF: A stateless Internet flow filter to mitigate DDoS flooding attacks. In *IEEE Symp. on Security and Privacy*, 2004.

[25] X. Yang, D. Wetherall, and T.E. Anderson. A DoS-limiting network architecture. In *SIGCOMM*, 2005.